BRASÍLIA 2023

ISBN 978-85-7267-134-7

Organização

WILLIAM ROBERTO MALVEZZI
ROBERTO SILVEIRA LEMOS

DATA SCIENCE EM REVISTA

CEUB
EDUCAÇÃO SUPERIOR

Organização

William Roberto Malvezzi
Roberto Silveira Lemos

DATA SCIENCE EM REVISTA

1ª Edição

Brasília
2023

CENTRO UNIVERSITÁRIO DE BRASÍLIA - CEUB
Reitor
Getúlio Américo Moreira Lopes

Diagramação
Biblioteca Reitor João Herculino

Disponível no link: repositório.uniceub.br

Dados Internacionais de Catalogação na Publicação (CIP)

Data science em revista. / organizadores, William Roberto Malvezzi; Roberto Silveira Lemos – Brasília: CEUB; ICPD, 2023.

311 p.

ISBN 978-85-7267-134-7

1 Data science. I. Centro Universitário de Brasília. II. Título.

CDU 004.62

Ficha catalográfica elaborada pela Biblioteca Reitor João Herculino

Centro Universitário de Brasília – CEUB
SEPN 707/709 Campus do CEUB
Tel. (61) 3966-1335 / 3966-1336

PREFÁCIO

É com imensa satisfação e entusiasmo que apresentamos a primeira edição da revista "Data Science em Revista", uma iniciativa pioneira dos cursos da área de Ciência de Dados do Centro Universitário de Brasília - CEUB. Esta revista é fruto do árduo trabalho e dedicação de nossa comunidade acadêmica, refletindo o espírito colaborativo e inovador que permeia nossos cursos e projetos.

A busca incessante pelo conhecimento e a paixão pela Ciência de Dados têm sido o norte para nossos estudantes e corpo docente. Nesta jornada, temos a felicidade de contar com um corpo docente altamente qualificado e comprometido, que atua como guia e mentor para os nossos estudantes. É graças ao esforço desses dedicados professores que nossos estudantes encontram o suporte e o estímulo necessário para se desenvolverem como pesquisadores e profissionais brilhantes na área de Ciência de Dados.

A coordenação dos cursos de graduação e pós-graduação em Ciência de Dados e Machine Learning, juntamente com a coordenação do curso Superior de Tecnologia em Marketing, tem sido fundamental na criação de um ambiente propício à pesquisa e ao desenvolvimento científico. O apoio incondicional desses líderes educacionais tem sido um catalisador para o crescimento exponencial do interesse na área de Ciência de Dados no Distrito Federal.

"Data Science em Revista" nasce como um espaço de compartilhamento do conhecimento e das descobertas que emergem dos projetos de pesquisa realizados em nosso centro universitário. Queremos que esta revista seja um veículo para a disseminação de pesquisas de ponta, trabalhos inovadores e projetos de qualidade desenvolvidos por nossos alunos e docentes.

Aqui, os leitores encontrarão uma coletânea diversificada de trabalhos que abrangem as mais diversas áreas da Ciência de Dados, desde inteligência artificial, aprendizado de máquina, análise de big data até as aplicações mais recentes e impactantes em diversos setores da sociedade.

Estamos convictos de que "Data Science em Revista" se tornará uma referência na comunidade acadêmica, um veículo para o fortalecimento da nossa instituição e um canal para a projeção do trabalho incansável de nossos pesquisadores. Este é apenas o início de uma jornada promissora e desafiadora, repleta de oportunidades para contribuirmos cada vez mais para o avanço da Ciência de Dados.

Agradecemos a todos os envolvidos neste projeto, aos autores que submeteram seus trabalhos, aos orientadores que se dedicaram a garantir a qualidade das publicações e a todos aqueles que, de alguma forma, contribuíram para a concretização desta revista.

Que "Data Science em Revista" seja a expressão do nosso compromisso com a excelência acadêmica, a inovação e o avanço do conhecimento em Ciência de Dados. Que ela ilumine os caminhos daqueles que buscam desbravar o fascinante mundo dos dados e que possa ser um farol para a comunidade acadêmica, inspirando futuras gerações de cientistas de dados.

A todos os leitores, desejamos uma estimulante jornada de descobertas e aprendizado ao explorar as páginas desta revista. Que ela seja uma fonte inesgotável de conhecimento e inspiração.

Boa leitura!

William Roberto Malvezzi
Coordenador da Graduação e Pós-graduação em Ciência de Dados e Machine Learning e Coordenador do Curso Superior de Tecnologia em Marketing do CEUB.

SUMÁRIO

REDES NEURAIS RECORRENTES: LSTM PARA PREDIÇÃO DE PREÇO DE AÇÕES

André Juan Costa Vieira
Ricardo José Menezes Maia

RESUMO

Na última década, estudos foram realizados para prever as tendências do mercado de ações com o intuito de buscar os preços dos ativos, porém esta é uma tarefa desafiadora. Visando solucionar esse desafio, várias técnicas e métodos foram propostos. Nesse estudo é apresentado um meio para tentar prever valores futuros das ações utilizando técnicas de inteligência artificial. Para tanto, dados referentes aos preços passados foram adquiridos e colocados em uma rede neural recorrente do tipo Long Short Term Memory (LSTM) para realizar o processo de predição. Em seguida, a rede foi avaliada pelas métricas pertinentes a regressões lineares.

Palavras-chave: Ações. Prever. Rede Neural. LSTM.

1 INTRODUÇÃO

A previsão do mercado de ações é uma questão importante e comentada no mundo financeiro. Existem diferentes hipóteses nesta área. Alguns pesquisadores enfatizam que os preços das ações refletem melhor todas informações disponíveis sobre o mercado. Outros tentaram prever o mercado por meio da análise fundamentalista de indicadores econômicos (ROE, ROIC, EBTIDA, PL/P, etc) ou por meio da análise técnica, que são análises gráficas utilizando estatística (SITE; BIRANT; ISIK, 2019). Mesmo com essas formas de análise, o mercado é considerado imprevisível.

Porém, um novo método de previsão foi implementado, o aprendizado de máquina.

Existem várias técnicas de aprendizado de máquina que são utilizadas com o intuito de prever oscilações de preços no mercado de ações. São elas: redes neurais LSTM, MLP, regressão logística, florestas randômicas, árvores de decisão e KNN, dentre outras (SANBOON; KEATRUANGKAMALA; JAIYEN, 2019).

Essas técnicas usam principalmente indicadores econômicos estruturados ou numéricos com resultados de análises técnicas das ações (SITE; BIRANT; ISIK, 2019).

Esses dados numéricos geralmente usam parâmetros de preços de ações anteriores para prever preços de ações futuras.

Com o aumento logarítmico do poder de processamento dos computadores, essas técnicas de aprendizado de máquina ficaram mais robustas, de modo que é possível usá-las em grandes bases de dados.

As previsões das oscilações de preços das ações desempenham um papel fundamental para os investidores tomarem uma decisão precisa sobre seus investimentos.

Para realizar essas previsões, analisam-se as tendências no decorrer de uma série temporal para criar um modelo de aprendizagem profunda (CHENG; HUANG; WU, 2018).

O objetivo deste estudo é o desenvolvimento de um modelo preditivo utilizando redes neurais que possa alcançar bom desempenho utilizando somente dados relativos aos preços anteriores.

Para isso, os dados referentes aos preços foram adquiridos utilizando o Yahoo Finance. Em seguida, estes dados foram processados por uma rede neural recorrente do tipo LSTM. Ao fim, os resultados foram avaliados por métricas referentes a regressões lineares.

2 REFERENCIAL TEÓRICO

Nesse tópico serão tratadas teorias sobre LSTM e meios de análise técnica de ações. Para isso, será utilizado bibliografias e artigos científicos pertinentes aos assuntos.

2.1 LSTM

Long Short Term Memory é um tipo de rede neural recorrente proposta como uma forma de solucionar o problema dos gradientes. Este problema é comum em qualquer rede neural, embora seja especialmente aparente nas redes neurais recorrentes pela forma como são estruturadas (HOCHREITER; SCHMIDHUBER, 1997).

Para calcular os gradientes de uma rede neural utiliza-se o backpropagation, que é um algoritmo que computa as derivadas da função por meio da regra da cadeia (GOODFELLOW; BENGIO; COURVILLE, p. 204).

Durante o backpropagation o sinal de gradiente pode ser multiplicado por um grande número de vezes pela matriz de peso associada às conexões entre os neurônios da camada oculta. Dessa forma a magnitude dos pesos na transição da matriz podem ter um forte impacto no processo de aprendizagem (PASCANU; MIKOLOV; BENGIO, 2013).

Para elucidar a questão, considere uma rede neural recorrente com apenas um neurônio por camada. Considere também conjunto de consecutivas camadas "T" com funções de ativação "tanh", Φ(·), aplicadas entre cada camada de neurônio. O peso compartilhado entre um par de nós ocultos é denotado por "w". Os valores ocultos nas várias camadas é "$h1$ ... h_t". A cópia dos pesos compartilhados "w" dentro das "T" camadas é denotado como "$wwtt$". Seja $\partial l/(\partial h\_t)$ a derivada da função de perda em relação a camada de ativação "h_t" (AGGARWAL, p.286). Sendo assim, tem-se:

$$\frac{\partial l}{\partial h_t} = \Phi'.(h_t + 1).(w_t + 1).\frac{\partial l}{\partial h_{t+1}}$$

Pelo fato dos pesos compartilhados dentre as diferentes camadas temporais serem os mesmos, o gradiente é multiplicado pela mesma quantidade $wwtt=ww$ para cada camada. Dependendo do valor da matriz que está sendo multiplicada,

existirá a possibilidade de desvanecer ou terá uma tendência a explodir (GOODFELLOW; BENGIO; COURVILLE, p. 306).

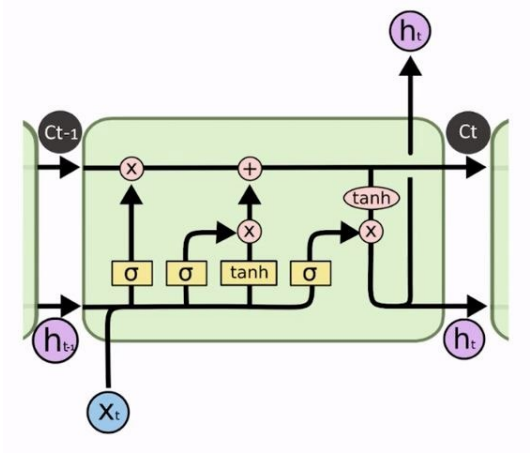
Com a matriz de pesos tendendo para baixo de um ($w<1$) o sinal gradiente fica tão pequeno que o aprendizado se torna muito lento para funcionar completamente. Nesse caso ocorre o desvanecimento das camadas.

Com a matriz de pesos tendendo para cima de um ($w>1$) o sinal gradiente se torna tão grande que ocorre divergência no aprendizado. Nesse caso ocorre a explosão do gradiente. Em suma, com $W<1$ o aprendizado nas primeiras camadas não ocorre, já que a taxa de atualização dos pesos não acontece em tempo hábil. Com $w>1$ os valores do gradiente são distorcidos de modo que a atualização dos pesos são errôneos (BENGIO; SIMARD; FRASCONI, 1994).

O LSTM é diferente das RNN tradicionais em decorrência da nova arquitetura implementada chamada de célula de memória. A célula é composta por quatro elementos principais, são eles: o “input gate”; o “cell state” que contém um neurônio com uma conexão auto redundante; o “forget gate”; e um “output gate” (GERS; SCHMIDHUBER; CUMMINS, 1994). A conexão auto recorrente tem peso igual a um, o que assegura que o estado da célula de memória se mantenha constante nos intervalos de tempo, barrando qualquer interferência externa. Os “gates” servem para modular as interações entre a própria célula de memória e seu ambiente. O “input gate” possibilita a entrada ou bloqueio do sinal que altera o estado da célula de memória. O “output gate” pode permitir ou prevenir que a célula de memória afete outros neurônios. Por fim, o “forget gate” modula a conexão auto recorrente da célula de memória permitindo a célula reter ou descartar o estado anterior (SIYUAN; GUANGZHONG; YIFAN, 2018).

A figura 1 ilustra a arquitetura da célula de memória da rede LSTM:

Figura 1- Estrutura LSTM



Fonte: JINGYI, QINGLI LIU, KANG CHEN, JIACHENG WANG.

A variável $c_t$representa o estado da célula e a informação da rede armazenada internamento no tempo "t", $x_t$representa a entrada de dados no tempo "t", h_té a informação indicando o próximo "output" no tempo "t".

Para determinar a informação contida no modelo da célula de memória o "forget gate" recebe a informação prévia do tempo de saída h_te a atual entrada de dados. O valor de saída do "forget gate" é zero ou um e é multiplicado pelo estado de célula. Se o produto for um, a célula retém a informação anterior e se for zero a célula irá descartar a informação anterior (DU; LIU; CHEN; WANG, 2019).

O cálculo do processo de descarte pelo "forget gate" pode ser visto pela a equação abaixo, onde f_té a saída do "forget gate", W representa os pesos e $b_f$são as matrizes de compensações:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f$$

A função do "input gate" pode ser dividida em duas partes. Uma parte irá descartar parcialmente dados prévios e outra parte irá preservar os novos dados. As

duas partes do resultado são adicionadas para completar o processo de atualização do estado da célula de memória. O cálculo para a realização desse processo é demonstrado pelas equações abaixo, onde $\tilde{C}_t$ é o vetor do novo estado da célula criado pela camada tangente hiperbólica da rede neural em preparação para a atualização do estado da célula:

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i$$

$$\tilde{C}_t = tanh(W_c . [h_{t-1}, x_t] + b_c$$

$$c_t = f_t . c_{t-1} + i_t . \tilde{C}_t$$

O "output gate" determina a saída dos dados, a camada interna sigmoidal irá determinar qual parte dos dados serão processados para a saída e então o estado da célula de memória é processado pela função tangente hiperbólica e multiplicado pela saída sigmóide para obter um resultado final (OJO et al., 2019).

Esse processo pode ser calculado pelas seguintes equações:

$$o_t = \sigma(W_0 . [h_{t-1}, x_t] + b_0$$

$$t_t = o_t . \tanh(c_t)$$

2.2 Indicadores de Análise Técnica

Nos subtópicos a seguir serão discutidos de forma breve a teoria sobre os indicadores de análise técnicas utilizadas para avaliação de tendências dos ativos.

Cumpre ressaltar que existem mais de 50 tipos de indicadores para análise técnica e muitos deles utilizam métodos de média móvel (EDWARDS; MAGEE;

BASSETTI, 2019). Por isso, foram escolhidos os indicadores mais utilizados no mercado financeiro.

2.2.1 Média Móvel Simples

A média móvel simples é um método estatístico de análise técnica que utiliza dados para formar um indicador de tendência sequencial (DROKE, p. 22). Esse indicador calcula a média de um determinado número em um determinado período de tempo e, por se aplicar na análise de oscilações de preços de ações, cria-se uma série temporal demonstrando as tendências dos preços em relação ao tempo. Todos os dados utilizados para a formação da série temporal têm peso igual a um (wi=1) (ZAKAMULIN, 2017, p.24). Segue a equação matemática para o cálculo da SMA (Simple Moving Average):

$$(SMA_t n) = \frac{1}{n}\sum_{i=0}^{n-1} P_t - 1$$

2.2.2 Média Móvel Exponencial

Uma desvantagem da média móvel simples é que todos os dados utilizados para a construção da série temporal têm peso igual um (wi=1). Essa questão pode ser tratada utilizando a média móvel exponencial, onde o peso de cada número utilizado para a construção da série temporal varia, de modo que pode-se computar um maior ou menor peso para os números mais recentes (ZAKAMULIN, 2017, p.30). Segue a equação matemática utilizada para o cálculo da EMA (Exponential Moving Average):

$$EMA_t(\gamma, n) = \frac{\sum_{i=0}^{n-1} \gamma^1 i P_{t-1}}{\sum_{i=0}^{n-1} \gamma^1}$$

2.2.3 Média Móvel Convergente Divergente

O MACD (Moving Average Convergence Divergence) é calculado usando a diferença entre duas médias móveis exponenciais. Tradicionalmente, uma média móvel exponencial de 26 períodos é subtraída de uma outra de 12 períodos, porém, os períodos são ajustáveis. Um MACD positivo indica que o preço médio durante os últimos 12 períodos excedeu o preço médio dos últimos 26 períodos (PATRICK; DAHLQUIST, p.432). Segue a equação matemática a realização do cálculo do MACD:

$$MCD_t(s,l) = EMA_t(S) - EMA_t(l)$$

2.2.4 Bollinger Bands

As bandas de Bollinger é um indicador técnico construído por meio do calculo do desvio padrão das médias móveis simples em 20 períodos. Dois desvios padrão são adicionados ao SMA para traçar uma banda superior. A banda inferior é construída subtraindo dois desvios padrão de o SMA (BOLLINGER, 2001, p.52).

3 ARQUITETURA E IMPLEMENTAÇÃO

Neste tópico serão tratados como a arquitetura foi projetada, como foi implementada, como os dados foram tratados com o intuito de obter os melhores resultados e quais equipamentos foram utilizados.

3.1 Preparação dos dados e pré-processamento

A implementação do código foi realizada na linguagem de programação Python na versão 3.8. A IDE (Integrated Development Environment) utilizada foi o Spyder na versão 4.2.5. Essas versões foram utilizadas em decorrência da plataforma Anaconda somente suportar as versões 10.2.89 do módulo "cuda toolkit" e 7.6.5 do módulo “cudnn”. Esses dois módulos são necessários para o treinamento da rede

neural LSTM com a utilização de uma interface gráfica. A interface gráfica utilizada foi a NVIDIA GTX 1070 Founders Edition.

Para a aplicação do modelo preditivo foram escolhidas 03 famosas empresas listadas na bolsa de valores americana, são elas: Alibaba (Ticker: BABA); Berkshire Hathaway (Ticker: BRK-A); The Walt Disney Company (Ticker: DIS).

Os conjuntos de dados foram obtidos pela a utilização do módulo Yahoo Finance (yfinance). Os dados são referentes a todos os dias úteis entre o período de 01/01/2015 a 01/03/2021 e transformados em um Data Frame.

O dado utilizado para o treinamento do modelo é o preço de fechamento do pregão no dia. A porcentagem utilizada para treinamento foi de 80% e a porcentagem utilizada para teste foi de 20%.

A entrada de dados na rede neural é de 3 dimensões, para isso, os dados foram formatados, já que o modelo irá receber somente o preço de fechamento.

A rede neural LSTM consiste de 4 camadas com 100, 150, 200 e 250 neurônios, respectivamente. Para evitar o Overfitting, 4 camadas do tipo DropOut foram utilizadas com a desativação de 50% dos neurônios de cada camada LSTM por época. Por fim, uma única camada foi adicionada com somente um neurônio. Essa rede foi treinada por 20 épocas. Os resultados do modelo preditivo serão comparados aos indicadores técnicos para uma análise, verificando se há uma congruência entre eles.

Para obter uma noção mais profunda da acurácia, as seguintes métricas foram utilizadas: Erro médio absoluto; Erro quadrático médio; Erro absoluto mediano; Pontuação de variância explicada; Pontuação R2.

A arquitetura foi escolhida com base em inúmeros artigos citados nesse trabalho, porém, como exposto na literatura (BROWNLEE, 2017, p.178), não existe uma arquitetura de redes neurais que possa ser utilizada para todos os problemas. Com base nesse argumento, a arquitetura foi desenvolvida mediante experimentação empírica.

Os valores de fechamento de preço previstos foram analisados com base nos métodos de análise técnica para aferir se os preços obtidos são suficientes para indicar ao investidor se há viés de compra ou venda de ações.

3.2 Porquê LSTM?

Foi realizado um levantamento bibliográfico para a confecção deste trabalho. O foco foi analisar trabalhos que utilizaram modelos de aprendizagem de máquina para prever futuros preços e/ou tendências de oscilações destes.

Em um primeiro estudo (SITE; BIRANT; ISIK, 2019) foram utilizadas técnicas de aprendizagem profunda como redes neurais recorrentes, LSTM, GRU, bem como métodos de aprendizagem de máquina, sendo eles a regressão linear, regressão ridge e Support Vector Machine. Aplicando todas essas técnicas, os autores chegaram à conclusão de que a rede neural LSTM teve o maior êxito em prever valores futuros.

Neste segundo estudo (SANBOON; KEATRUANGKAMALA; JAIYEN, 2019) foram utilizados métodos de aprendizagem de máquina como redes neurais LSTM, MLP, regressão logística, florestas randômicas, árvores de decisão e KNN.

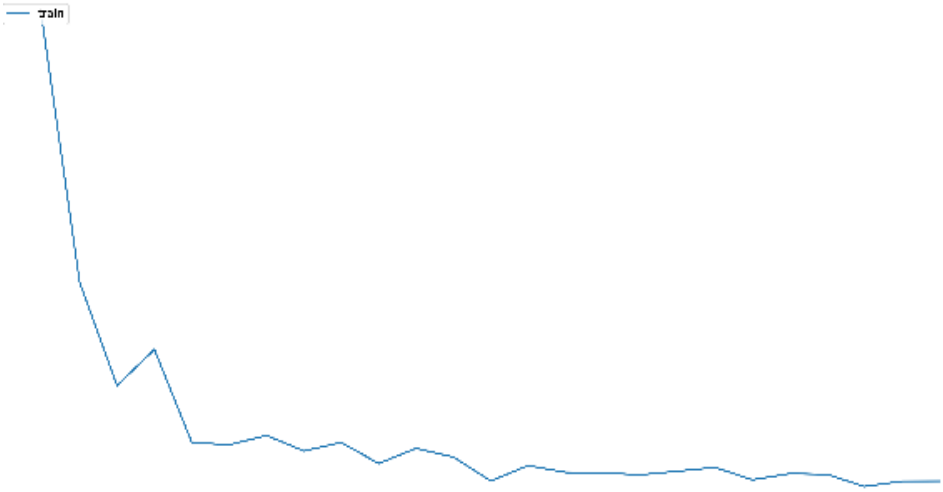
Após a aplicação de todos esses modelos, as redes neurais LSTM tiveram maior êxito em prever valores futuros.

Num terceiro estudo (DAMRONGSAKMETHEE; NEAGOE, 2020) foi construído um modelo preditivo utilizando LSTM e comparando-o com os resultados obtidos por outros 3 estudos que utilizaram inúmeros métodos de aprendizagem de máquina.

Mais uma vez as redes neurais LSTM tiveram maior êxito em prever valores futuros.

Poderia ser elencado aqui inúmeros estudos, artigos e textos acadêmicos que possam comprovar a maior eficiência das redes neurais LSTM, porém o motivo da escolha desse método já está claro.

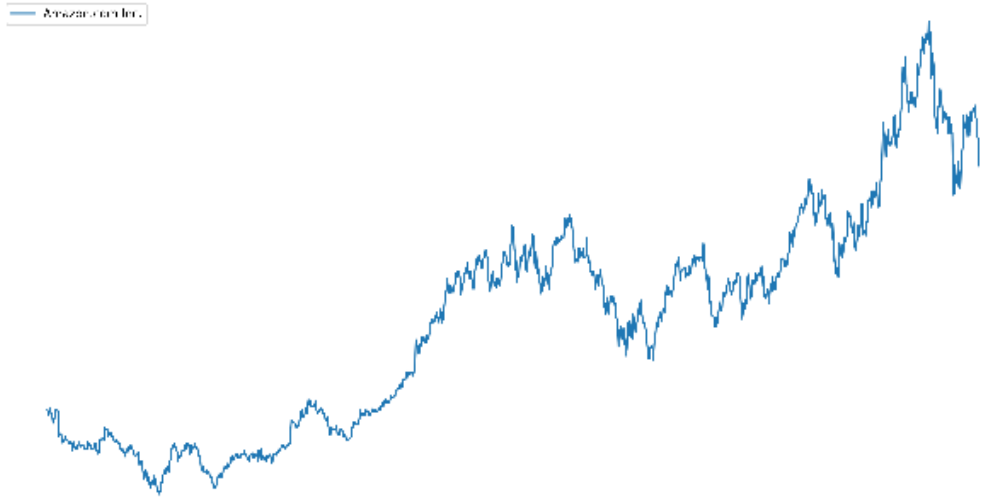
4 RESULTADOS E ANÁLISE



Neste tópico serão apresentados os resultados da aplicação do modelo preditivo proposto nas três empresas listadas anteriormente.

No gráfico 1, pode-se verificar que as ações da Alibaba tinham um preço de USD 103,02 em 01/01/15 e um preço de USD 241,69 em 01/03/21.

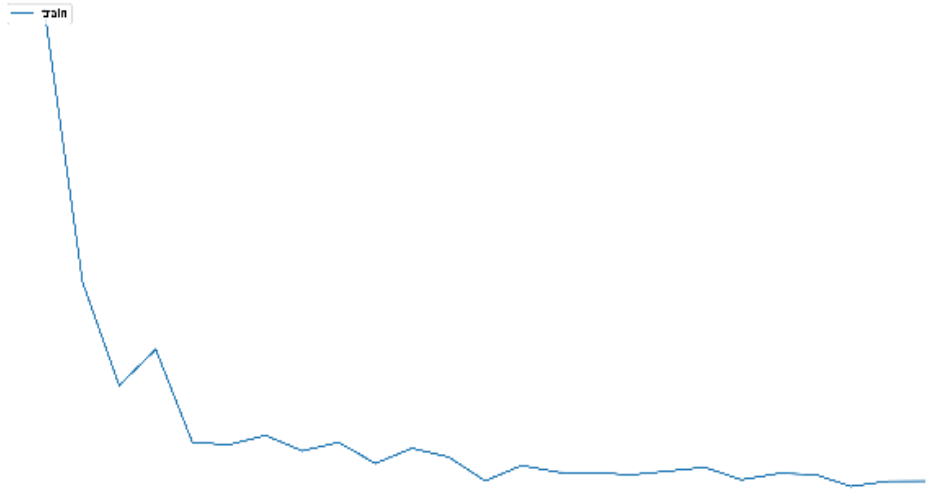
Gráfico 1 - Histórico de preços da Alibaba.



Fonte: Autor.

O gráfico 2 mostra o erro por época do treinamento da rede neural LSTM.

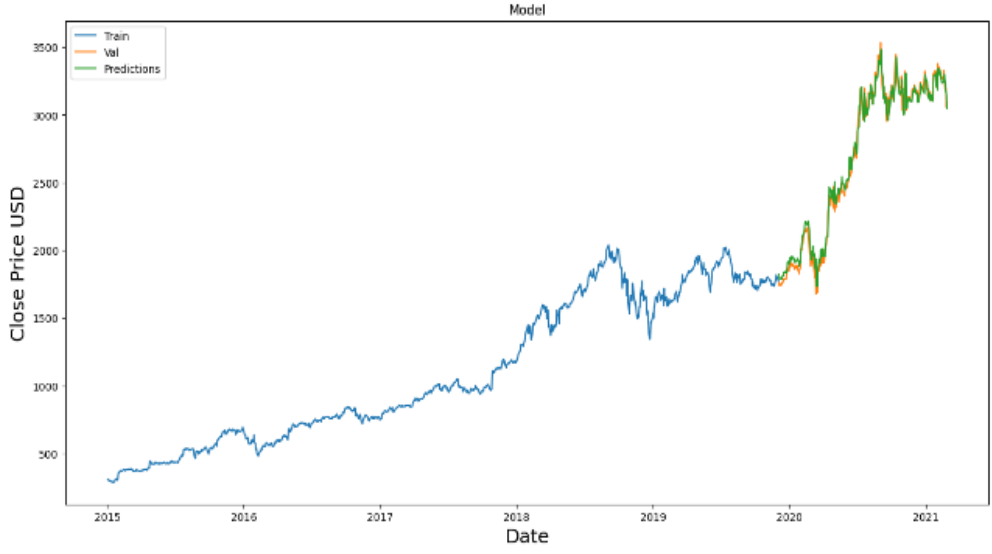
Gráfico 2 – Erro por época da Alibaba.



Fonte: Autor.

O gráfico 3 mostra o todo o histórico com o resultado do treinamento do modelo preditivo.

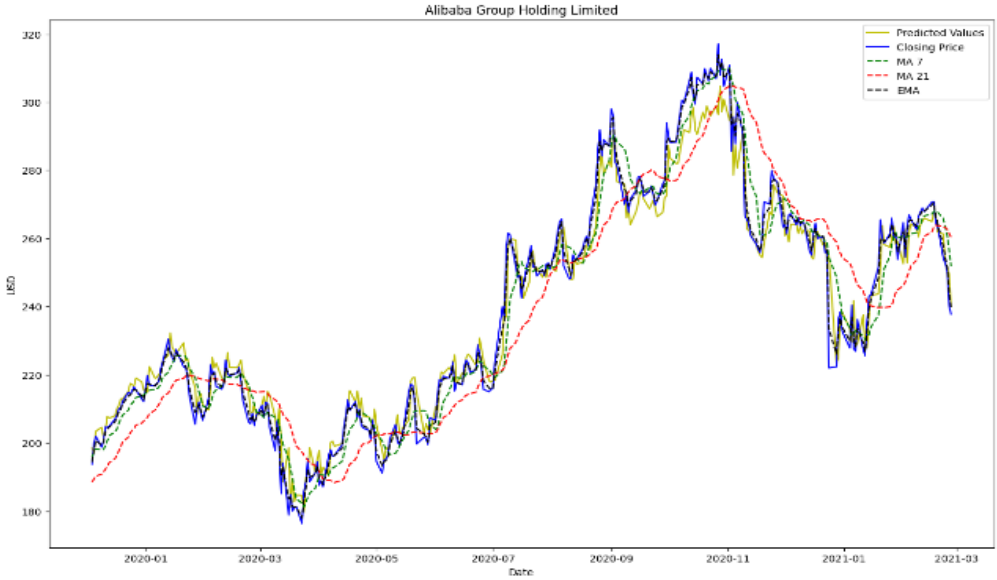
Gráfico 3 – Histórico com treinamento da Alibaba.



Fonte: Autor.

O Gráfico 4 contém somente o treinamento com o histórico real e os indicadores de média móvel de 7 e 21 dias, bem como a média móvel exponencial.

Gráfico 4 – Alibaba indicadores MA e EMA.

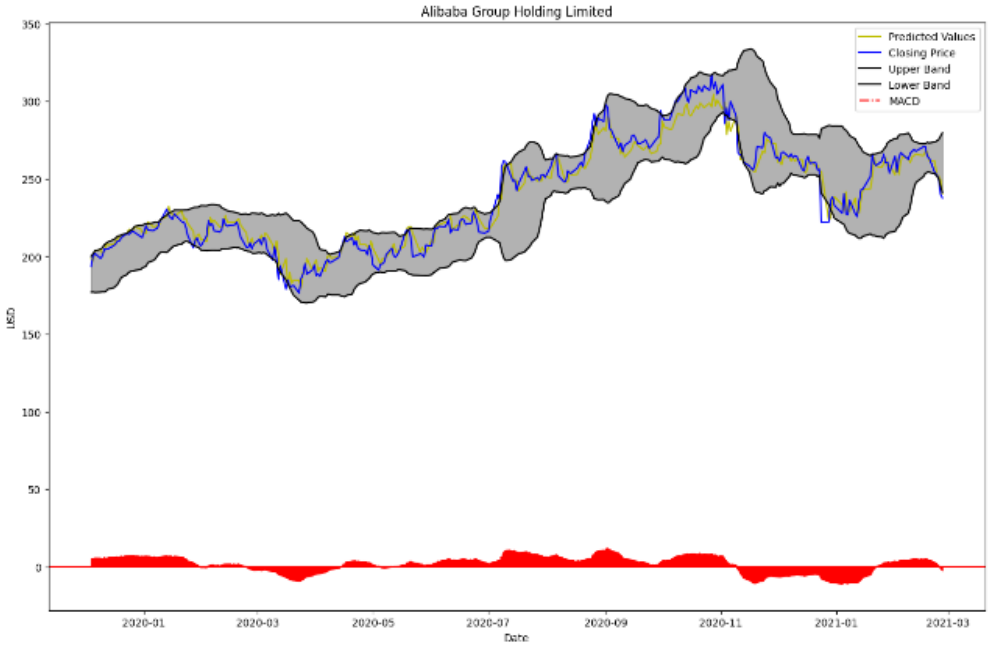


Fonte: Autor.

A linha amarela representa o modelo preditivo. A linha azul representa o histórico de preços das ações do Alibaba. A linha verde representa a média móvel de 7 dias. A linha vermelha representa a média móvel de 21 dias e a linha preta representa a média móvel exponencial.

A figura a seguir traz os indicadores de Bollinger e a média móvel convergente divergente. As linhas pretas representam os limites de Bollinger, a linha amarela representa o modelo preditivo e a linha azul representa o histórico de preços. O indicador do MACD encontra-se em vermelho na área inferior do gráfico.

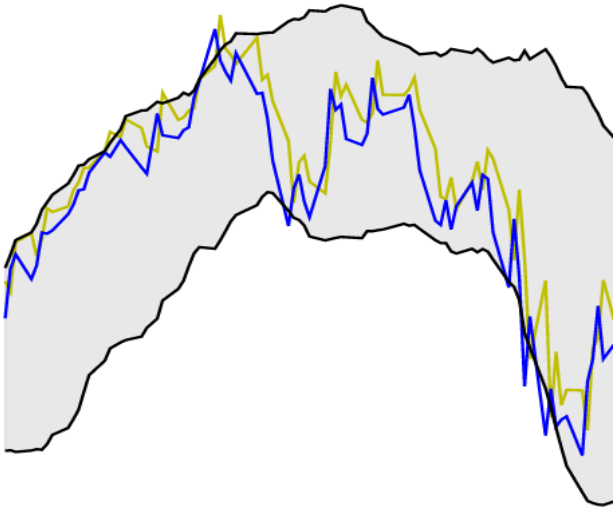
Gráfico 5 – Alibaba indicadores Bollinger e MACD.



Fonte: Autor.

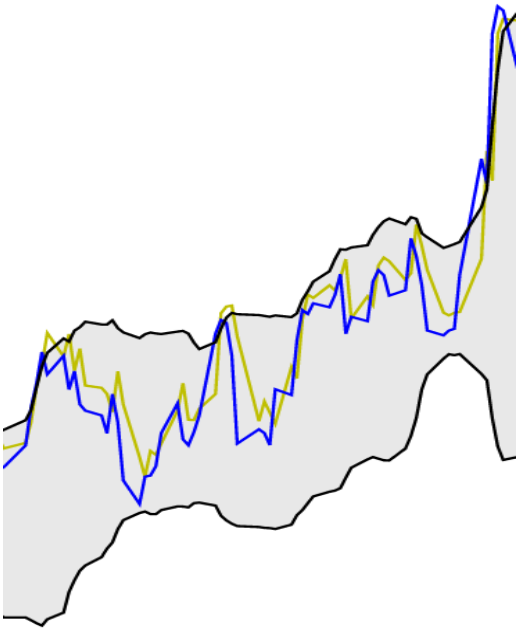
Os gráficos 6, 7 e 8 mostram momentos de maiores convergências e divergências do modelo preditivo em relação aos preços reais.

Gráfico 6 – Pontos de convergência e divergência.



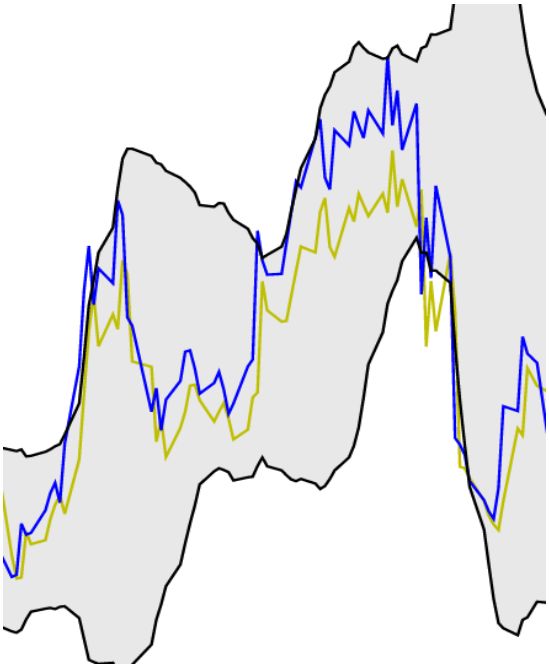
Fonte: Autor

Gráfico 7 – Pontos de convergência e divergência.



Fonte: Autor.

Gráfico 8 – Pontos de convergência e divergência.

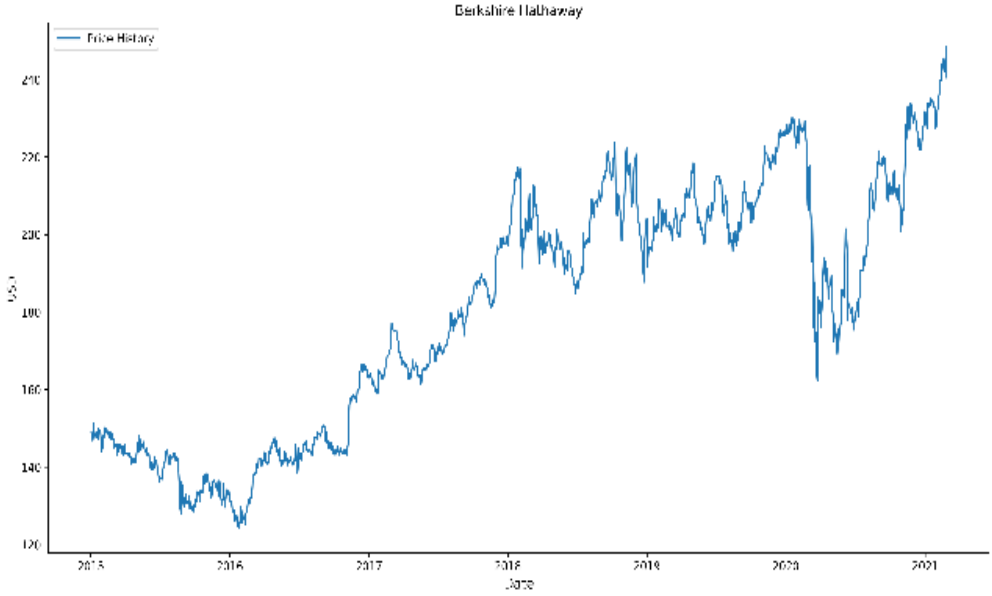


Fonte: Autor.

4.1.2 Berkshire Hathaway Company

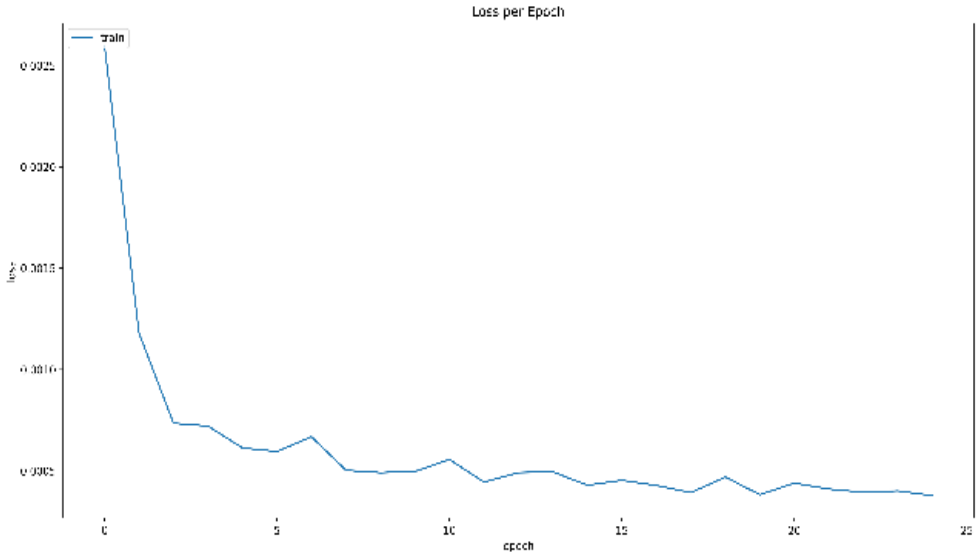
No gráfico 9, pode-se verificar que as ações da Berkshire Hathaway tinham um preço de USD 149,17 em 01/01/15 e um preço de USD 253,15 em 01/03/21.

Gráfico 9 – Histórico de preços da Berkshire.



Fonte: Autor.

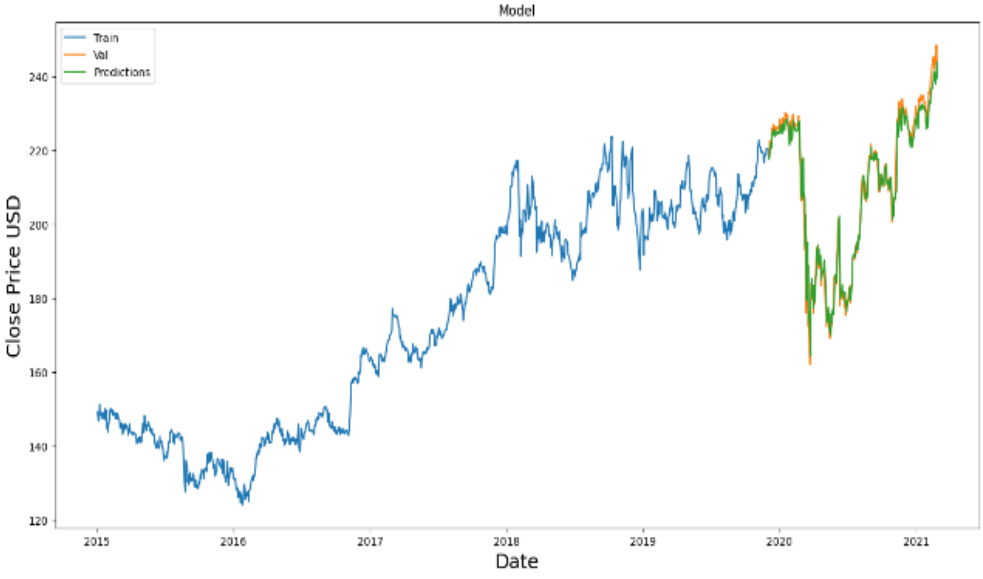
O gráfico 10 mostra o erro por época do treinamento da rede neural LSTM.



Fonte: Autor.

O gráfico 11 mostra o todo o histórico com o resultado do treinamento do modelo preditivo.

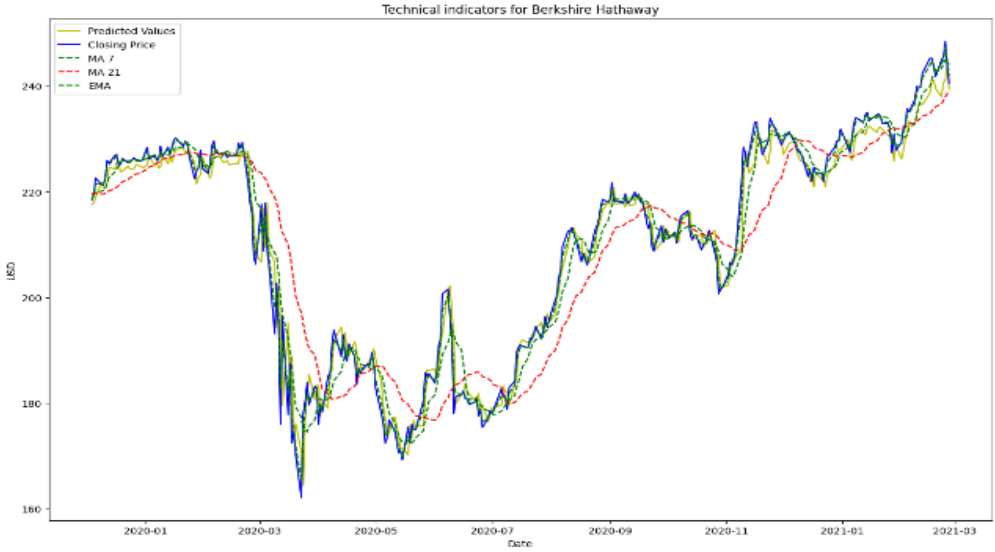
Gráfico 11 – Histórico com treinamento da Berkshire.



Fonte: Autor.

O gráfico 12 contém somente o treinamento com o histórico real e os indicadores de média móvel de 7 e 21 dias, bem como a média móvel exponencial.

Gráfico 12 – Berkshire indicadores MA e EMA.

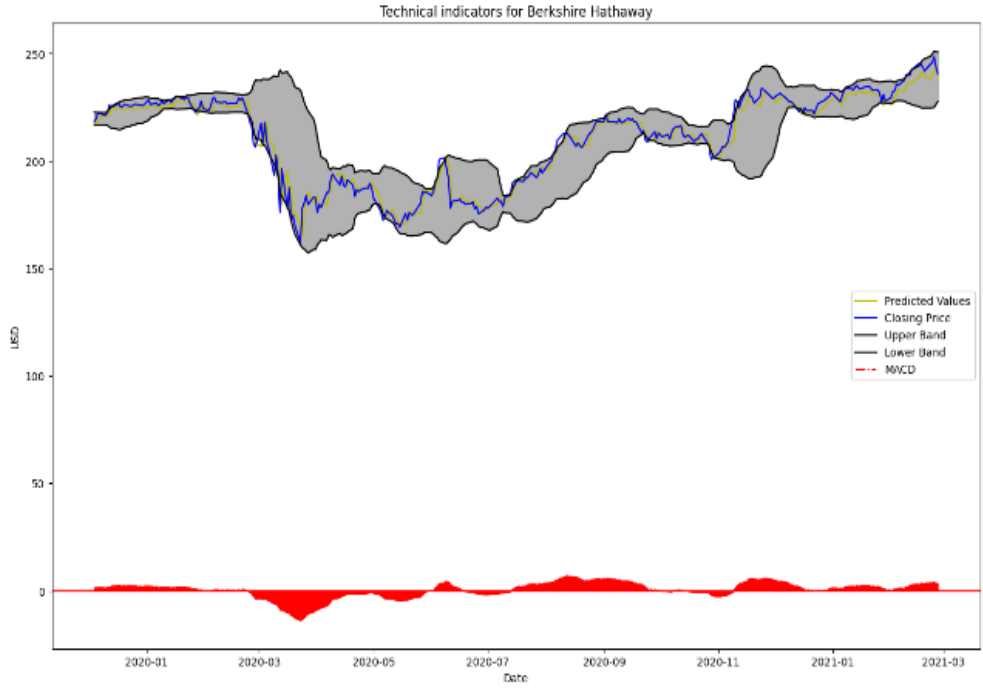


Fonte: Autor.

A linha amarela representa o modelo preditivo. A linha azul representa o histórico de preços das ações Berkshire. A linha verde representa a média móvel de 7 dias. A linha vermelha representa a média móvel de 21 dias e a linha preta representa a média móvel exponencial.

O gráfico 13 traz os indicadores de Bollinger e a média móvel convergente divergente. As linhas pretas representam os limites de Bollinger, a linha amarela representa o modelo preditivo e a linha azul representa o histórico de preços. O indicador do MACD encontra-se em vermelho na área inferior do gráfico que representa o histórico de preços. O indicador do MACD encontra-se em vermelho na área inferior do gráfico.

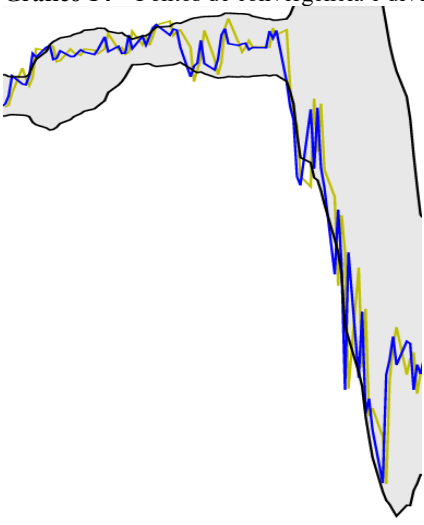
Gráfico 13 – Berkshire indicadores Bollinger e MACD.



Fonte: Autor.

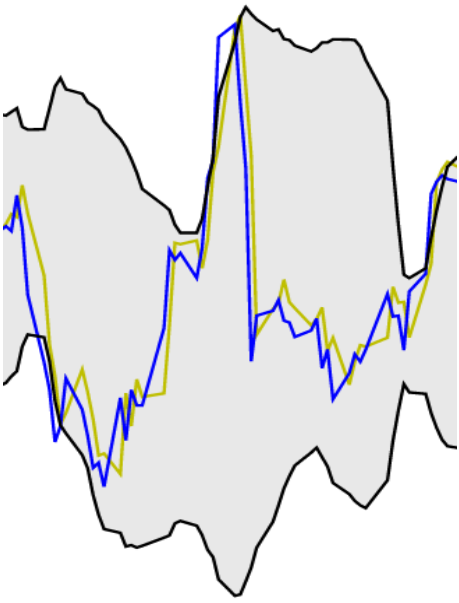
Os gráficos 14 e 15 mostram momentos de maior convergência e divergência do modelo preditivo em relação aos preços reais.

Gráfico 14 – Pontos de convergência e divergência.



Fonte: Autor.

Gráfico 15 – Pontos de convergência e divergência.

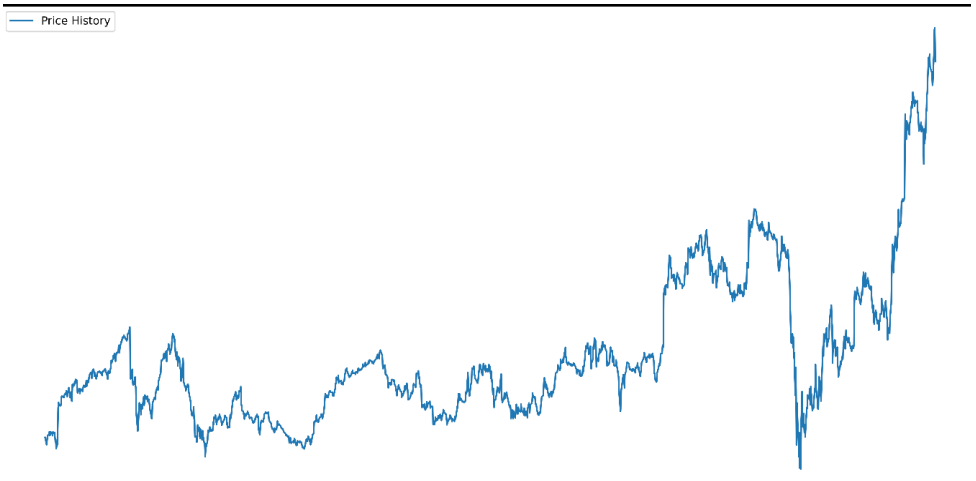


Fonte: Autor.

4.1.2 The Walt Disney

No gráfico 16, pode-se verificar que as ações da Walt Disney tinham um preço de USD 104,89 em 01/01/15 e um preço de USD 253,15 em 01/03/21.

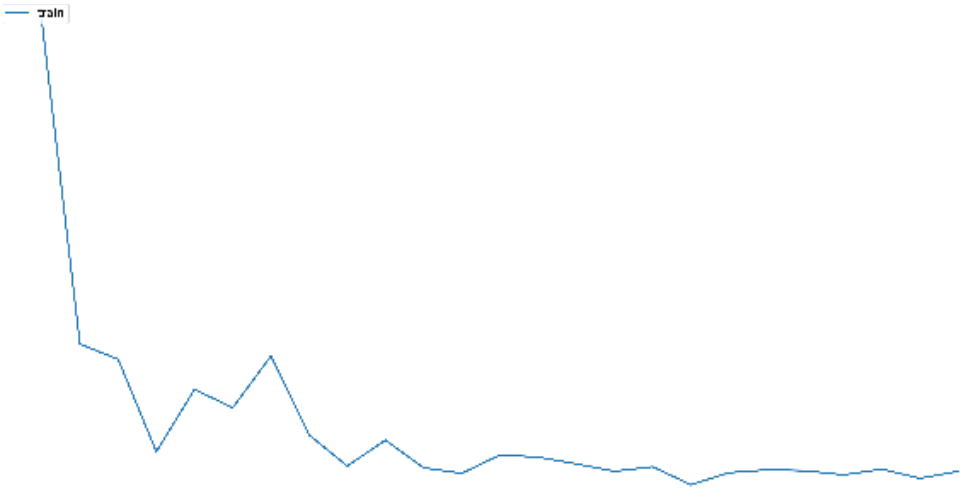
Gráfico 16 – Histórico de preços da Disney.



Fonte: Autor

O gráfico 17 mostra o erro por época do treinamento da rede neural LSTM.

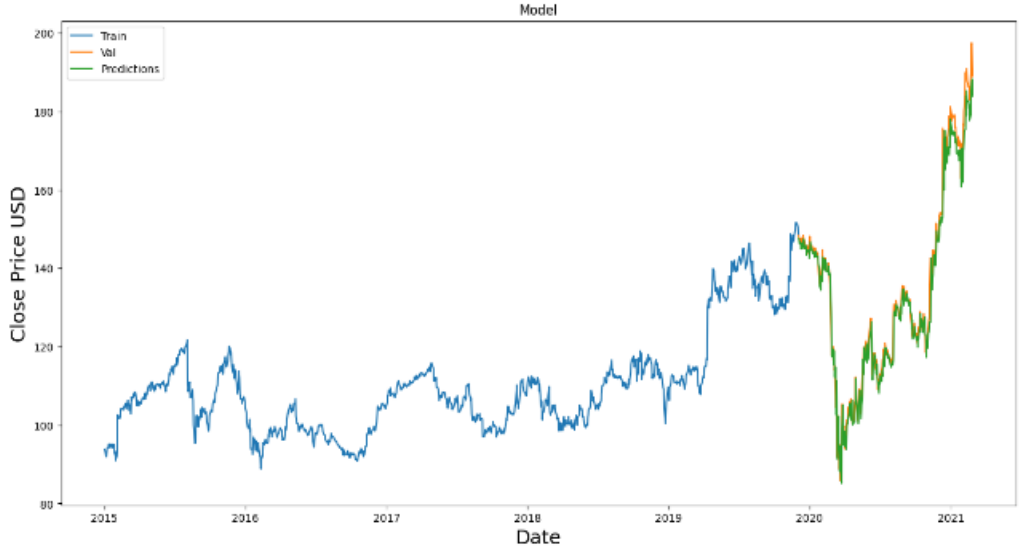
Gráfico 17 – Erro por época da Disney.



Fonte: Autor.

O gráfico 18 mostra o todo o histórico com o resultado do treinamento do modelo preditivo.

Gráfico 18 – Histórico com treinamento da Disney.



Fonte: Autor.

O gráfico 19 contém somente o treinamento com o histórico real e os indicadores de média móvel de 7 e 21 dias, bem como a média móvel exponencial.

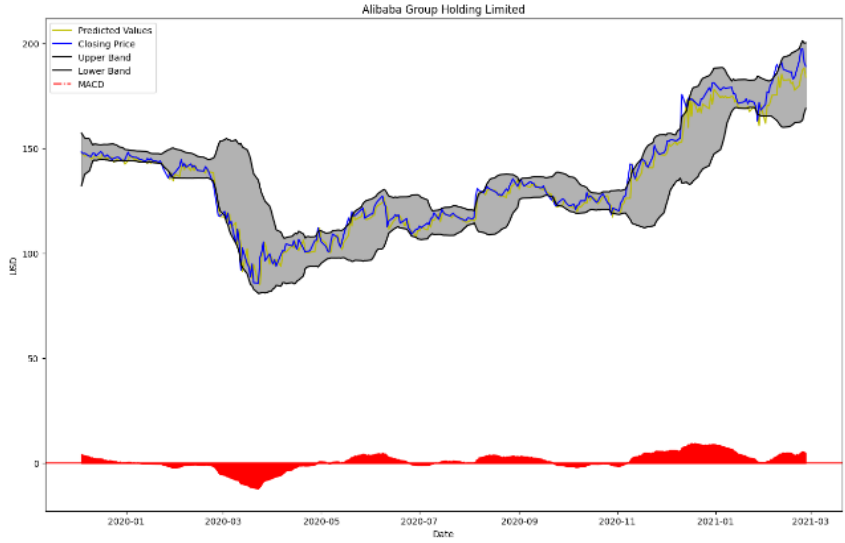
Gráfico 19 – Disney indicadores MA e EMA.

Fonte: Autor.

A linha amarela representa o modelo preditivo. A linha azul representa o histórico de preços das ações Berkshire. A linha verde representa a média móvel de 7 dias. A linha vermelha representa a média móvel de 21 dias e a linha preta representa a média móvel exponencial.

O gráfico 20 traz os indicadores de Bollinger e a média móvel convergente divergente. As linhas pretas representam os limites de Bollinger, a linha amarela representa o modelo preditivo e a linha azul representa o histórico de preços. O indicador do MACD encontra-se em vermelho na área inferior do gráfico.

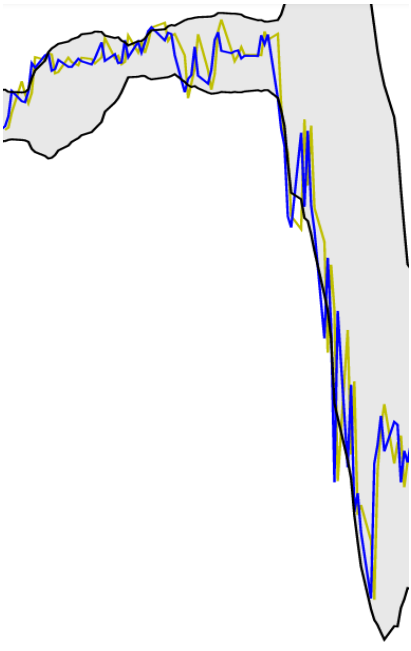
Gráfico 20 – Disney indicadores Bollinger e MACD.



Fonte: Autor.

O gráfico 21 mostra momentos de maior convergência e divergência do modelo preditivo em relação aos preços reais.

Gráfico 21 – Pontos de convergência e divergência.



Fonte: Autor.

4.2 Análise

A tabela a seguir contém as métricas de erro da rede neural LSTM relativas às 3 empresas em questão:

Tabela 1 – Métricas.

	Alibaba	Berkshire	Disney
MeanAE	5.16	2.78	2.92
MSE	48.19	16.21	17.98
MedianAE	3.92	2.01	1.98
EVS	0.96	0.96	0.98
R2 Score	0.96	0.96	0.97

A primeira métrica é o erro absoluto médio. Essa métrica mede a diferença entre o módulo dos resultados obtidos e os resultados reais e calcula a média. A segunda métrica é o erro quadrático médio. Essa métrica mede a diferença entre os resultados obtidos e os resultados reais, eleva cada diferença ao quadrado, e depois calcula a média. A terceira métrica é o erro mediano absoluto. Essa métrica mede a mediana de todos os erros. A quarta métrica é a pontuação da variância explicada. Essa métrica mede o quão bem o modelo pode explicar a variação no conjunto de dados. A quinta métrica é a pontuação R2. Essa métrica é relativa ao coeficiente de determinação, demonstrando o quão bem o modelo foi ajustado aos dados e as quão bem futuras amostras serão preditas (CIABURRO; JOSHI, 2019, p. 36).

A relação entre preço real e preço predito das empresas em relação ao erro absoluto médio foram os seguintes: Alibaba teve um erro de 5.16; Berkshire teve um erro de 2.78; Disney teve um erro de 2.92.

A relação entre preço real e preço predito das empresas em relação ao erro quadrático médio foram os seguintes: Alibaba teve um erro de 48.19; Berkshire teve um erro de 16.21; Disney teve um erro de 17.98.

A relação entre preço real e preço predito das empresas em relação ao erro mediano absoluto foram os seguintes: Alibaba teve um erro de 3.92; Berkshire teve um erro de 2.01; Disney teve um erro de 1.98.

A relação entre preço real e preço predito das empresas em relação a pontuação da variância explicada foram as seguintes: Alibaba teve um erro de 0.96; Berkshire teve um erro de 0.96; Disney teve um erro de 0.98.

A relação entre preço real e preço predito das empresas em relação a pontuação R2 foram as seguintes: Alibaba teve um erro de 0.96; Berkshire teve um erro de 0.96; Disney teve um erro de 0.97.

As métricas obtidas são consideravelmente boas, mas por si só não relatam todo o contexto.

Numa análise final, verificou-se por meio de uma comparação entre o modelo preditivo e os indicadores técnicos média móvel simples, média móvel exponencial e média móvel convergente e divergente, que a rede neural teve sucesso em acompanhar as tendências, indicando que as predições poderiam ser utilizadas como mais uma métrica de análise técnica.

Porém, houveram momentos em que o modelo preditivo destoou significativamente dos preços reais das ações, não relatando bem algumas oportunidades que poderiam ser aproveitadas pelo investidor. Isso ficou claro quando o indicador técnico de Bollinger foi aplicado.

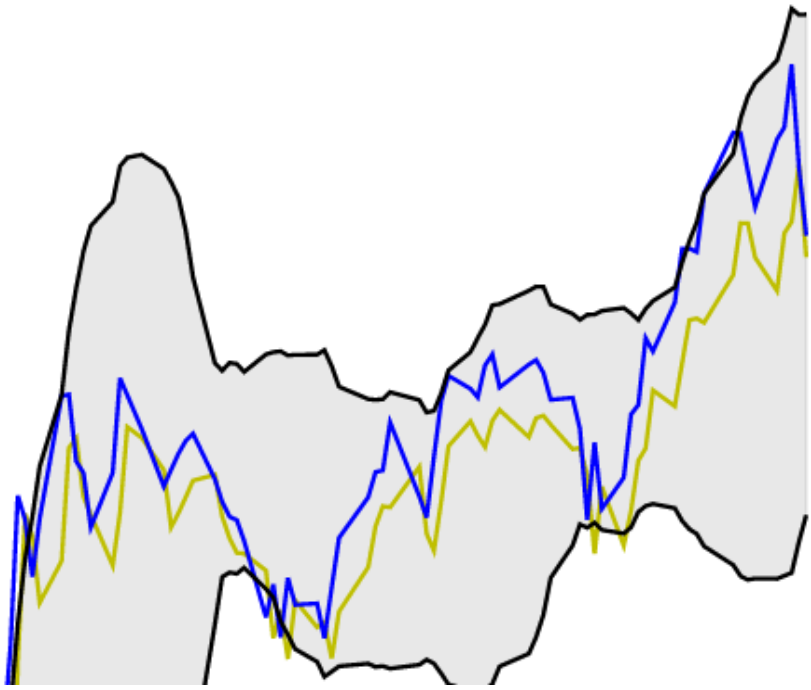
Em relação a empresa Alibaba, os gráficos 6 e 7 trazem momentos em que houveram boas predições, ao mesmo tempo em que houveram discrepâncias consideráveis. Percebe-se que em alguns momentos o modelo preditivo obteve êxito em acompanhar os preços reais quando romperam pontos de suporte e resistência, bem como momentos em que o rompimento desses pontos não foi apontado.

Em relação a empresa Berkshire Hathaway, os gráficos 14, 15 e 16 demonstram excelentes resultados. O modelo conseguiu acompanhar as tendências de modo satisfatório. Porém, ao analisar o gráfico 17, percebe-se que o erro entre os valores preditos e os valores reais é muito alto. Erros como estes demonstrados trazem grande insegurança ao investidor, isso porque se comparado os resultados

com os indicadores técnicos, fica nítido que o modelo está resultando em valores discrepantes.

Em relação a empresa The Walt Disney, os gráficos trouxeram resultados muito bons, onde o modelo seguiu todas as tendências dos preços. Porém, o gráfico 22 demonstrou um erro consideravelmente grande, acarretando na perda de credibilidade do modelo.

Gráfico 22 – Pontos de convergência e divergência.



Fonte: Autor.

5 CONCLUSÃO

Ao todo, o conjunto de dados contém 1549 dias úteis de pregão, sendo que 1239 dias foram utilizados para treinamento da rede neural e 310 dias foram utilizados para controle de acurácia.

A análise gráfica realizada proporcionou a visão de que o modelo começou a obter maiores erros depois de 200 dias de predição.

Na maioria dos casos, o modelo teve excelentes resultados em prever tendências, isso porque as métricas obtidas demonstraram a boa performance, mas os erros obtidos em outros momentos podem fazer com que o investidor tenha prejuízos maiores que os ganhos obtidos.

Resta claro que o modelo preditivo pode ser utilizado como um parâmetro para análise técnica de ações, se acompanhado por outros indicadores.

Conclui-se que a eficácia do modelo é relativa. Somente a utilização de preços passados não é suficiente para criar um modelo preditivo que possa ser utilizado como ferramenta única de análise técnica.

Em trabalhos futuros serão utilizados outros dados, além dos preços passados. Esses dados são referentes a números contábeis das empresas a serem analisadas. Dessa forma, espera-se que o modelo preditivo consiga ter melhores resultados.

REFERÊNCIAS

AGGARWAL, C. Neural Networks and Deep Learning. Springer, 2018.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Leaming Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks, vol. 5, no. 2, 1994.

BOLLINGER, J. Bollinger on Bollinger Bands. McGraw-Hill, 2001.

BROWNLEE, J. Long Short-Term Memory Networks With Python: Develop Sequence PredictionModels With Deep Learning. Machine Learning Mastery, 2017.

CHENG, L.; HUANG, Y.; Wu, M. Applied attention-based LSTM neural networks in stock predictions. IEEE, 2018.

CIABURRO, G.; JOSHI, P. Python Machine Learning Cookbook: Over 100 recipes to progress from smart data analytics to deep learning using real-world datasets. Second Editon, 2019.

DAMRONGSAKMETHEE, T.; NEAGOE, V. Stock Market Prediction Using a Deep Learning Approach. IEEE, 2020.

DROKE, C. Moving Averages Simplified. Marketplace Books, 2001.

DU, J.; LIU, G.; CHEN, K.; WANG, J. Forecasting stock prices in two ways based on LSTM neural networks. IEEE, 2019.

EDWARDS, R. D.; MAGEE, J.; BASSETI, W.H.C. Technical Analysis of Stock Trends. Routlegde, 2019.

GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. "Learning to Forget: Continual Prediction with LSTM." Neural Computation 2.10(1999): 2451-71.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning, 2016.

HOCHREITER, S., SCHMIDHUBER, J. Long Short-term Memory. Neural Computation 9(8):1735-1780, 1997.

JINGYI, Q. L.; CHEN, K.; WANG, J. Forecasting stock prices in two ways based on the LSTM neural network. 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2019).

LIU, S.; LIAO, G., DING, Y. Stock Transaction Prediction Modeling and Analisys based on LSTM. 2018, 13th IEEE Conference on Industrial Electronics and Applications (ICIEA).

OJO, S. O.; OWOLAWI, P. A.; MPHAHLELE, M.; ADISA, J. A. Stock Market Behaviour Prediction using Stacked LSTM Networks. IEEE, 2019.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. ICML Conference, 28, pp. 1310–1318, 2013.

PATRICK, C. K.; DAHLQUIST, J. Technical Analysus: The complete resource for financial market technician. Pearson Education, 2011.

SANBOON, T.; KEATRUANGKAMALA, K.; JAIYEN, S. A Deep Learning Model for Predicting Buy and Sell Recommendations in Stock Exchange of Thailand using Long Short-Term Memory. IEEE 4th International Conference on Computer and Communication Systems, 2019.

SITE, A.; BIRANT, D.; IŞIK, Z. Stock Market Forecasting Using Machine Learning Models. IEEE, 2019.

ZAKAMULIN, V. Market Timing with Moving Averages: The Anatomy and Performance of Training rules. Palgrave Macmillan, 2017.

ANÁLISE DE DADOS EM FORMATO TEXTUAL: APLICAÇÃO DE MÉTODOS DE APRENDIZADO DE MÁQUINA PARA CLASSIFICAÇÃO DE ACORDOS DE DESEMPENHO DE PESSOAS

Ângela Raquel de Araújo Danquimaia

William Roberto Malvezzi

RESUMO

A classificação é altamente significativa na mineração de dados, o que leva a uma série de estudos em aprendizado de máquina com pré-processamento e técnicas algorítmicas. Como usualmente em ambientes corporativos são firmados "Acordos de Desempenho" com os empregados em formato textual, é necessário avaliar se os Acordos firmados colaboram para a entrega das estratégias e, dado o volume de Acordos, é necessária a sistematização de regra para rejeição daqueles que não atendam a estratégia. Para essa sistematização foi utilizado técnicas de mineração de texto, com o saneamento da base, pré-processamento dos dados e aplicação dos algoritmos de classificação Regressão Logística, Naive Bayes e Máquina de Vetores de Suporte (SVM). Foram aplicadas técnicas de balanceamentos dos dados, SMOTE, Oversampling e Downsampling, dado o desequilíbrio observado no corpus com diferença significativa entre as classes. O algoritmo SVM com a aplicação da técnica SMOTE para balanceamento foi o mais eficaz na identificação das classes, no entanto, ressalta-se que também foi o algoritmo com maior tempo de processamento frente aos demais, com ganhos no resultado variando de 0,77% a 1,33%.

Palavras-chave: Mineração de texto. Naive Bayes. SVM. Regressão Logística. Dados desbalanceados.

1 INTRODUÇÃO

Segundo De Aquino Guimarães (1998), a avaliação de desempenho de pessoal deve ser entendida como um processo, que se inicia com o planejamento e

termina com a comparação entre o executado e o planejado. Podendo ser formal (registrado) ou informal (não registrado), sendo que a necessidade de formalização está relacionada diretamente ao tamanho da organização. Em empresas de grande porte a informalidade não permitiria à empresa o registro histórico e a uniformidade de procedimentos.

Ainda, De Aquino Guimarães (1998) afirma que avaliação de desempenho possui duas faces. Comumente vista como instrumento de gestão de desenvolvimento de recursos humanos e como instrumento de controle de gestão. No primeiro caso, de interesse do empregado, que a utilizaria para se beneficiar de treinamentos, prêmios, reconhecimentos. Já no segundo caso, onde a empresa pode registrar o que é feito e como é feito, mantendo um controle sobre o rendimento e eficiência da mão-de-obra. No entanto, o controle pode ser visto de forma negativa, com menor grau de liberdade sobre o trabalho a ser executado e sobre qual comportamento adotar para a execução.

A avaliação de desempenho para Lotta (2002) é um mecanismo que busca conhecer e medir o desempenho dos indivíduos na organização, estabelecendo uma comparação entre o desempenho esperado e o apresentado por esses indivíduos.

Brandão et al. (2008) afirma que, num contexto de globalização e de busca por maximização de resultados, ferramentas de gestão do desempenho capazes de integrar estratégia, aprendizagem, competências e indicadores quantitativos e qualitativos são muito desejadas, mas dificilmente encontradas.

Ainda, a função de aumentos salariais e promoção, por sua vez, diz respeito à capacidade que uma avaliação de desempenho tem de dar subsídios à política de promoção e salários, ou seja, a avaliação pode ser utilizada a fim de gerar informações para discriminação do aumento salarial e promoção, segundo Lotta (2002). Indicando assim, a importância na qualidade e na fidedignidade que as avaliações devem ter, uma vez que podem apresentar reflexos diretos na vida funcional dos empregados.

Lotta (2002), lembra ainda que a avaliação de desempenho serve para permitir que a organização estabeleça políticas de desenvolvimento de seus funcionários nos quesitos necessários ao melhor desempenho e utilização dos

talentos disponíveis. Em relação aos aspectos legais, na área pública, em geral, a avaliação de desempenho é exigida por lei, a fim de registro do desempenho dos funcionários, como no caso do Brasil, onde serve para cômputo de pontos para promoção salarial.

Percebe-se na literatura levantada que, em praticamente todos os casos, a avaliação de desempenho é feita por meio de formulários preenchidos, em formato texto, onde constam as expectativas de entregas e comportamentos e os resultados obtidos, como visto em Lotta (2002); Brandão et al. (2008) e De Aquino Guimarães (1998).

Assim, foi possível observar que diversas instituições possuem um ampla base de informações em formato textual, onde estão apostas os resultados esperados, como eles foram alcançados, o possível comportamento do avaliador e do avaliado, a evolução dos empregados avaliados entre uma série de possíveis achados pouco explorados, em especial em grandes instituições, em função do formato de registro em texto que até pouco tempo carecia de técnicas que pudessem simplificar e automatizar a análise desse conteúdo. Informações que, se analisadas, são capazes de, segundo Lotta (2002), fornecer subsídios para planejamento estratégico da empresa, bem como mecanismo de identificação das deficiências e aptidões dos funcionários, o que permitiria desenvolver programas de capacitação e treinamento que possam diminuir ou até mesmo suprir tais deficiências.

Santos et al. (2015) afirma que a popularidade da Internet nos últimos anos ocasionou um aumento na quantidade de informações disponíveis na web, trazendo como consequência, uma sobrecarga de informação que faz com que encontrar informação relevante geralmente envolva a análise de uma grande quantidade de dados textuais. Neste contexto, a maior parte dos dados disponíveis está armazenada em documentos na forma de textos escritos em linguagem natural.

Assim, Ribeiro (2019) comenta que as interações na Internet ocorrem majoritariamente de forma textual, tornando necessário utilizar métodos apropriados para a extração de informações deste meio. Para atingir tal objetivo, surgem técnicas que buscam converter a linguagem humana em uma representação formal tal que esta, seja facilmente manipulável por computadores.

A partir dessa necessidade surgida pelo volume de informações dispostas e coletadas pela internet, essas mesmas técnicas podem ser úteis para avaliação das bases, também textuais, dos “Acordos de Desempenho” e o resultado das avaliações realizadas.

Assim, de modo geral, trata-se de um problema de classificação, onde é possível rotular as informações e, a partir disso retirar os insights interessantes para evolução da instituição nos aspectos já levantados.

Uma instituição com cerca de 85 mil empregados realiza sistematicamente avaliação de desempenho de um público-alvo de cerca de 70 mil empregados, com critério mínimo de definição de 3 Acordos de entrega por empregado, gerando uma base que superou 250.000 Acordos individuais em formato textual no ano de 2019, que ao final do ciclo são avaliados. A instituição já realiza a avaliação neste modelo há 3 anos.

A equipe responsável pela implementação, manutenção e evolução da avaliação de desempenho da instituição, observou muitos casos de “Acordos de Desempenho” sem relação com as entregas esperadas das unidades, o que torna a ferramenta descalibrada em termos de fidedignidade, uma vez que os resultados do desempenho são utilizados para consequências, como promoção, distribuição de bônus, premiações, podendo não contribuir para o atingimento dos marcos estratégicos definidos para a empresa, levando ao mal aproveitamento da força de trabalho.

Assim, faz-se necessário avaliar se os Acordos firmados colaboram para a entrega das estratégias da empresa e, dado o volume de acordos firmados, também se faz necessária a sistematização de regramento para rejeição de Acordos que não atendam a estratégia e são costumeiramente registrados.

Os objetivos do presente trabalho são: Avaliar os “Acordos de Desempenho” firmados entre empregados e a gestão de uma Instituição a fim de identificar a aderência à estratégia da empresa destes Acordos por meio de técnica de aprendizagem de máquina, ainda, especificamente, identificar os algoritmos de classificação para aprendizado de máquina mais utilizados para o domínio textual; executar o saneamento e pré-processamento da base identificando os principais

problemas na base para uma estruturação que permita a análise dos dados em ferramentas comumente utilizados para aplicação de técnicas de aprendizado de máquina; aplicar os algoritmos mais utilizados identificando o de melhor desempenho; analisar, por meio de medidas estatísticas, os resultados obtidos ao aplicar os modelos desenvolvidos para propor um classificador baseado em aprendizado de máquina que permita classificar quanto a aderência dos "Acordos de Desempenho".

Para o alcance dos objetivos, foi utilizada base de dados com as seguintes características, base de ciclo de avaliação de desempenho de pessoas de 2019 cuja informação é capturada em ferramenta própria da empresa, gerando relatórios em formato não estruturado, com aposição de título e numeração de página a cada conjunto de linhas. Ainda, pelo campo da ferramenta onde são registrados os acordos permitir qualquer tipo de caractere especial, incluindo "enter" e "tab", as ferramentas usualmente utilizadas para análise de dados não permitem a avaliação direta sem antes o saneamento e tratamento da base que viabilizou o início da análise dos acordos.

Ainda, na base de dados, houve a segregação dos dados de Acordos considerando a vinculação de unidades, sendo disponibilizado para o estudo o grupo de unidades onde houve análise prévia da equipe responsável pelo processo gerando uma classificação dos Acordos quanto a aderência com a estratégia da empresa, resultando numa base com 69.536 Acordos relacionados à área de suporte da empresa para o ano de 2019.

A partir do saneamento inicial da base de dados, foi realizada a etapa de pré-processamento dos dados textuais, o que permitiu a análise e implementação de 3 classificadores em mineração de texto, Support Vector Machine - SVM (Máquina de Vetores de Suporte), Naive Bayes e Regressão Logística para construção e avaliação dos modelos supervisionados por meio de resultados estatísticos das aplicações.

O presente trabalho foi então estruturado nas seguintes seções: na seção dois apresentam-se trabalhos relacionados ao tema de classificação de textos, na seção três são detalhadas características da variável em análise, "Acordos de Desempenho", e os principais tratamentos dados à base no pré-processamento; na

seção quatro apresenta-se os modelos de classificação utilizados no experimento e o resultado das análises sobre os resultados encontrados. Na última seção a conclusão e futuros trabalhos.

2 TRABALHOS RELACIONADOS

Segundo Aranha (2006), a mineração de textos é uma área multidisciplinar abarcando conhecimentos de computação, estatística e linguística. Mineração de textos tem como objetivo extrair regularidades, padrões ou tendências de grandes volumes de textos em linguagem natural. Tem sua inspiração em data mining ou mineração de dados, que busca identificar padrões em banco de dados estruturados, enquanto a mineração de textos procura extrair conhecimentos de base de dados não estruturadas ou semiestruturadas.

No contexto da mineração de texto, a análise de sentimento, ou mineração de opinião, é a atividade de identificar as opiniões sobre alguma entidade específica e, conforme Feldman (2013) já contava com mais de 7.000 artigos escritos sobre o tema. Ainda, segundo o mesmo autor, enorme explosão de divulgação de "sentimentos" nas mídias sociais, incluindo Twitter, Facebook, quadros de mensagens, blogs e fóruns de usuários são uma mina de ouro para empresas e indivíduos que querem monitorar sua reputação e obter feedback oportuno sobre seus produtos e ações. A análise de sentimentos oferece a essas organizações a capacidade de monitorar os diferentes sites de mídia social em tempo real e agir de acordo.

Geralmente, trata-se de uma análise de classificação onde, os textos em avaliação apresentam um nível de interpretação para o pesquisador positivo ou negativo. Assim como é observado no objeto dessa pesquisa, onde uma instituição, que possui processo estruturado de avaliação de desempenho dos seus empregados, tem o interesse em identificar se os “Acordos de Desempenho” firmados com seus colaboradores são “positivos” ou “negativos”, ou seja, “estão alinhados com a estratégia da empresa” ou “não estão alinhados com a estratégia da empresa”.

Assim, para pesquisa de trabalhos relacionados, observa-se o ineditismo na aplicação para o tema, vez que não se identificou estudos que aplicassem técnicas de

mineração de texto em bases de avaliação de desempenho de pessoas. No entanto, as características da aplicação assemelham-se àquelas realizadas para análise de sentimento, com uma gama significativa de trabalhos realizados.

Dentre os estudos identificados, Nascimento (2019) utiliza textos do Twitter para construção e avaliação dos modelos supervisionados, realizando uma análise comparativa dos algoritmos de aprendizagem de máquina: SVM, Naive-Bayes e Regressão Logística, combinados com a técnica de processamento de linguagem natural stemming para classificação automática de discursos de ódio em textos do Twitter.

Yuan (2014) realizou estudo com foco na discriminação de duas categorias emocionais: positiva e negativa. Em primeiro lugar, realizou o pré-processamento retirando os elementos ruidosos do microblog, depois extraiu os recursos do microblog e, por fim, classificou o microblog usando a técnica de aprendizado de máquina Support Vector Machine (SVM), concluindo pela eficácia do método.

Zhang (2016), apresenta a problemática de análise de sentimento em texto em chinês, com poucas pesquisas realizadas e utilizam do verbo, adjetivos e advérbios como recursos do texto, uso da técnica TF-IDF para cálculo de pesos das palavras e adotam, por fim, SVM e o Extreme Learning Machine (ELM) para analisar a tendência emocional do texto.

Ainda, em problemas de classificação, não é incomum que uma das classes de classificação tenha na população recorrência bem superior em termos de quantidade de ocorrências do que a outra classe minoritária, impactando no desempenho desses modelos de predição. Entre os estudos identificados, destaca-se aqui o trabalho de melhoria de classificação da análise de sentimento em dados do Youtube usando algoritmo para balanceamento de classes, o SMOTE realizado por Sarakit (2015).

3 MINERAÇÃO DE TEXTOS

A classificação de texto é o processo de classificação de documentos de texto em um conjunto predefinido de classes. É uma abordagem de aprendizagem supervisionada em que um conjunto de documentos de treinamento rotulados com

classes de 1 a n classes são usados para construir um modelo de classificação e com o intuito de prever a categoria da classe de um novo documento recebido com base no modelo de treinamento.

Vijayan (2017) esclarece que os tipos de classificação de texto incluem classificação de rótulo único e rótulo múltiplo. Quando um documento é atribuído a apenas uma classe, ele é chamado de rótulo único e quando mais de uma classe é atribuída para um documento, ele se torna uma classificação de rótulo múltiplo. A classificação binária que prevê se um documento pertence ou não a uma determinada classe é o melhor exemplo de uma classificação de rótulo único. Nesta pesquisa é tratada a categorização rígida do texto, o que significa que cada documento deve ser categorizado em uma das classes específicas.

A análise de categorização de textos possui estágios, que devem ser aplicadas conforme a necessidade e características da base de dados em estudo, como pré-processamento, indexação e redução de dimensionalidade, classificação e avaliação de desempenho da modelagem.

3.1 Características da variável de interesse

A base de dados utilizada para o estudo considerou dados de Empresa referente aos "Acordos de Desempenho" firmados com os empregados no ciclo de avaliação de desempenho de frequência anual ocorrida no ano de 2019. Os empregados devem ter no mínimo 3 "Acordos de Desempenho" registrados em ferramenta própria da Empresa para posterior análise, avaliação e pontuação quanto a efetiva entrega. No fim do processo, a fim de gerar a nota e classificação em 9 quadrantes distintos, são consideradas outras variáveis de análise, como o resultado da unidade da vinculação, efetividade em execução de capacitações e o desenvolvimento do empregado nos quesitos de competências comportamentais.

No que tange aos "Acordos de Desempenho", objeto da análise desse estudo, são firmados entre empregado e gestor imediato, registrados em forma textual de livre preenchimento e devem ser construídos considerando a metodologia SMART, desenvolvida por Peter Drucker, conhecido como "pai da Administração" e

publicada no ano de 1954 em seu livro “The Practice of Management” como relata Cardoso (2018).

O termo SMART trata-se de um acrônimo de 5 palavras da língua inglesa:

S – specific (específica): o acordo precisa ser específico e facilmente entendida por todos.

M – measurable (mensurável): se deve determinar o indicador pelo qual a entrega será medida.

A – attainable (atingível): a meta definida deve ser atingível. Em termos de quem executa, deve possuir condições e conhecimentos mínimos e do lado da empresa deve ter algum nível de aceitação do produto.

R – relevant (relevante): o acordo deve fazer sentido no contexto da empresa e colaborar com os objetivos estratégicos.

T – time based (temporizável): é necessário estabelecer um prazo para execução do acordo e se possível, marcos intermediários para acompanhamento e possíveis correções de rumo.

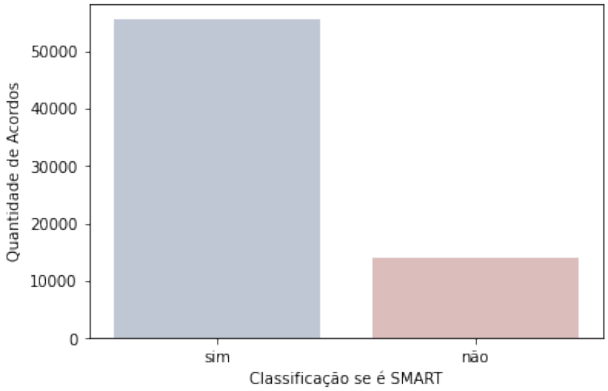
Considerando este contexto, os “Acordos de Desempenho” foram avaliados pela equipe responsável pela implementação e acompanhamento dos ciclos de desempenho da Empresa e classificados como sendo ou não sendo SMART.

3.2 Pré-processamento

A base de dados contou com 69.536 “Acordos de Desempenho”, de 14 unidades distintas, não sendo possível identificar a quantidade exata de empregados, uma vez que a base não apresenta dados pessoais, em atendimento ao regramento de segurança da informação da empresa.

Quanto à avaliação realizada para classificação dos Acordos, identifica-se desbalanceamento das classes, com classe majoritária entre os Acordos classificados como SMART e, como classe minoritária, os Acordos classificados como não sendo SMART como apresentado no Gráfico 1.

Gráfico 1 – Quantidade de Acordos conforme classificação se é SMART



Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Os Acordos classificados como SMART representam 79,7% da base, sendo o restante, 20,3% classificados como não sendo SMART. Em outros termos, para cada 5 Acordos firmados, 1 não possuía alguma das características de SMART.

Sarakit (2015) explana sobre o tema, esclarecendo que, em dados reais, existem muitas situações em que o número de instâncias em uma classe é muito menor do que o número de instâncias nas outras classes. Designando como problema de conjunto de dados desbalanceados. Com isso, a desempenho da classificação geralmente é baixo em várias técnicas de mineração de dados, incluindo reconhecimento de padrões, gerenciamento de dados e categorização de texto. Ocorre que, a maioria dos métodos convencionais atribui a classe majoritária aos dados e ignora a classe minoritária devido à distorção dos dados. Podendo resultar em modelos que apresentam alta acurácia, no entanto, como por exemplo, no problema exposto nesse trabalho, poderia hipoteticamente resultar numa acurácia alta, de 80% tendo errado praticamente todos da classe minoritária.

Uma das soluções mais populares para este problema é a Amostragem que será utilizada com aplicação de técnicas presentes no pacote IMBLEARN do Scikit Learn.

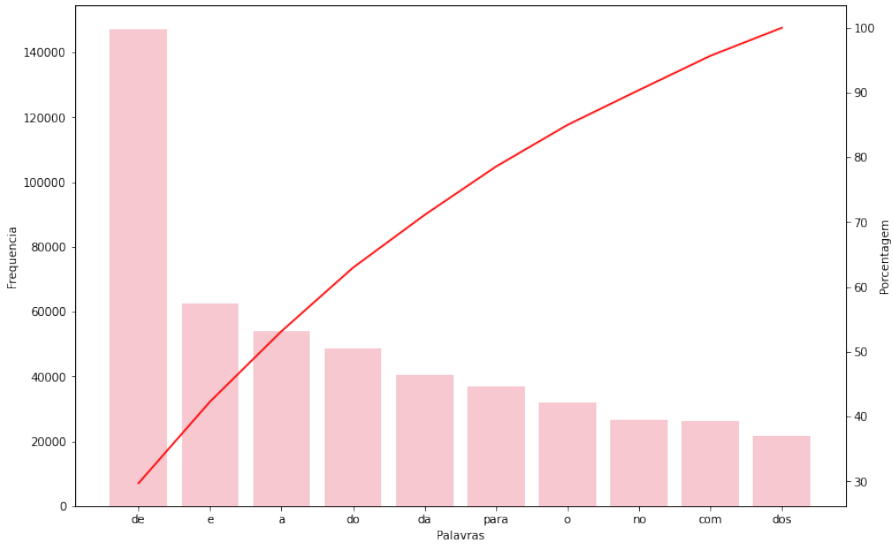
3.3 Pré-processamento

Para execução do pré-processamento foi utilizada a linguagem de programação Python e aplicados os recursos das bibliotecas SKLEARN; SEABORN; NLTK; UNIDECODE e IMBLEARN.

A base de dados, comumente designada “corpus” em mineração de texto, Como descrito anteriormente o conjunto de textos (corpus) utilizado neste trabalho é composto por mais de 69 mil Acordos totalizando 109.732 termos únicos (palavras, pontuação e símbolo). Dessa maneira antes de desenvolver qualquer modelo de classificação, Oliveira (2020) afirma que é necessário caracterizar o corpus, padronizar os textos e extrair o máximo de sentido das palavras utilizadas.

O Gráfico 2 apresenta as 10 palavras com maior número de ocorrências no corpus, onde é possível observar que as mais frequentes não contribuem para definição das características SMART. Ainda, podem ocorrer palavras cuja grafia correta seria com acentuação, mas possa ter sido registrada sem acento ou com acento inserido correta ou erroneamente e uso de letra maiúscula e minúscula para a mesma palavra, sendo consideradas como palavras diferentes pelas técnicas de mineração, além das pontuações. Assim, para o pré-processamento dos dados, inicialmente utilizou-se a abordagem de bag of words, transformando os textos dos Acordos em um formato estruturado por meio de vetorização das palavras e atribuição de um valor numérico único para cada palavra e, na sequência a abordagem de tokenize que, basicamente, consiste na separação das palavras pelos espaços em brancos e, para as correções necessárias. o Gráfico 2 apresenta as 10 palavras mais frequentes que tratam de artigos e preposições que pouco contribuem para a análise.

Gráfico 2 – Dez palavras mais frequentes no corpus pós bag of words e tokenização

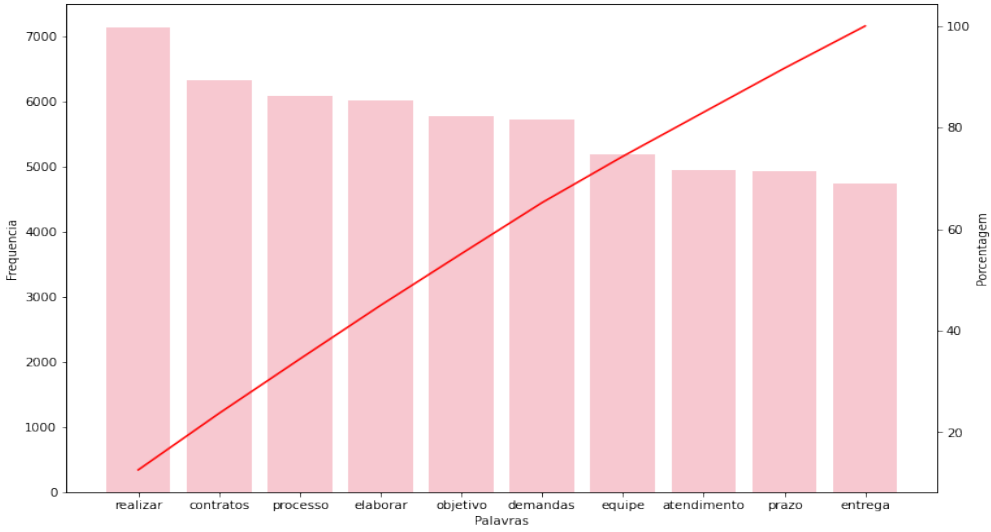


Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Considerando os termos que não contribuem na obtenção de sentido dos Acordos, procedeu-se com a remoção das chamadas stop words. A biblioteca NLTK já possui um corpus de stop words da língua portuguesa que foi aplicada ao corpus em estudo somado à exclusão de números e do nome da empresa, vez que números, como o ano de 2019, ano do ciclo apresentou quantitativo significativo entre as palavras.

Ainda, foram excluídas do corpus toda a pontuação, utilizando a biblioteca NLTK, todos os acentos por meio da biblioteca UNIDECODE e transformação de todas as palavras para letra minúscula a fim de capturar todos os stop words definidos. O Gráfico 3 apresenta as 10 palavras mais frequentes após a aplicação das técnicas, onde se observa ganho de informação para prosseguimento das análises.

Gráfico 3 – Dez palavras mais frequentes no corpus após retirar de pontuação, acentuação e transformação em letra minúscula



Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Observando por meio de gráfico de nuvem de palavras, segregando pelas classes de Acordos em estudo, já é possível observar diferentes termos se destacando como mais frequentes entre os grupos, como apresentado no Gráfico 4 e no Gráfico 5.

Gráfico 4 – Nuvem de palavras da classe de Acordos classificados como SMART

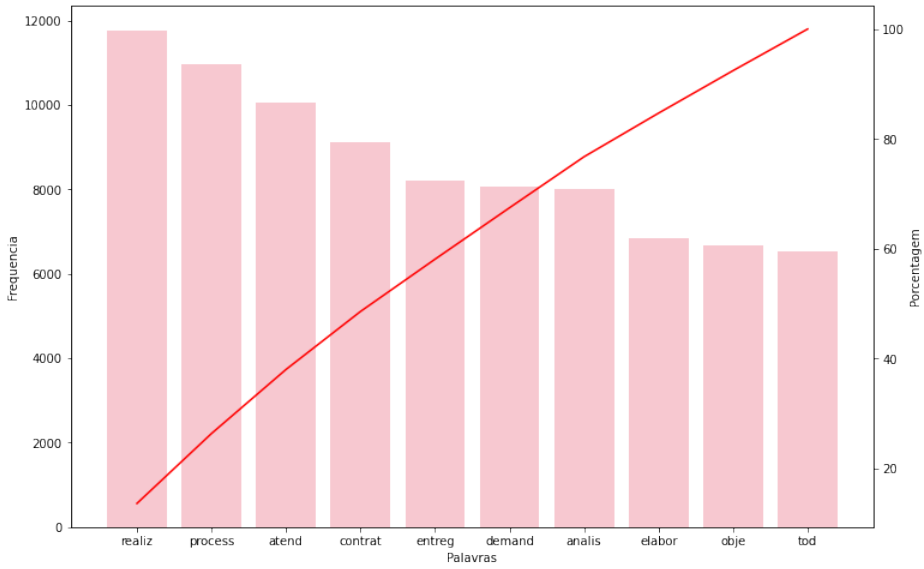
Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Gráfico 5 – Nuvem de palavras da classe de Acordos classificados como que não são SMART

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Ainda, enquanto etapa de pré-processamento, foi aplicada técnica de stemming que consiste na redução das palavras para o radical comum, evitando assim as inflexões de palavras que guardam o mesmo sentido e informação. Após aplicação do processo, comparando o Gráfico 3 com o Gráfico 6, é possível observar a elevação na quantidade do radical mais frequente diante da palavra “realizar”, por exemplo, que passa de um pouco mais de 7.000 repetições para cerca de 12.000 repetições.

Gráfico 6 – Dez radicais mais frequentes no corpus após a aplicação de stemming



Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Até o momento, todas as técnicas aplicadas para pré-processamento consideram os termos presentes nas frases do Acordo com o mesmo peso, ou seja, não diferencia palavras entre as classes. Com isso, palavras como “demanda”, “processo”, “realizar” que são frequentes tanto nos Acordos que são SMART como os que não são, possuem o mesmo peso que termos mais comuns em uma ou na outra classe. Um exemplo corriqueiro entre os Acordos que não são classificados como SMART, versam sobre realização de “capacitação”, “cursos” ou ‘treinamentos”, ações que são consideradas em outra dimensão do modelo de avaliação de desempenho.

Assim, a fim de diferenciar o peso relativos de palavras nas frases dos Acordos, Vijayan (2017) apresenta como método de ponderação o TF-IDF que atribui um peso levando em consideração a Frequência do Termo e a Frequência Inversa do Documento. Os seja, atribui maior peso para uma palavra no documento se a Frequência do termo, isto é, o número de vezes que a palavra aparece no documento, for alta, e menos peso se a frequência do documento, ou seja, o número de documentos de treinamento em que a palavra aparece é alto. Dentre os 10 radicais com maiores pesos por classe de Acordo, é possível observar naqueles que não são

SMART termos relacionados à capacitação e gestão da jornada de trabalho enquanto na classe considerada SMART os termos estão relacionados a “elaborar”, “propor”, “mapear”, “especificar”.

Por fim, foi aplicado o TF-IDF em conjunto a técnica de N-GRAM que, segundo Nascimento (2019) explica, é um método de verificação de “n” palavras contínuas de uma determinada sequência de texto, ou seja, ao invés de procurar, quanto importante é uma palavra para o corpus, procura-se a importância de uma sentença de duas ou mais palavras. Esse método foi adotado, utilizando bigrama, ou seja, conjunto de duas palavras, esperando um ganho semântico na interpretação dos resultados.

4 CLASSIFICADORES PARA MINERAÇÃO DE TEXTO

4.1 Algoritmos de classificação

Para geração dos modelos classificadores foram identificados 3 algoritmos citados frequentemente em estudos já realizados, o Support Vector Machine (SVM), (Máquina de Vetores de Suporte), Naive Bayes e Regressão Logística.

Segundo Zheng (2019), Support Vector Machine (SVM) é um modelo de classificação binária cujo modelo básico é de classificador linear que busca otimizar o cálculo da margem da reta maximizando essa margem. No entanto, o SVM possui variação para modelo não-linear, neste estudo foi testado o modelo linear (linear), não linear com a função radial (RBF) e polinomial.

O algoritmo de classificação Naive Bayes é um classificador baseado no teorema de Bayes. Este classificador utiliza-se de aprendizado supervisionado e método estatístico de classificação baseado no Teorema Bayesiano, calcula probabilidades para cada classe sendo também robusto a ruídos em dados (Flores, 2018).

Quanto à Regressão Logística, Zheng (2019) define como um modelo de classificação, expresso por uma distribuição de probabilidade condicional, na forma de distribuição logística parametrizada. A variável aleatória, neste estudo, as classes de Acordos, é um número real entre 0 e 1, sendo tratado como uma probabilidade de

estar em 0 ou 1, com corte em 0,5, ou seja, acima de 0,5 o modelo classifica como Acordo SMART, do contrário, como não SMART.

4.2 Problema do balanceamento de dados

Considerando a possibilidade de baixo desempenho na classificação gerada pelos algoritmos em função do desbalanceamento das classes, foram aplicadas 3 técnicas para balanceamento dos dados, SMOTE, Oversampling e Downsampling, a fim de comparar os seus resultados com o resultado dos classificadores com os dados nas suas proporções reais.

A técnica SMOTE, detalhada por Chawla (2002), altera as proporções reais dos dados para proporcionar equilíbrio entre as classes contando com a criação de exemplos sintéticos.

A técnica Oversampling, como descrito na documentação do Scikit Learn, se trata um processo de repetição de algumas amostras da classe minoritária para equilibrar o número de amostras entre as classes no conjunto de dados e as técnicas.

Enquanto o método Downsampling realiza a redução da classe majoritária, para que as classes de dados sejam equilibradas. Entre as técnicas de Downsampling foi aplicado RandomUnderSampler que reduz a classe majoritária removendo aleatoriamente dados da classe majoritária e a técnica NearMiss, apresentada na documentação do Scikit Learn com o uso de 3 regras diferentes. Na regra 1 são mantidos os pontos da classe majoritária que são semelhantes à classe minoritária. Na regra 2 são mantidos os pontos da classe majoritária cuja distância média para os pontos mais distantes da classe minoritária é menor, ou seja, manterá os pontos da classe majoritária que são mais diferentes da classe minoritária. Por fim, a regra 3 faz uso da seleção de “k” vizinhos mais próximos na classe majoritária para cada ponto da classe minoritária, nesse caso, o “k” definido aleatoriamente foi 4.

4.3 Aplicação dos algoritmos

Inicialmente apresenta-se os resultados agregados por técnica de classificação, considerando a aplicação dos dados com as classes nas proporções reais e, na sequência, aplicação com a inserção das técnicas de balanceamento

apresentadas no item 4.2, quais sejam: SMOTE, Oversampling e Undersampling – RandomUnderSampler, NearMiss1, NearMiss2 e NearMiss3.

A primeira aplicação considerou o uso do algoritmo de Regressão Logística, cuja estatísticas de análise foram obtidas utilizando cross validation, ou seja, foram construídas 3 amostras diferentes para cada aplicação e apresenta-se a título de exemplo na Tabela 1 o resultado da primeira aplicação, visto que não se observou diferenças significativas entre os resultados das amostras. A Tabela 2 apresenta os valores medianos das 3 amostras.

Tabela 1 – Resultado das estatísticas de avaliação das técnicas de amostragem e para a base com as proporções originais para o algoritmo de Regressão Logística na primeira amostragem do cross validation

Técnicas aplicadas		Estatísticas/ Classes	Precisão	Recall	F1 Score
Base Original		É SMART	0,87143677	0,98566405	0,92503744
		Não é SMART	0,88331132	0,42735407	0,57602297
Oversampling		É SMART	0,92804138	0,89304842	0,91020870
		Não é SMART	0,63327769	0,72731146	0,67704512
SMOTE		É SMART	0,92614748	0,89618610	0,91092049
		Não é SMART	0,63737717	0,71857691	0,67554576
UnderSampling	RandomUnder Sampler	É SMART	0,93305977	0,82418177	0,87524775
		Não é SMART	0,52561670	0,76714955	0,62381984
	NearMiss1	É SMART	0,91847684	0,75942656	0,83141343
		Não é SMART	0,43673211	0,73455475	0,54777981
	NearMiss2	É SMART	0,93098887	0,64733568	0,76367350
		Não é SMART	0,36868100	0,81103536	0,50692410
	NearMiss3	É SMART	0,91579594	0,80605897	0,85743059
		Não é SMART	0,48111159	0,70813805	0,57295527

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Tabela 2 – Resultado da média e desvio padrão das estatísticas de avaliação das técnicas de amostragem e para a base com as proporções originais para o algoritmo de Regressão Logística considerando as 3 amostras geradas por cross validation

Técnicas aplicadas		Acurácia	Precisão	Recall	F1 Score
Base Original		87,05% (+/- 0,15%)	87,74% (+/- 0,02%)	70,03% (+/- 0,44%)	74,41% (+/- 0,45%)
Oversampling		85,90% (+/- 0,18%)	78,01% (+/- 0,25%)	80,62% (+/- 0,33%)	79,17% (+/- 0,26%)
SMOTE		85,95% (+/- 0,18%)	78,09% (+/- 0,25%)	80,26% (+/- 0,44%)	79,07% (+/- 0,31%)
Undersampling	Random UnderSampler	81,15% (+/- 0,20%)	72,71% (+/- 0,25%)	79,08% (+/- 0,39%)	74,68% (+/- 0,28%)
	NearMiss1	75,21% (+/- 0,22%)	67,43% (+/- 0,23%)	74,17% (+/- 0,38%)	68,60% (+/- 0,26%)
	NearMiss2	65,65% (+/- 3,70%)	64,09% (+/- 1,31%)	71,57% (+/- 1,90%)	61,57% (+/- 2,97%)
	NearMiss3	78,70% (+/- 0,21%)	69,82% (+/- 0,23%)	75,49% (+/- 0,30%)	71,48% (+/- 0,25%)

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

A análise da Tabela 1, no que tange à medida de Precisão, é utilizada na mensuração de quantos dos dados classificados eram realmente daquela classe, aqui observa-se a técnica SMOTE com o melhor resultado para Acordos não SMART e a Random UnderSampler para os Acordos SMART. Enquanto o Recall, apresenta o resultado daqueles acordos de determinada classe, qual foi a proporção classificada como da classe, neste caso a NearMiss2 teve melhor desempenho para Acordos que não são SMART e a SMOTE para Acordos SMART. Por fim, o F1 Score é resultado da média harmônica entre Precisão e Recall, neste caso, a Oversampling apresentou o melhor resultado para a classe minoritária e SMOTE o melhor resultado para a classe majoritária.

Na Tabela 2, a Acurácia, que indica a performance geral do modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, não é uma boa medida para bases desbalanceadas, visto que seus resultados são influenciados por classes majoritárias. Já nos experimentos com balanceamento, verifica-se um

melhor desempenho, em termos de Acurácia para o modelo utilizando a técnica de SMOTE. Já para a medida de Precisão, novamente o melhor resultado foi da SMOTE. Enquanto para o Recall e F1 Score a Oversampling teve melhor desempenho.

Assim, para algoritmo de Regressão Logística, o balanceamento utilizando a técnica SMOTE obteve resultados mais robustos na maioria das métricas avaliadas.

Na segunda aplicação, foi utilizado o algoritmo Naive Bayes e suas estatísticas de análise estão dispostas na Tabela 3 e Tabela 4.

Tabela 3 – Resultado das estatísticas de avaliação das técnicas de amostragem e para a base com as proporções originais para o algoritmo Naive Bayes na primeira amostragem do cross validation

Técnicas		Estatísticas/ Classes	Precisão	Recall	F1 Score
Base Original		É SMART	0,85882463	0,99453611	0,92171166
		Não é SMART	0,9430344	0,35619940	0,51708675
Oversampling		É SMART	0,92187588	0,88796321	0,90460182
		Não é SMART	0,61462598	0,70366425	0,65613826
SMOTE		É SMART	0,92129112	0,89093860	0,90586067
		Não é SMART	0,61983783	0,70025565	0,65759728
UnderSampling	RandomUnder Sampler	É SMART	0,92820759	0,83792264	0,88075742
		Não é SMART	0,53850893	0,74478057	0,62506705
	NearMiss1	É SMART	0,90260382	0,88888288	0,89569081
		Não é SMART	0,58713568	0,62228377	0,60419899
	NearMiss2	É SMART	0,92535511	0,70484176	0,80018425
		Não é SMART	0,40037367	0,77609715	0,52823896
	NearMiss3	É SMART	0,89728745	0,90549094	0,90137053
		Não é SMART	0,61392265	0,59181934	0,60266840

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação

Tabela 4 – Resultado da média e desvio padrão das estatísticas de avaliação das técnicas de amostragem e para a base com as proporções originais para o algoritmo Naive Bayes considerando as 3 amostras geradas por cross validation

Técnicas aplicadas		Acurácia	Precisão	Recall	F1 Score
Base Original		86,36% (+/- 0,18%)	90,01% (+/- 0,19%)	67,10% (+/- 0,45%)	71,42% (+/- 0,54%)
Oversampling		85,21% (+/- 0,20%)	77,03% (+/- 0,29%)	79,19% (+/- 0,40%)	78,00% (+/- 0,27%)
SMOTE		85,34% (+/- 0,22%)	77,21% (+/- 0,32%)	79,12% (+/- 0,45%)	78,09% (+/- 0,32%)
Undersampling	RandomUnder Sampler	82,04% (+/- 0,19%)	73,35% (+/- 0,18%)	78,70% (+/- 0,33%)	75,23% (+/- 0,17%)
	NearMiss1	83,01% (+/- 0,56%)	73,81% (+/- 0,78%)	75,02% (+/- 0,48%)	74,38% (+/- 0,65%)
	NearMiss2	70,42% (+/- 3,31%)	65,42% (+/- 1,55%)	72,80% (+/- 1,74%)	65,05% (+/- 2,69%)
	NearMiss3	84,34% (+/- 0,27%)	75,83% (+/- 0,46%)	74,64% (+/- 0,36%)	75,20% (+/- 0,36%)

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Para o algoritmo Naive Bayes, em avaliação à Tabela 3, a Precisão, Recall e F1 Score apresentaram respectivamente melhor desempenho nas técnicas, SMOTE, NearMiss2 e SMOTE. Para a Tabela 4, a Acurácia, Precisão, Recall e F1 Score, os resultados mais satisfatórios foram SMOTE, SMOTE, Oversampling e SMOTE. Com isso, para algoritmo Naive Bayes, o balanceamento utilizando a técnica SMOTE obteve resultados mais robustos.

Por último, foi realizada a aplicação do algoritmo SVM. Neste caso, houve a necessidade de identificação da função de distribuição, parâmetro "kernel" do modelo, mais adequada para geração dos resultados. Assim, optou-se por identificar o parâmetro de melhor desempenho na base com as classes em suas proporções originais, conforme Tabela 5.

Tabela 5 – Resultado das estatísticas de avaliação para as funções de distribuição utilizadas na primeira amostragem do cross validation para o algoritmo SVM

Funções utilizadas	Estatísticas/Classes	Precisão	Recall	F1 Score
Linear	É SMART	0,88833905	0,98084934	0,93230492
	Não é SMART	0,87215601	0,51448650	0,64719282
RBF	É SMART	0,84515626	0,99772789	0,91512640
	Não é SMART	0,96904937	0,28014487	0,43463890
Polinomial	É SMART	0,87961592	0,99610495	0,93424324
	Não é SMART	0,96794301	0,46314444	0,62651297

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Tabela 6 – Resultado da média e desvio padrão das estatísticas para as funções de distribuição utilizadas na primeira amostragem do cross validation para o algoritmo SVM

Técnicas aplicadas	Acurácia	Precisão	Recall	F1 Score
Linear	88,53% (+/- 0,09%)	87,99% (+/- 0,21%)	74,42% (+/- 0,24%)	78,67% (+/- 0,22%)
RBF	85,13% (+/- 0,08%)	90,60% (+/- 0,15%)	63,64% (+/- 0,19%)	67,15% (+/- 0,26%)
Polinomial	88,69% (+/- 0,13%)	92,38% (+/- 0,32%)	72,61% (+/- 0,27%)	77,68% (+/- 0,30%)

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Em análise à Tabela 5, observa-se uma situação de alta precisão para a classe que não é SMART. A alta precisão indica que de todos os Acordos classificados como não sendo SMART, entre 98% e 99,7% são realmente não SMART. No entanto, observa-se para todas as funções baixo recall (quantidade de Acordos classificados como não SMART pela quantidade real de não SMART), demonstrando a tendência em classificar como não SMART inclusive Acordos que originalmente são SMART. O que pode estar indicando um sobre ajuste do modelo ao conjunto de dados, em especial, para a classe não SMART. Assim, optou-se por utilizar a métrica de F1 Score para escolha do parâmetro do algoritmo SVM, neste caso, a função “linear” foi a utilizada para o cálculo dos classificadores para as bases

balanceadas, apresentado também como maior valor do parâmetro de recall e F1 Score, como observado na Tabela 6.

Na sequência, assim como para os demais algoritmos foram realizados os balanceamentos das classes, e suas estatísticas de análise estão demonstradas na Tabela 7 e Tabela 8.

Tabela 7 – Resultado das estatísticas de avaliação para as funções de distribuição utilizadas na primeira amostragem do cross validation para o algoritmo SVM

Técnicas		Estatísticas/ Classes	Precisão	Recall	F1 Score
Base Original		É SMART	0,88833905	0,98084934	0,93230492
		Não é SMART	0,87215601	0,51448650	0,64719282
Oversampling		É SMART	0,92418135	0,90538274	0,91468547
		Não é SMART	0,65502959	0,70749893	0,68025399
SMOTE		É SMART	0,92444420	0,90879091	0,91655073
		Não é SMART	0,66327142	0,70749893	0,68467168
UnderSampling	RandomUnder Sampler	É SMART	0,93688509	0,81909656	0,87404029
		Não é SMART	0,52351097	0,78270132	0,62739071
	NearMiss1	É SMART	0,91910069	0,77625101	0,84165762
		Não é SMART	0,45341615	0,73093311	0,55966071
	NearMiss2	É SMART	0,93606519	0,64630782	0,76465694
		Não é SMART	0,37231183	0,82616106	0,51330245
	NearMiss3	É SMART	0,91961796	0,83862591	0,87725652
		Não é SMART	0,52815565	0,71133362	0,60620915

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Tabela 8 – Resultado da média e desvio padrão das estatísticas de avaliação das técnicas de amostragem e para a base com as proporções originais para o algoritmo SVM considerando as 3 amostras geradas por cross validation

Técnicas aplicadas		Acurácia	Precisão	Recall	F1 Score
Base Original		88,53% (+/- 0,09%)	87,99% (+/- 0,21%)	74,42% (+/- 0,24%)	78,67% (+/- 0,22%)
Oversampling		86,49% (+/- 0,18%)	78,93% (+/- 0,27%)	80,39% (+/- 0,26%)	79,62% (+/- 0,25%)
SMOTE		86,67% (+/- 0,16%)	79,22% (+/- 0,22%)	80,30% (+/- 0,42%)	79,74% (+/- 0,30%)
Undersampling	RandomUnder Sampler	81,31% (+/- 0,12%)	73,09% (+/- 0,12%)	79,99% (+/- 0,17%)	75,14% (+/- 0,13%)
	NearMiss1	76,76% (+/- 0,18%)	68,60% (+/- 0,05%)	75,22% (+/- 0,14%)	70,05% (+/- 0,08%)
	NearMiss2	65,74% (+/- 4,21%)	64,51% (+/- 1,40%)	72,20% (+/- 2,09%)	61,82% (+/- 3,38%)
	NearMiss3	81,10% (+/- 0,27%)	72,13% (+/- 0,35%)	77,12% (+/- 0,44%)	73,88% (+/- 0,38%)

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Para o algoritmo SVM, em avaliação à Tabela 8, a Precisão, Recall e F1 Score apresentaram melhor desempenho na técnica SMOTE. Para a Tabela 8, a Acurácia, Precisão e F1 Score, os resultados mais satisfatórios foram SMOTE, somente apresentando desempenho médio superior 0,09% para a métrica Recall.

Com isso, para algoritmo SVM, com função linear, o balanceamento utilizando a técnica SMOTE obteve resultados mais satisfatórios.

Após identificados os balanceamentos para cada algoritmos com melhores desempenhos na classificação das classes, toma-se como segunda etapa a comparação entre os algoritmos a fim de identificar o mais adequado para predição quanto à classe dos “Acordos de Desempenho”.

A Tabela 9 e a Tabela 10 apresentam as estatísticas das métricas dos balanceamentos identificados como mais robustos por algoritmo.

Tabela 9 – Resultado das estatísticas de avaliação das técnicas de amostragem mais robustas por algoritmo de classificação na primeira amostragem do cross validation

Técnicas	Estatísticas/ Classes	Precisão	Recall	Score
Regressão Logística	É SMART	0,92614748	0,89618610	0,91092049
	Não é SMART	0,63737717	0,71857691	0,67554576
Naive Bayes	É SMART	0,92129112	0,89093860	0,90586067
	Não é SMART	0,61983783	0,70025565	0,65759728
SVM	É SMART	0,92444420	0,90879091	0,91655073
	Não é SMART	0,66327142	0,70749893	0,68467168

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Tabela 10 – Resultado da média e desvio padrão das estatísticas de avaliação das técnicas de amostragem mais robustas por algoritmo de classificação considerando as 3 amostras geradas por cross validation

Algoritmo	Acurácia	Precisão	Recall	F1 Score
Regressão Logística	85,90% (+/- 0,18%)	78,01% (+/- 0,25%)	80,62% (+/- 0,33%)	79,17% (+/- 0,26%)
Naive Bayes	85,34% (+/- 0,22%)	77,21% (+/- 0,32%)	79,12% (+/- 0,45%)	78,09% (+/- 0,32%)
SVM	86,67% (+/- 0,16%)	79,22% (+/- 0,22%)	80,30% (+/- 0,42%)	79,74% (+/- 0,30%)

Fonte: Produzido pelos autores do trabalho com os dados disponibilizados pela Empresa detentora da informação.

Na Tabela 9 observa-se que das 6 estatísticas avaliadas, 4 delas apresentaram resultado maior no algoritmo SVM, inclusive na métrica F1 Score apresentou os melhores resultados para as duas classes. Resultado confirmado em avaliação à Tabela 10 onde a Acurácia, Precisão e F1 Score apresentaram melhor desempenho com o uso do algoritmo SVM e apenas a métrica Recall obteve melhor resultado médio para o algoritmo de Regressão Logística.

Com isso, para algoritmo SVM, com função linear, utilizando a técnica SMOTE de balanceamento obteve os resultados mais robustos entre as combinações de técnicas utilizadas. Cabe ressaltar que o tempo de processamento para geração

dos modelos foram muito superior para o algoritmo SVM quando comparados aos demais, cabendo avaliar o custo benefício do ganho em utilizar o algoritmo SVM frente à Regressão Logística (diferença de 0,77 pontos percentuais de Acurácia) e Naive Bayes (diferença de 1,33% na Acurácia).

5 CONCLUSÃO

A classificação de textos é assunto indispensável nos tempos atuais dada a quantidade de informação no formato textual existente e o potencial de inteligência que pode ser extraído dessas bases.

O estudo permitiu compreender que para um mesmo corpus é possível aplicar técnicas diferentes a fim de avaliar as mais adequadas para os problemas propostos. Nesta aplicação, ressalta-se a importância na realização das etapas de pré-processamento como retiradas de stop words, ponderação dos termos por meio de TF-IDF e, em especial, o tratamento do desbalanceamento das classes. Pode-se concluir que das técnicas aplicadas nos diferentes algoritmos a SMOTE foi a mais eficaz em todos os casos.

Quanto a avaliação do algoritmo mais robusto, o SVM teve melhor desempenho em avaliação às métricas de análise, no entanto, ressalta-se a necessidade de avaliação quanto ao tempo de processamento do algoritmo SVM que é muito superior aos demais algoritmos, visto a possibilidade de estudos futuros agregando todos os Acordos da empresa, que passaria de uma base de cerca de 65.000 Acordos para 250.000 Acordos com diferença de ganho entre 0,77% e 1,33%.

REFERÊNCIAS

ARANHA, Christian; PASSOS, Emmanuel. A tecnologia de mineração de textos. Revista Eletrônica de Sistemas de Informação, v. 5, n. 2, 2006.

BRANDÃO, Hugo Pena et al. Gestão de desempenho por competências: integrando a gestão por competências, o balanced scorecard e a avaliação 360 graus. Revista de Administração Pública, v. 42, n. 5, p. 875-898, 2008.

CARDOSO, Adriano Lindon Leite et al. Planejamento de metas para redução de falhas no processo de distribuição de uma empresa transportadora. Revista Gestão Industrial Ponta Grossa, v. 14, n. 2, p. 206-226, 2018.

CHAWLA, Nitesh V. et al. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, v. 16, p. 321-357, 2002.

DE AQUINO GUIMARÃES, Tomás; NADER, Rosa Maria; RAMAGEM, Sérgio Pinela. Avaliação de desempenho de pessoal: uma metodologia integrada ao planejamento e à avaliação organizacionais. Revista de Administração Pública, v. 32, n. 6, p. 43-61, 1998.

FELDMAN, R. Techniques and applications for sentiment analysis. Communications of the ACM, ACM, v. 56, n. 4, p. 82–89, 2013.

FLORES, Andrew Christian et al. An evaluation of svm and naive bayes with smote on sentiment analysis data set. In: 2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST). IEEE, 2018. p. 1-4.

LOTTA, Gabriela Spanguero. Avaliação de desempenho na área pública: perspectivas e propostas frente a dois casos práticos. RAE eletrônica, v. 1, n. 2, p. 1-12, 2002.

NASCIMENTO, Robson Murilo Ferreira do. Classificação automática de discursos de ódio em textos do Twitter. 2019. Trabalho de Conclusão de Curso. Brasil.

OLIVEIRA, Felipe R. Modelos lineares e não lineares aplicados à análise de sentimentos de consumidores de e-commerce no Brasil. 2020. Disponível em https://www.researchgate.net/publication/344772477_modelos_lineares_e_nao_lineares_aplicados_a_analise_de_sentimentos_de_consumidores_de_e-commerce_no_brasil, Acesso em 29/08/2021.

RIBEIRO, Luiz Carlos Felix. Análise de Sentimento Contextual em Diálogos Utilizando Aprendizado de Máquina / Luiz Carlos Felix Ribeiro. – Bauru, 2019 113.

SANTOS, Ronnie E. S. et al. Técnicas de processamento de linguagem natural aplicadas ao processo de mineração de textos: resultados preliminares de um mapeamento sistemático. Revista de Sistemas e Computação-RSC, v. 4, n. 2, 2015.

SARAKIT, Phakhawat; THEERAMUNKONG, Thanaruk; HARUECHAIYASAK, Choochart. Improving emotion classification in imbalanced YouTube dataset using SMOTE algorithm. In: 2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA). IEEE, 2015. p. 1-5.

VIJAYAN, Vikas K.; BINDU, K. R.; PARAMESWARAN, Latha. A comprehensive study of text classification algorithms. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2017. p. 1109-1113.

YUAN, Ding et al. Sentiment analysis of microblog combining dictionary and rules. In: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014). IEEE, 2014. p. 785-789.

ZHANG, Xueying; ZHENG, Xianghan. Comparison of text sentiment analysis based on machine learning. In: 2016 15th international symposium on parallel and distributed computing (ISPDC). IEEE, 2016. p. 230-233.

ZHENG, Yuhan. An exploration on text classification with classical machine learning algorithm. In: 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI). IEEE.

EXTRAÇÃO DE TEXTOS DA CNH COM YOLO, ESRGAN E TESSERACT OCR

Arthur Porfírio de Castro Siqueira
William Roberto Malvezzi

RESUMO

O artigo em questão disserta sobre aplicação de técnicas de aprendizado de máquina na extração de textos de imagens de CNHs (Carteira Nacional de Habilitação) para viabilizar a automação de partes de processos de cadastro e confirmação de identidade em plataformas digitais. As técnicas de machine learning aplicadas foram o YOLOv3 para detecção dos campos da CNH, ESRGAN para aumentar a qualidade das imagens e tesseract OCR (reconhecimento óptico de caracteres) para realizar a extração dos textos das imagens. Os resultados obtidos mostram-se satisfatórios tendo em vista que a combinação das tecnologias aplicadas foi capaz de extrair os textos da CNH com acurácia acima de 90%.

Palavras-chave: Extração de Textos. YOLOv3. ESRGAN. TESSERACT OCR. Machine Learning.

1 INTRODUÇÃO

Atualmente, serviços e plataformas digitais têm ganhado cada vez mais popularidade e, muitos deles, utilizam sistemas de cadastro e de comprovação de identidade que exigem o envio de uma foto da Carteira Nacional de Habilitação (CNH) para ser analisada por um agente humano. Assim, utilizando YOLO, Transfer Learning, ESRGAN, LSTM baseado em reconhecimento óptico de caracteres e expressões regulares, o presente trabalho busca viabilizar a automação de uma parte dos processos de cadastro e confirmação de identidade em ambientes digitais.

O YOLO (You Only Look Once)(REDMON et al., 2016) é um sistema de detecção de objetos em tempo real que aplica uma única rede neural convolucional (CNN) à imagem completa. Por ter capacidade de realizar processamento em

Unidades de Processamento Gráfico (GPU) e realizar previsões com uma única avaliação da rede neural, seu treinamento e aplicação tornam-se mais rápidos que outras técnicas de redes neurais para identificação de objetos em imagens e, ainda assim, mantendo elevadas taxas de acurácia.

O tesseract OCR é um excelente algoritmo de reconhecimento ótico de caracteres em imagens (SMITH, 2016), capaz de transcrever textos de imagens com alta acurácia. Entretanto, imagens com baixa qualidade resultam em extrações de textos incompletas ou com caracteres errôneos.

A maioria dos serviços que requerem o envio da CNH para cadastro ou validação de dados recebem imagens do documento capturadas através de câmeras de smartphones. Embora essas câmeras sejam capazes de registrar fotos com excelentes qualidades, alguns campos do documento possuem textos muito pequenos e, em alguns casos, podem ficar distorcidos ou ilegíveis. Como forma de contornar este problema que pode afetar a extração de textos da CNH, foram utilizadas redes neurais geradoras de super resolução (ESRGAN) (WANG et al., 2018) nas imagens antes da extração dos textos.

A finalidade geral deste trabalho é realizar a extração dos textos dos campos de uma foto de CNH com acurácia suficiente para utilização das informações em plataformas digitais e, para isto, treinar e aplicar uma rede neural supervisionada com o algoritmo YOLO para identificar os campos da CNH na imagem, recortar os campos identificados, utilizar o algoritmo de rede neural ESRGAN para aumentar a qualidade dos recortes, realizar a extração do texto dos recortes com o Tesseract OCR e por fim, verificar a porcentagem de similaridade entre o texto extraído e o texto real de cada campo da CNH. Ainda dentro do contexto do presente trabalho, visando enriquecer o conhecimento da comunidade científica sobre a eficácia de aplicar ESRGAN em imagens antes de extrair textos, é realizada uma comparação entre a acurácia da extração dos textos dos recortes com aplicação do ESRGAN e a acurácia da extração dos textos dos recortes sem a aplicação do ESRGAN.

2 REFERENCIAL TEÓRICO

Nesta secção são abordadas todas as tecnologias utilizadas no projeto e as principais referências teóricos.

2.1 Inteligência artificial

A IA (acrônimo de Inteligência Artificial) engloba todas as técnicas que viabilizam dispositivos capazes de solucionar problemas ou realizar tarefas que, quando executadas por seres humanos, requeiram a inteligência cognitiva (RAPHAEL, 1976). O conceito IA surgiu em meados do século XX, quando o cientista John McCarthy começou a abordar o termo, mas só com o aumento da capacidade computacional no início da década de 1980 que começaram a aparecer grandes avanços na área, como os primeiros algoritmos de Machine Learning capazes de executar tarefas específicas (FRADKOV, 2020).

Diversas abordagens de Machine Learning foram criadas, mas nenhuma se destacou tanto ao longo dos anos como as Redes Neurais Artificiais (SHAW, 1986), que foram criadas inspiradas na biologia do cérebro humano. Esta abordagem simula os neurônios do cérebro com a transmissão de informação entre neurônios, mas, diferentemente de um cérebro biológico em que os neurônios podem se conectar com outros neurônios em diversas direções, as Redes Neurais Artificiais possuem camadas de neurônios e uma direção em que os dados podem ser propagados, formando, segundo Haykin (2007), um elo do conjunto de sinapses com pesos específicos. Por exigir mais capacidade computacional que diversos outros algoritmos da época em que foram criados, inicialmente esta abordagem era pouco usada. Entretanto, com a evolução da capacidade computacional, novas pesquisas com redes neurais artificiais foram realizadas, e constatou-se que em muitos casos, a capacidade de solucionar problemas mais complexos era bem superior à capacidade de outros algoritmos (FRADKOV, 2020).

Em meados de 2010, pesquisadores começaram a criar grandes redes neurais artificiais, aumentando o número de camadas de neurônios, o que gerou algoritmos com maior capacidade de reconhecimento de imagens e padrões mais complexos. Devido ao fato de estes algoritmos possuírem diversas camadas de neurônios, o

cientista da computação Andrew Ng, em 2012, denominou estes algoritmos de Machine Learning como Deep Learning (Aprendizado profundo) (COOPELAND, 2016).

2.2 Machine learning

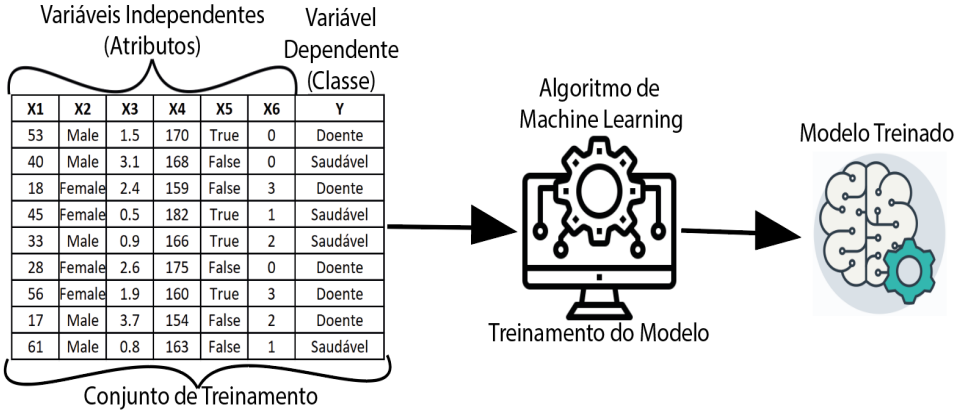
O ML (acrônimo de "Machine Learning" – Aprendizado de Máquina) é um ramo da IA que estuda técnicas computacionais e métodos estatísticos para permitir que um software (máquina) adquira conhecimento e altere seu comportamento automaticamente, à medida que são realizadas interações com os dados (KOVÁCS, 2006). Os algoritmos de ML podem ser supervisionados ou não supervisionados, sendo um sistema de ML supervisionado um programa de computador que toma decisões indutivas a partir de exemplos previamente apresentados, como a solução de problemas anteriores. Já um sistema de ML não supervisionado gera inferências a respeito dos dados apresentados.

2.2.1 Modelos supervisionados

Os algoritmos de ML supervisionados necessitam que os registros presentes no conjunto de dados tenham classes definidas. Existem diversos algoritmos de ML supervisionados, cada um com métodos estatísticos e aplicações diferentes. Para realizar previsões e inferências utilizando algoritmos de ML supervisionados, é necessário realizar o treinamento de um modelo de ML. Este é um algoritmo capaz de adquirir conhecimento com dados de um determinado contexto.

No treinamento do modelo, é fornecido, ao algoritmo de ML, um conjunto de exemplos de treinamento com cada exemplo (registro) sendo formado por um vetor de atributos (variáveis independentes) e o rótulo da classe associada (variável dependente), conforme retratado na Figura 1. Nesta etapa, é preciso determinar qual algoritmo será utilizado, o tamanho dos dados de entrada (input shape) e a quantidade de classes únicas. No exemplo da Figura 1, o input shape é 6, pois cada vetor de atributos possuí 6 atributos (dados de X1 até X6) e a quantidade de classes é 2 (Doente e Saudável).

Figura 1 – Representação do treinamento de um algoritmo de ML supervisionado

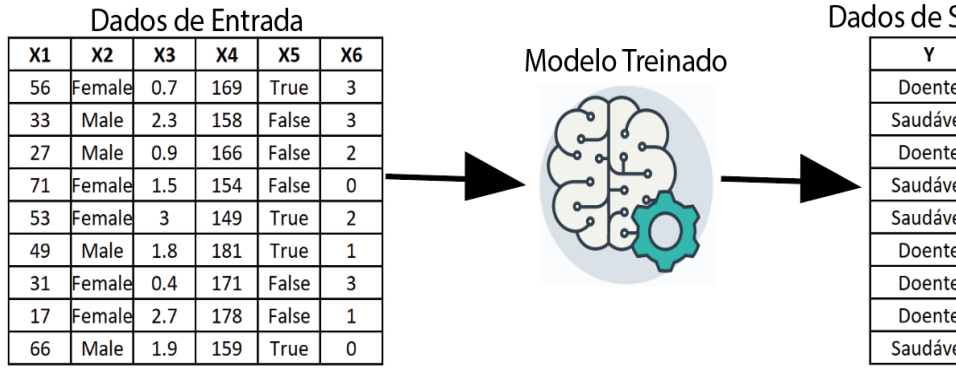


X1	X2	X3	X4	X5	X6	Y
53	Male	1.5	170	True	0	Doente
40	Male	3.1	168	False	0	Saudável
18	Female	2.4	159	False	3	Doente
45	Female	0.5	182	True	1	Saudável
33	Male	0.9	166	True	2	Saudável
28	Female	2.6	175	False	0	Doente
56	Female	1.9	160	True	3	Doente
17	Male	3.7	154	False	2	Doente
61	Male	0.8	163	False	1	Saudável

Fonte: Do Autor.

Após a criação e treinamento do modelo, é possível utilizá-lo para realizar previsões e inferências. A Figura 2 ilustra este processo, que é iniciado com o fornecimento de um conjunto de dados de entrada (Atributos) com a mesma estrutura dos dados de treinamento (mesmo input shape) ao modelo treinado, porém, sem fornecer a classe. O modelo irá associar uma das classes do conjunto de treinamento a cada um dos vetores de atributos fornecidos, gerando assim, os dados de saída.

Figura 2 – Aplicação de um modelo treinado de ML supervisionado



X1	X2	X3	X4	X5	X6
56	Female	0.7	169	True	3
33	Male	2.3	158	False	3
27	Male	0.9	166	False	2
71	Female	1.5	154	False	0
53	Female	3	149	True	2
49	Male	1.8	181	True	1
31	Female	0.4	171	False	3
17	Female	2.7	178	False	1
66	Male	1.9	159	True	0

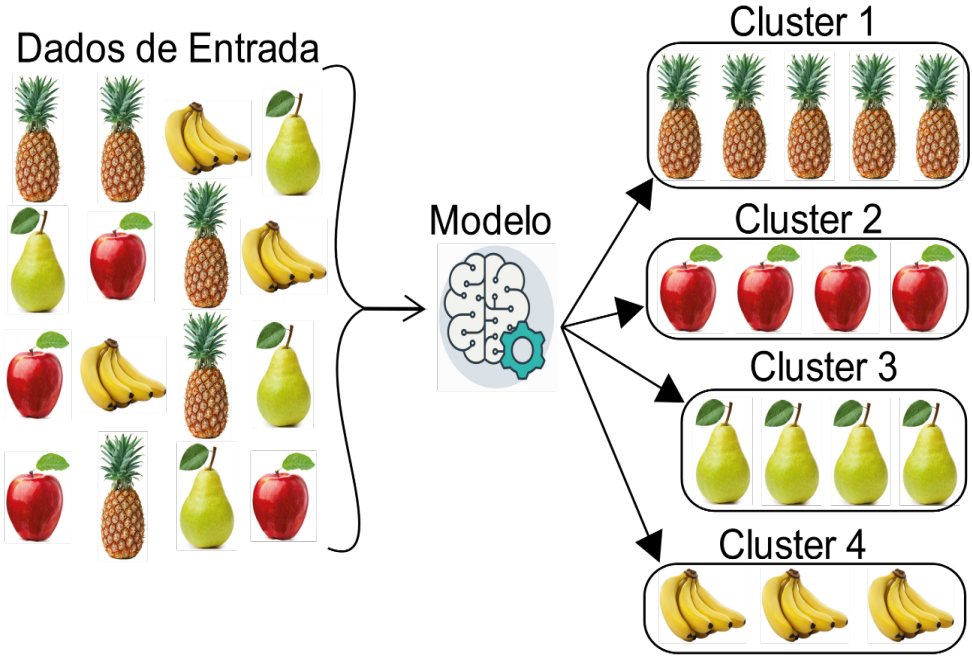
Y
Doente
Saudável
Doente
Saudável
Saudável
Doente
Doente
Doente
Saudável

Fonte: Do Autor.

2.2.2 Modelos não supervisionados

Nos algoritmos não supervisionados, os registros dos dados de exemplo fornecidos não possuem classes definidas, sendo função do algoritmo determinar quais registros podem ser agrupados, gerando agrupamentos (clusters). Com os clusters definidos, geralmente é necessário realizar uma análise para determinar o significado de cada cluster no contexto dos dados. A Figura 3 representa a aplicação de um algoritmo de clusterização em uma base de dados e as saídas geradas (clusters).

Figura 3 – Aplicação de um modelo de ML não supervisionado



Fonte: Do Autor.

2.3 Deep learning

O DL (acrônimo de “Deep Learning”), assim como o ML, é um algoritmo que irá receber dados de entrada de treino e resultará em um modelo treinado que também é capaz de receber dados de entrada e produz uma saída (classificações).

As redes neurais artificiais possuem 3 tipos de camada de neurônios:

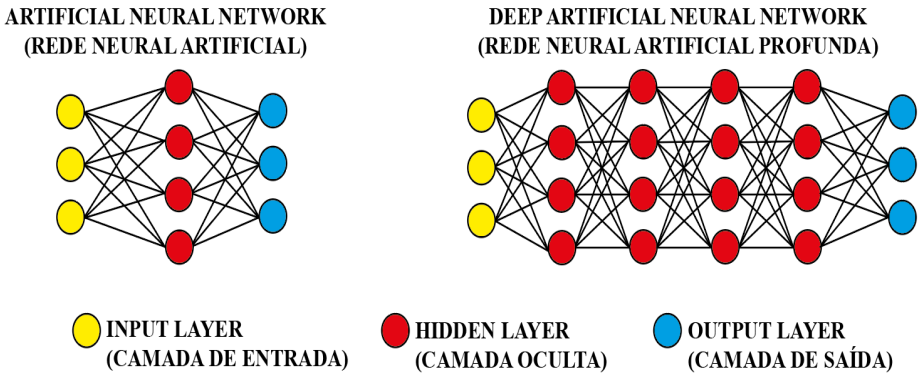
Camada de Entrada (input layer) – Responsável por receber os atributos do registro, sendo um neurônio para cada atributo. Assim sendo, a quantidade de neurônios nesta camada é a mesma quantidade de atributos presentes no vetor de atributos;

Camada Oculta (hidden layer) – Esta camada de neurônios inicia-se após a camada de entrada e pode haver diversas camadas ocultas subsequentes. A quantidade de neurônios em cada camada oculta independe de características dos dados;

Camada De Saída (output layer) – Camada que recebe os dados da última camada oculta e é responsável por determinar a qual classe o registro pertence. Nesta camada existe um neurônio para cada classe existente na base de treinamento.

O que diferencia as redes neurais artificiais das redes neurais artificiais profundas é a quantidade de camadas ocultas. Como exemplificado na Figura 4, as redes neurais artificiais possuem somente uma camada oculta, ao passo que as redes neurais artificiais profundas possuem duas ou mais camadas ocultas (COOPELAND, 2016).

Figura 4 – Estrutura de rede neural e estrutura de neural artificial



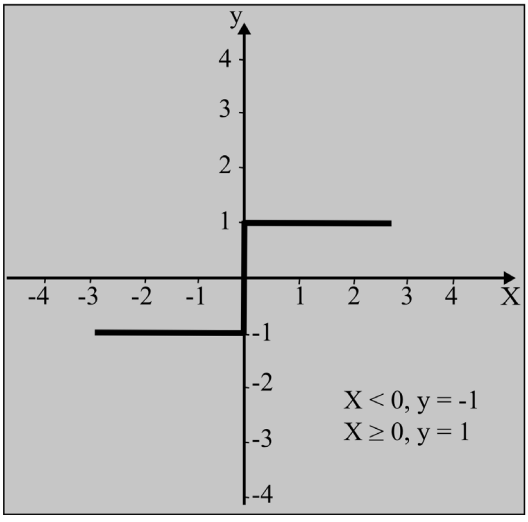
Fonte: Do Autor.

O funcionamento das redes neurais artificiais (profundas ou não) é realizado por meio de pesos entre as sinapses, uma notação sigma (Soma) e uma função de ativação.

No treinamento de um modelo de rede neural artificial, os pesos são utilizados para multiplicar o valor de um atributo, sendo o peso o responsável por agregar conhecimento à rede. Inicialmente os pesos recebem valores aleatórios e vão sendo atualizados conforme o treinamento vai sendo realizado. Após a definição dos pesos, é realizada uma soma (representada pela notação sigma) da multiplicação do valor de todas as entradas por seus respectivos pesos, o que resulta em um valor a ser passado à função de ativação.

Existem diversas funções de ativação e elas definem a partir de qual valor o neurônio será "ativado", fazendo com que passe informação aos neurônios subsequentes (BROWNLEE, 2021). Conforme ilustrado na Figura 5 a função de ativação degrau, enquanto o valor de X (valor que é fornecido pela notação sigma) for negativo, o valor de Y é -1 e o neurônio não é ativado, mas a partir do momento em que X é maior ou igual a zero, Y se torna 1 e o neurônio é ativado.

Figura 5 – Representação da função de ativação step (degrau)

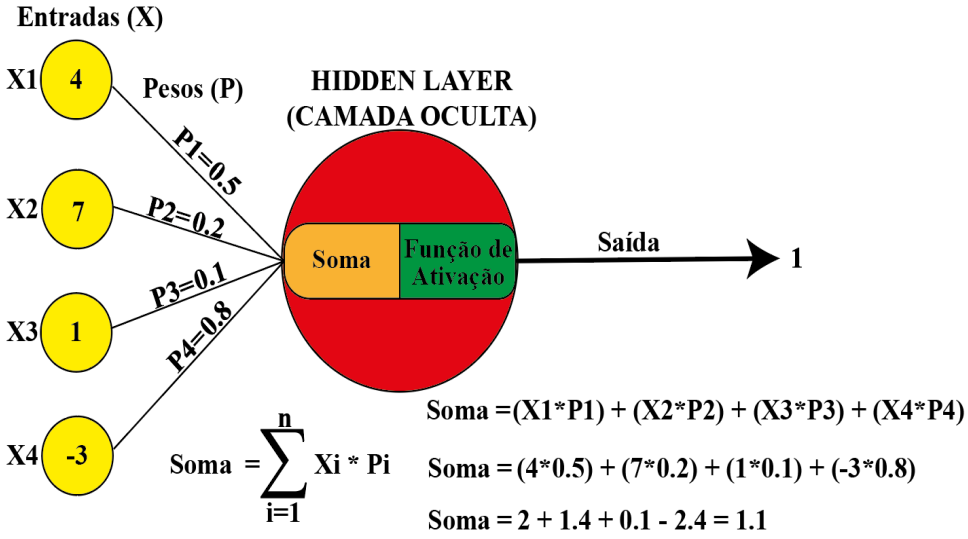


Fonte: Do Autor.

Conforme representada na Figura 6, inicialmente os dados da camada de entrada (X) serão multiplicados por um peso (P) e irão para o neurônio da camada oculta. Após o processamento da soma dos dados de entrada pelos pesos, caso o valor obtido seja suficiente para ativar a função de ativação, o neurônio da camada

oculta desempenha o papel de uma camada de entrada e fornece um valor para a camada subsequente, que pode ser outra camada oculta ou a camada de saída.

Figura 6 – Estrutura de pesos dos neurônios da camada de entrada e da camada oculta



Fonte: Do Autor.

No processo de treinamento, o modelo consegue adquirir conhecimento sobre os dados com a alteração dos pesos após cada iteração (processo em que os dados de entrada percorrem todo o modelo), de modo que a quantidade de erros seja a menor possível. Existem diversas funções e cálculos matemáticos que podem ser utilizados para realizar a alteração dos pesos à medida que as iterações vão ocorrendo.

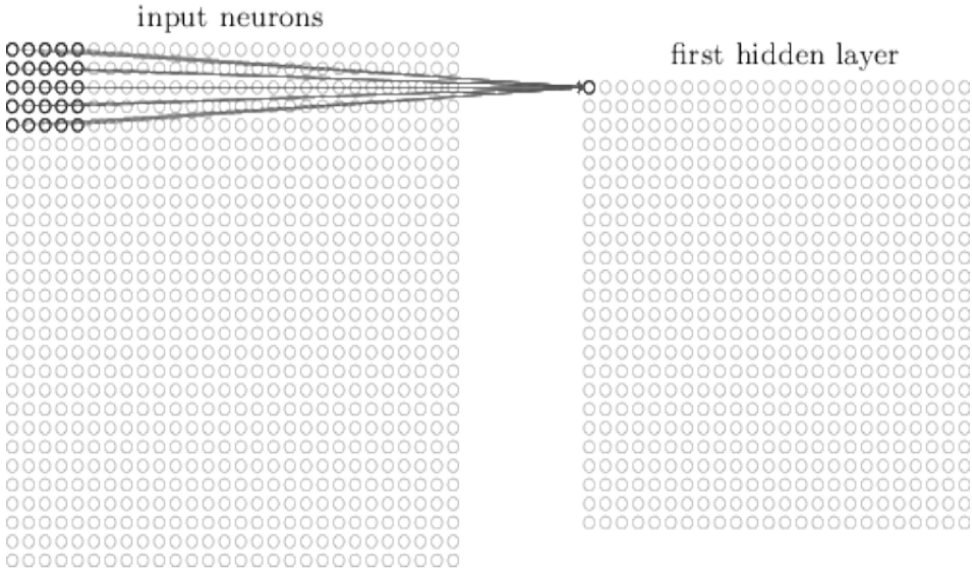
2.4 Rede Neural Convolucional - CNN

A CNN é uma subárea do DL capaz de simular a capacidade cognitiva humana na detecção de objetos em imagens baseado nos campos receptivos do córtex visual (FUKUSHIMA, 1982). Diferente de como os humanos enxergam imagens, o computador interpreta matrizes de valores. Uma imagem de dimensão 32x32 pixels possuí a sua matriz representativa no formato 32x32.

As redes convolucionais possuem 3 tipos de neurônios (YAMAGUCHI et al.,1990). A primeira camada sempre será uma camada de convolução, que mapeia regiões da imagem por meio de filtros. Conforme ilustrado na Figura 7, um filtro de

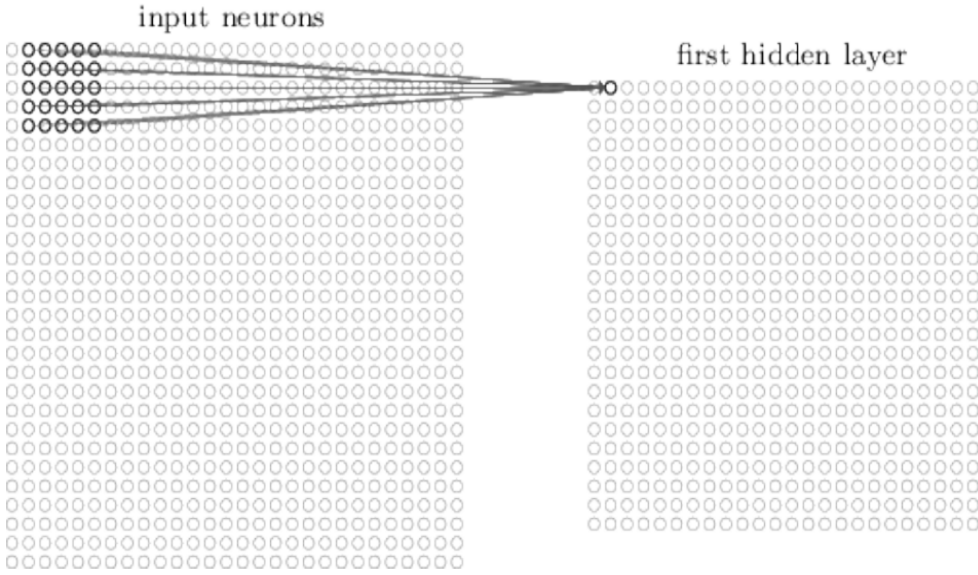
5x5 na matriz da imagem 32x32, o filtro percorre regiões de 5x5 pixels na imagem e em cada região, será realizada a soma da multiplicação entre os valores dos pixels pelos valores dos pesos do filtro. O valor desta somatória é passado para o mapa de recursos que é uma matriz de 28x28 (quantidade de posições possíveis que o filtro 5x5 pode ter dentro da matriz 32x32). Em seguida, conforme ilustrado na Figura 8, o filtro desloca-se uma posição à direita, e gera um novo valor para ser adicionado ao mapa de recursos, seguindo este processo até o filtro 5x5 percorrer as 784 possíveis posições na matriz 32x32.

Figura 7 – Representação da camada convolucional mapeando a imagem.



Fonte: Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

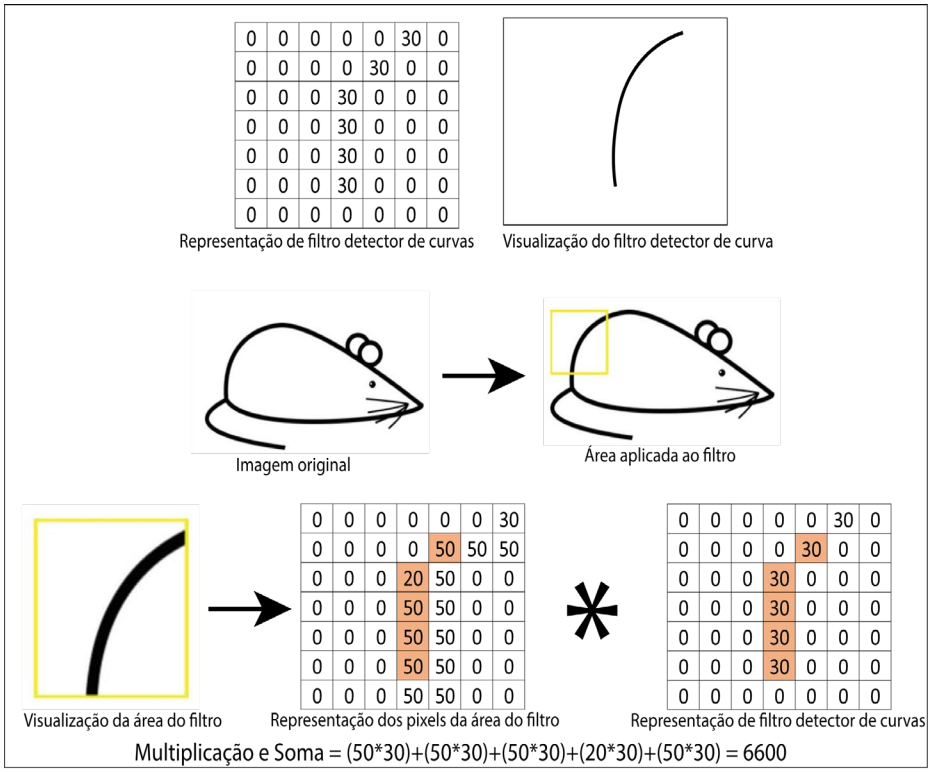
Figura 8 – Representação do filtro se deslocando dentro da matriz 32x32



Fonte: Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

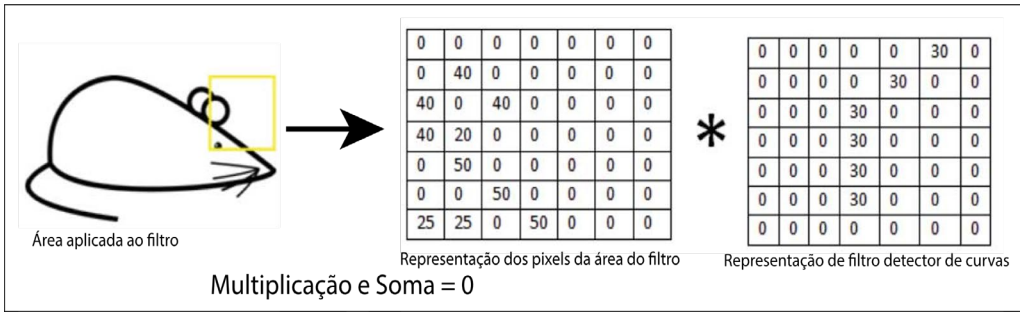
A Figura 9 representa a aplicação de um filtro 7x7 na uma região de uma imagem que corresponde ao filtro. Já a Figura 10 ilustra a aplicação do mesmo filtro em uma região da imagem sem correspondência com o filtro.

Figura 9 – Representação de um filtro 7x7 aplicado à área de uma imagem correspondente ao filtro.



Fonte: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

Figura 10 – Resultado da aplicação do filtro 7x7 em parte da imagem que não corresponde ao filtro

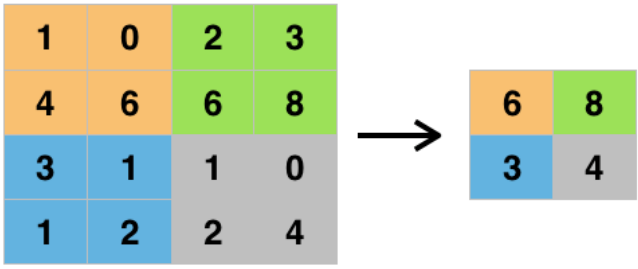


Fonte: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

As camadas convolucionais executam, basicamente, operações lineares, porque as únicas operações realizadas serão multiplicações e somas. Para acrescentar não linearidade ao modelo, é comum acrescentar uma ou mais camadas com função de ReLU após cada camada convolucional (VEDALDI; LENC, 2015).

Após as camadas com função de ativação ReLU, podem vir outras camadas de convolução ou a camada de pooling. Assim como a camada de convolução, a camada de pooling também utiliza filtros para realizar uma redução da dimensionalidade dos dados, mas percorre os dados e calcula o valor do filtro de forma diferente das camadas de convolução. O filtro da camada de pooling percorre os dados sem repetir os locais em que já esteve, e o valor retornado de cada posição do filtro geralmente é o valor máximo local (Max Pooling). O funcionamento desta camada está ilustrado na Figura 11, com a camada de pooling recebendo o mapa de recursos 4x4 criado pelas camadas convolucionais e aplicando um filtro de 2x2, que percorre o mapa de convolução e retorna apenas o maior valor encontrado na região do filtro. Este tipo de sumarização nos dados é útil para diminuir a quantidade de pesos a serem aprendidos pelo modelo, evitando overfiting (modelo se adequar muito às imagens de treinamento e não performar bem com os dados de teste) (HINTON et al.,2012).

Figura 11 – Representação da camada de pooling aplicando um filtro de 2x2 em um mapa de resultados



Fonte: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

Após o processo de pooling, é necessária uma camada totalmente conectada onde a saída é composta por N neurônios, com N sendo a quantidade de classes presentes no modelo (VEDALDI; LENC, 2015). Nesta etapa é realizada a

classificação gerada a partir dos dados transformados na camada anterior (pooling), sendo a camada final da CNN também uma camada totalmente conectada que utiliza a função de ativação SoftMax, que retorna um vetor de valores cuja soma é 1 e possui o mesmo número de valores que a quantidade de classes, sendo cada posição do vetor a representação de uma classe o seu valor representa a probabilidade de o elemento pertencer à classe, sendo o maior número do vetor a classe prevista pelo modelo.

2.5 Yolo

O algoritmo de machine learning YOLO ("You Only Look Once") é um detector de imagens considerado totalmente convolucional por ter seus recursos de aprendizado formados por 75 redes convolucionais (REDMOND; FARHADI, 2017). A principal funcionalidade deste algoritmo é identificar objetos específicos (é um algoritmo supervisionado por necessitar que sejam definidas as classes que o modelo deve identificar), suas respectivas localizações nas imagens e determinar a porcentagem de certeza de o objeto identificado pertencer à classe. A localização do objeto na imagem é dada por 4 parâmetros, sendo "x_center" e "y_center" correspondentes ao centro do objeto, "w" a largura do objeto e "h" a altura.

O YOLOv3 usa a estrutura de rede básica DarkNet-53, que utiliza 106 camadas. Devido ao fato de utilizar um filtro 1x1 nas 3 últimas reduções de dimensionalidade (downsampling) e filtros maiores ao longo da rede, o modelo é capaz de localizar grandes e pequenos objetos nas imagens (FANG; WANG, 2021).

Nas imagens utilizadas para realizar o treinamento do algoritmo YOLO, é necessário delimitar e classificar todas as classes que se deseja identificar. Após estas delimitação e classificação das classes nas imagens, é gerado um arquivo de texto para cada imagem, contendo uma linha para cada classe (objeto) delimitada na imagem, com 5 valores em cada linha (classe, x_center, y_center, w e h), sendo o primeiro valor a classe numérica da respectiva classe (Valor no YOLO) e os 4 valores seguintes representam as coordenadas da localização da classe na imagem. A Figura 12 representa uma imagem com os objetos gato (0) e cachorro (1)

delimitados, enquanto a Tabela 1 representa os valores de cada linha do arquivo “figura_12.txt”.

Figura 12 – Delimitação de Gato e Cachorro

Fonte: Do Autor.

Tabela 1 – Coordenadas dos objetos delimitados da figura 18

Classe	x_center	y_center	w	h
0	0,15	0,75375	0,18	04675
1	0,62	0,4975	0,76	0,99

Fonte: Do Autor.

Os valores que correspondem à localização de cada objeto derivam das equações presentas no Quadro 1.

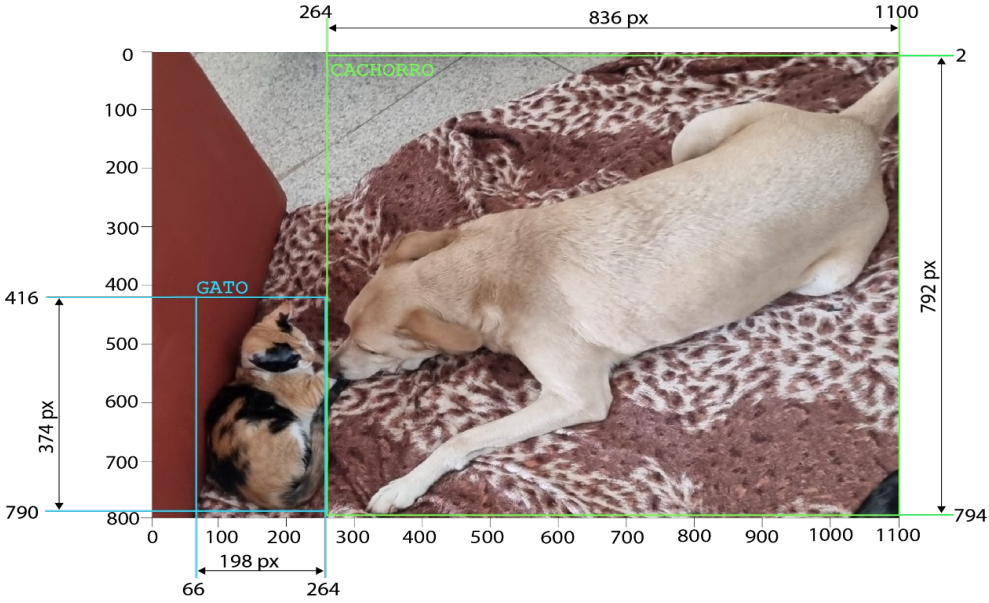
Quadro 1 – Equações da localização, interpretada pelo YOLO, das classes

x_max_recorte = Largura máxima do objeto
x_min_recorte = Largura mínima do objeto
y_max_recorte = Altura máxima do objeto
y_min_recorte = Altura mínima do objeto
x_center = ((x_max_recorte + x_min_recorte) / 2) / largura da imagem
y_center = ((y_max_recorte + y_min_recorte) / 2) altura da imagem
w = largura do recorte / largura da imagem
h = altura do recorte / altura da imagem

Fonte: Do Autor.

Considerando que a Figura 12 possua 1100 pixels de largura e 800 pixels de altura, a Figura 13 representa as coordenadas geográficas e dimensões das classes gato (0) e cachorro (1). Devido ao fato de o YOLO começar a processar as imagens pela parte superior esquerda, os pontos y = 0 e x = 0 localizam-se também na parte superior esquerda da imagem.

Figura 13 – Coordenadas geográficas dos objetos da figura 12



Fonte: Do Autor.

Então, para a classe “gato”, com os valores x_max_recorte = 264, x_min_recorte = 66, y_max_recorte = 790, y_min_recorte = 416, largura_recorte = 198 e altura_recorte = 374, os valores do YOLO x_center, y_center, w e h são calculados de acordo com o Quadro 2.

Quadro 2 – Cálculo das coordenadas de gato na figura 13

x_center = ((264 + 66) / 2) / 1100 = 0,15
y_center = ((790 + 416) / 2) / 800 = 0,75375
w = 198 / 1100 = 0,18
h = 374 / 800 = 0,4675

Fonte: Do Autor.

E para a classe “cachorro”, com os valores x_max_recorte = 1100, x_min_recorte = 264, y_max_recorte = 794, y_min_recorte = 2, largura_recorte = 836 e altura_recorte = 792, os valores do YOLO x_center, y_center, w e h são calculados de acordo com o Quadro 3.

Quadro 3 – Cálculo das coordenadas de cachorro na Figura 13

x_center = ((1100 + 264) / 2) /1100 = 0,62
y_center = ((792 + 2) /2) /800 = 0,4975
w = 836 / 1100 = 0,76
h = 792 / 800 = 0,99

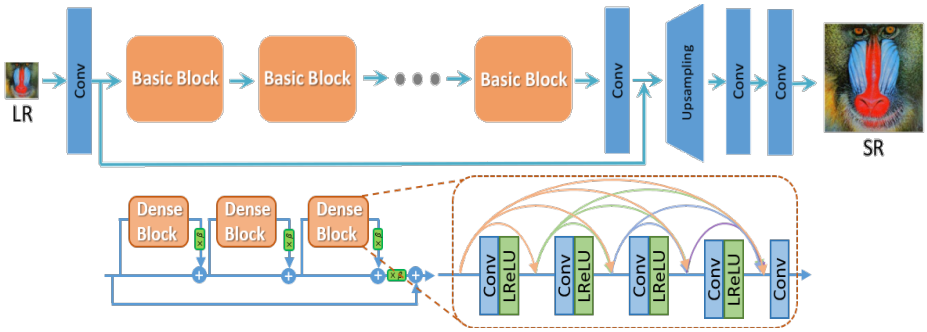
Fonte: Do Autor.

2.6 Esrgan

ESRGAN (acrônimo de “Enhanced Super Resolution Generative Adversarial Network”) é uma rede geradora (GAN) de super resolução de imagens de baixa qualidade, podendo aumentar a resolução de imagens até 4 vezes por execução (WANG et al., 2018), conforme ilustrado na Figura 14. As redes GAN são capazes de gerar dados falsos realistas e sua arquitetura é formada por duas redes principais, as redes geradoras e as discriminadoras. A rede geradora tenta gerar dados falsos enquanto a rede discriminadora tenta distinguir dados reais e falsos, assim

aumentando a capacidade da rede geradora criar dados mais realistas. Por ser uma rede GAN, a rede ESRGAN utiliza um discriminador para tentar prever se os dados de entrada são mais realistas que os dados de saída (dados gerados) (WANG et al., 2018).

Figura 14 – Arquitetura da rede ESRGAN

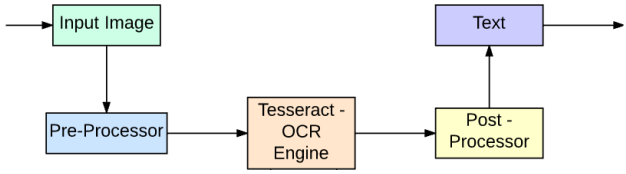


Fonte: arXiv:1809.00219v2 [cs.CV] 17 Sep 2018

2.7 Tesseracr OCR

O Tesseract OCR é um mecanismo de reconhecimento óptico de caracteres desenvolvido em 1980 pela HP (Hewlett-Packards) e mantido pela Google desde 2006. Este mecanismo é capaz de reconhecer todos os caracteres UTF-8 com elevada precisão e, em sua última versão, foi implementado um algoritmo de LSTM para identificar textos e linhas. O LSTM (Longo Short Term Memmory) (GERS; SCHMIDHUBER, 2001) é um algoritmo de rede neural recorrente com excelente capacidade de detecção de sequências, o que faz os textos serem extraídos das linhas como um todo.

Figura 15 – Fluxo de processo do Tesseracr_OCR



Fonte: https://medium.com/@balaajip/optical-character-recognition-99aba2dad314

2.8 Python

Python é uma das linguagens de programação mais utilizadas no mundo devido à sua simplicidade, alta capacidade e desempenho (SEVERANCE, 2015). Além de ser capaz de realizar operações de forma mais simples que a maioria das linguagens de programação, as suas bibliotecas possibilitam abstrair diversas operações, o que acelera e simplifica o processo de escrita de código.

As bibliotecas no Python são coleções de código que executam tarefas específicas. Com o uso do Python, diversas operações podem ser facilmente realizadas sem a necessidade de desenvolvê-las. Na ciência de dados, as bibliotecas do Python são extremamente importantes, facilitando diversos processos como o tratamento de dados, a criação de modelos de ML e a visualização de dados.

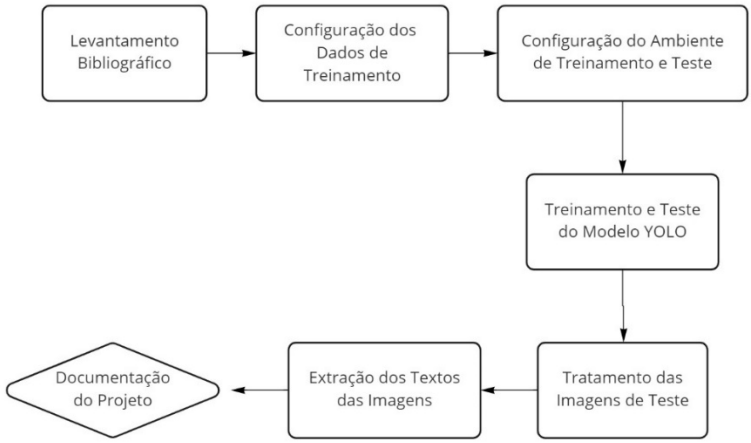
2.9 Conda

O conda é um gerenciador de pacotes do Python utilizado para baixar tanto o Python como as suas bibliotecas. Também é utilizado para criar ambientes virtuais onde cada ambiente possuí uma versão específica do Python e das bibliotecas, fazendo assim, que cada aplicação que utilize o Python possua um ambiente virtual específico, evitando erros de compatibilidade entre versões do Python e suas bibliotecas.

3 METODOLOGIA

O estudo realizado é quali-quantitativo aplicado a dados reais. É qualitativo tendo em vista a origem dos dados e a forma como são tratados e interpretados. Em razão da natureza do objetivo do estudo e das metodologias aplicadas, o presente estudo também é quantitativo.

Figura 16 – Fluxo do trabalho



Fonte: Do Autor.

A Primeira Etapa foi composta por pesquisas de literaturas científicas com explicações sólidas sobre detecção de objetos com machine learning e visão computacional, além de buscas sobre métodos que elevam a acurácia em extração de textos de imagens.

A Segunda Etapa consistiu na configuração dos dados de treinamento no formato do YOLO, com a realização da delimitação, nas imagens de treinamento, dos campos que o modelo será treinado para identificar na CNH.

A Terceira Etapa abordou sobre a configuração física e virtual do ambiente em que o modelo foi aplicado e sobre a criação e configuração de arquivos necessários para a execução do treinamento e do teste.

A Quarta Etapa consistiu na execução dos procedimentos para treinar o modelo e o aplicar para detectar os campos das CNHs da base de teste.

Na Quinta Etapa, foram realizados os recortes dos campos identificados pelo modelo nas CNHs de treinamento e aplicado o algoritmo ESRGAN para aumentar a qualidade dos recortes.

Na Sexta Etapa, foram realizadas as extrações dos textos dos recortes (gerados na etapa anterior) com o algoritmo tesseract OCR e aplicadas expressões regulares para remover possíveis caracteres erroneamente extraídos.

3.1 Desenvolvimento do projeto

3.1.1 Configuração dos dados de treinamento

A base de dados utilizada neste artigo é constituída de 195 fotos de CNH com a descrição textual de 10 campos de cada CNH. Para treinar o modelo YOLO, foram selecionadas 12 imagens nas quais, para adequá-las ao modelo de treinamento, foram delimitados e classificados todos os campos desejados para identificação. Para isto, foi utilizada a biblioteca labelImade, do Python, que fornece uma interface visual para delimitar e classificar regiões de imagens.

No total da base de treinamento, foram delimitados 16 campos em cada uma das 12 imagens de CNH. A Figura 17 representa os campos delimitados em uma CNH fictícia enquanto a Tabela 2 representa o nome dos campos delimitados, o valor interpretado pelo YOLO e os respectivos significados.

Figura 17 – CNH Fictícia com delimitação dos campos a serem identificados pelo modelo

Fonte: Do Autor.

Tabela 2 – Campos delimitados na CNH

Campo	Valor no YOLO	Significado
documento	0	Delimitação do documento na imagem
nome	1	NOME
rg	2	DOC. IDENTIDADE / ÓRG. EMISSOR / UF
cpf	3	CPF
dt _nas	4	DATA NASCIMENTO
filia	5	FILIAÇÃO
perm	6	PERMISSÃO
acc	7	ACC
cat	8	CATEGORIA
reg	9	Nº REGISTRO
val	10	VALIDADE
1_hab	11	1ª HABILITAÇÃO
obs	12	OBSERVAÇÕES
local	13	LOCAL
dt_emi	14	DATA EMISSÃO
num_lat	15	Número lateral ao lado da foto do condutor

Fonte: Do Autor.

Após a definição e a delimitação dos campos nas imagens, na mesma pasta em que estão presentes as imagens, foi gerado o arquivo “classes.names” contendo o conteúdo da coluna “Campo” da Tabela 1 e um arquivo de texto “nome_da_imagem.txt” para cada imagem. Este arquivo de texto possui uma linha para cada classe (objeto) delimitada na imagem, com 5 valores em cada linha (classe, x_center, y_center, w e h), conforme Tabela 1.

3.1.2 Ambiente de treinamento e teste

Antes de executar o treinamento, foi necessário configurar o ambiente local para ser capaz de executar o algoritmo da rede neural YOLO. O ambiente utilizado para realizar o treinamento da rede possuí o sistema operacional, Zorin OS Pro 16

(Baseado no ubuntu 20.04), processador (CPU) i7-9750H e placa gráfica (GPU) Nvidia RTX 2070.

Idealmente o treinamento de uma rede neural YOLO deve ser executado em GPU devido ao fato de GPU ter capacidade de processamento muito mais elevada que a capacidade da CPU, fazendo que o processamento de redes neurais seja cerca de 100 vezes mais rápido quando executado em GPU (OH; JUNG, 2004). Para executar os cálculos de treinamento da rede neural YOLO na GPU, foi necessário instalar o NVIDIA® CUDA® Toolkit, que é um compilado de bibliotecas que permitem executar processamento paralelo na GPU. Após instalar o Nvidia CUDA na versão 11.06, foi instalado o cuDNN (Acrônimo de CUDA Deep Neural Network) versão 8.4.0, que é a biblioteca CUDA com implementações altamente ajustadas para rotinas de redes neurais profundas, como convoluções para frente e para trás, agrupamento (pooling) e cálculo das camadas de ativação.

O ambiente de treinamento possui o Conda 4.10.3 instalado, que é um sistema de gerenciamento de pacotes crucial para configurar o ambiente YOLO. Com o Conda ativado no terminal na pasta /home/meu_nome/yolo, foram executados os seguintes comandos:

“conda create --name yolo python=3.7.13” para criar um ambiente virtual de nome yolo e com o python 3.7.13;

“conda activate yolo” para ativar o ambiente virtual yolo;

“pip install opencv-python” para instalar a biblioteca python de visão computacional;

“conda install git” para instalar o sistema gerenciador de códigos, git;

“git clone https://github.com/AlexeyAB/darknet.git” para clonar a pasta darknet, presente no link;

“cd darknet” para acessar a pasta darknet.

Após o download da pasta “darknet”, foi definido que o processamento utilizaria a GPU, o gerenciamento de processamento seria realizado pela biblioteca cuDNN e o openCV seria o responsável por interpretar o conteúdo das imagens.

Para isto, no arquivo “Makefile”, presente dentro da pasta “darknet”, o valor da linha 1 foi alterado de “GPU=0” para “GPU=1”, na linha 2 “CUDNN=0” para “CUDNN=1” e na linha 4 “OPENCV=0” para “OPENCV=1”. Após definir como o processamento será realizado, foram executados os seguintes comandos no terminal:

“make” para compilar o framework darknet;

“./darknet” para verificar se a instalação ocorreu corretamente, neste caso, é retornado “usage: ./darknet”.

O Darknet é uma biblioteca de ML de código aberto que fornece a estrutura de rede neural capaz de executar o processamento em GPU com a biblioteca cuDNN.

Antes de executar o treinamento e teste do modelo, foram criados os arquivos de textos “train.txt” e “test.txt”, respectivamente contendo a rota de todas as imagens de treinamento e a rota de todas as imagens de teste. O arquivo “train.txt” possui o conteúdo semelhante ao quadro 4 e o arquivo “test.txt” com o quadro 5.

Quadro 4 – Representação dos valores no arquivo “train.tst”

```
/home/meu_nome/yolo/dados/train/imagem_01.png
/home/meu_nome/yolo/dados/train/imagem_02.png
/home/meu_nome/yolo/dados/train/imagem_03.png
```

Fonte: Do Autor.

Quadro 5 – Representação dos valores no arquivo “test.tst”

```
/home/meu_nome/yolo/dados/test/imagem_04.png
/home/meu_nome/yolo/dados/test/imagem_05.png
/home/meu_nome/yolo/dados/test/imagem_06.png
```

Fonte: Do Autor.

Dentro da pasta “cfg”, localizada dentro da pasta “darknet”, foi criado o arquivo “labelled_data.data” contendo o número de classes a serem identificadas, a rota para o arquivo “train.txt”, a rota para o arquivo “test.txt”, a rota para o arquivo “classes.names” (arquivo criado no processo de marcação das classes nas imagens de treinamento) e o local onde serão salvos os pesos da rede (backup), sendo o conteúdo deste arquivo semelhante com o Quadro 6.

Quadro 6 – Representação dos valores no arquivo “labelled_data.data”

```
classes = 16
train = /home/meu_nome/yolo/train.txt
valid = /home/meu_nome/yolo/test.txt
names = /home/meu_nome/yolo/dados/train/classes.names
backup = backup
```

Fonte: Do Autor.

Ainda como pré-requisito para executar o treinamento do modelo, foi criado o arquivo de configuração do modelo de treinamento. Para isto, na pasta “cfg” (home/meu_nome/yolo/darknet/cfg), foi criado o arquivo de configuração do modelo “yolov3_labelled_train.cfg”, que é uma cópia do arquivo “yolov3.cfg” (arquivo de configuração padrão da rede neural YOLOv3). Entretanto, para o modelo se adequar à base de dados, os valores batch, subdivisions, max_batches, steps, filters e classes foram editados, da seguinte forma:

Batch é a quantidade de imagens que será carregada por iteração. Devido ao fato de a quantidade de imagens de treinamento ser inferior à quantidade de batch, as imagens são carregadas repetidamente até atingir a quantidade de batch. Além de ser a quantidade de imagens a serem carregadas por iteração, o YOLOv3 realiza uma normalização por batch, o que eleva a precisão média em até 2%, entretanto, quanto maior o batch, maior o tempo de processamento. Para o treinamento do modelo, o batch é 32;

Subdivisions divide a quantidade de imagens no batch a serem processadas simultaneamente na GPU. Com subdivisions setado para 16 no modelo, o batch (32

imagens) é dividido por 16, resultando em processamento simultâneo na GPU de 2 imagens;

Max_batches é a quantidade máxima de iterações de treinamento, finalizando automaticamente o treinamento após atingir esta quantidade de iterações. Por padrão este valor é definido pela quantidade de classes multiplicado por 2000, sendo:

max_batches = 16 * 2000 = 32000;

Steps corresponde às quantidades de interações executadas em que a taxa de aprendizado será multiplicada pelo valor de escala. Os valores de steps são definidos por 80% e 90% de max_batches, resultando em steps = 25600 e 28800. Considerando que os valores de escala são 0.1 e 0.1, após a iteração 25600 a taxa de aprendizado é multiplicada por 0.1 e, após a iteração 28800, a taxa de aprendizado será multiplicada novamente por 0.1;

Os filters alterados são os filters das camadas convolucionais antes das camadas yolo. Por padrão, o valor de filters nestas camadas é dado pelo número de classes acrescido da quantidade de coordenadas (4 coordenadas, sendo x_center, y_center, width e height) mais 1(este 1 representa a porcentagem de certeza que a rede neural tem de a classe predita estar correta (confidence)):

filters = (16 + 4 + 1) *3 = 63;

Classes é a quantidade de classes que as camadas YOLO irá detectar na imagem, sendo 16 a quantidade de classes.

A Tabela 3 representa as linhas que foram alteradas, com os valores antigos e os novos.

Tabela 3 – Alterações do arquivo yolov3_labelled_train.cfg

Linha	Valor Antigo	Valor Novo
3	batch = 1	batch = 32
4	subdivisions = 1	subdivisions = 16
20	max_batches = 500200	max_batches = 32000
22	steps = 400000,450000	steps = 25600,28800
603	filters = 255	filters = 63
610	classes = 80	classes = 16
689	filters = 255	filters = 63
696	classes = 80	classes = 16
776	filters = 255	filters = 63
783	classes = 80	classes = 16

Fonte: Do Autor.

Assim como no treinamento, também foi necessário criar o arquivo de configuração para a realização dos testes. Também dentro da pasta “cfg”, foi criado o arquivo “yolov3_labelled_test.cfg” com os dados de “yolov3.cfg” e alterados os valores das linhas de acordo com a Tabela 4.

Tabela 4 – Alterações do arquivo yolov3_labelled_test.cfg

Linha	Valor Antigo	Valor Novo
20	max_batches = 500200	max_batches = 32000
22	steps = 400000,450000	steps = 25600,28800
603	filters = 255	filters = 63
610	classes = 80	classes = 16
689	filters = 255	filters = 63
696	classes = 80	classes = 16
776	filters = 255	filters = 63
783	classes = 80	classes = 16

Fonte: Do Autor.

Diferente do arquivo de configuração de treinamento, no arquivo de configuração de teste, batch e subdivisions permanecem com valor igual a 1, sendo

executada somente uma imagem por iteração, pois é necessário que o modelo treinado informe a rota dos campos categorizados de cada imagem separadamente.

3.2 Treinamento e teste do modelo

Para realizar o treinamento do modelo YOLOv3 para detecção dos campos da CNH, com o terminal em home/meu_nome/yolo/darknet, foi executado o seguinte comando:

"./darknet detector train cfg/labelled_data.data cfg/yolov3_labelled_train.cfg weights/darknet53.conv.74 -dont_show"

Neste comando, "cfg/labelled_data.data" é o arquivo com os dados das imagens, "cfg/yolov3_labelled_train.cfg" é o arquivo com a configuração do modelo de treinamento, "weights/darknet53.conv.74" é o peso que o modelo irá utilizar para começar o treinamento e "-dont_show" é usado para não exibir visualmente a identificação dos campos nas imagens.

Cada iteração demorou cerca de 16 segundos, o que resultou em 145 horas de treinamento. Durante o processo de treinamento, foi gerado, dentro da pasta "backup", o arquivo com os pesos do modelo "yolov3_labelled_train_last. weights", que teve os valores atualizados a cada 100 iterações, evitando que todo o processo de treinamento ocorrido antes da finalização do treinamento fosse perdido caso ocorresse alguma instabilidade durante o processo.

Para utilizar o modelo treinado para detectar os campos das imagens das CNHs utilizadas para teste, também com o terminal em home/meu_nome/yolo/darknet, foi executado o comando:

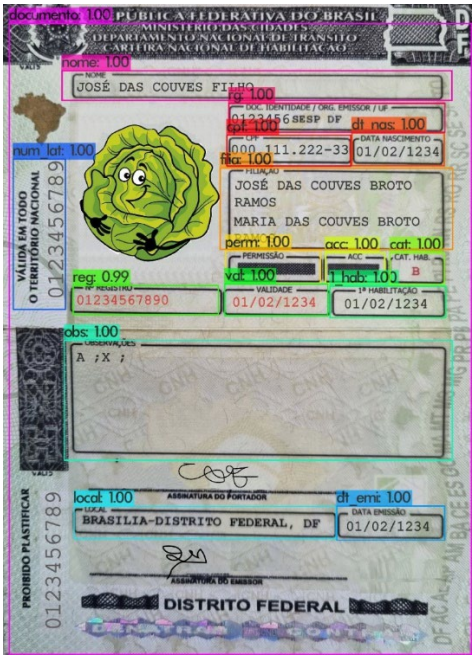
"./darknet detector test cfg/labelled_data.data cfg/yolov3_labelled_test.cfg backup/yolov3_labelled_train_last.weights -thresh 0.8 -dont_show -out /home/meu_nome/yolo/result.json < /home/arthur/Documentos/yolo/test.txt"

Neste comando, foram passados os parâmetros "backup/yolov3_labelled_train_last.weights" informando qual peso seria utilizado para realizar o teste, "-thresh 0.8" foi usado para não salvar as classificações com menos de 80% de confiança, "-dont_show" serve para não exibir visualmente a

categorização das imagens e “-out /home/meu_nome/yolo/result.json” foi usado para salvar os resultados no arquivo “result.json” dentro da pasta “yolo”.

Para demostrar a capacidade do modelo identificar os 16 campos em uma imagem de CNH, a imagem da CNH fictícia da Figura 17 foi submetida ao modelo treinado, sendo a Figura 18 o resultado gerado pelo modelo e a Tabela 5 os respectivos valores de localização dos campos (classes) identificados.

Figura 18 – Identificação, pelo modelo treinado, dos campos da CNH fictícia



Fonte: Do Autor.

Tabela 5 – Coordenadas obtidas das classes da CNH Fictícia

Classe	x_center	y_center	width	height
0	0.500896	0.500322	0.998207	0.999355
1	0.543254	0.125766	0.796056	0.045792
2	0.709547	0.174299	0.460780	0.046759
3	0.611609	0.222993	0.264904	0.047404
4	0.842896	0.224121	0.196773	0.049016
5	0.710444	0.313931	0.465262	0.131248
6	0.580681	0.403741	0.203048	0.049016
7	0.751457	0.406159	0.142985	0.046114
8	0.882116	0.405514	0.121918	0.049339
9	0.305693	0.453402	0.325415	0.049016
10	0.583819	0.456143	0.231735	0.049984
11	0.821381	0.457272	0.245182	0.048371
12	0.541909	0.605127	0.803227	0.186069
13	0.429180	0.793937	0.566114	0.052241
14	0.827656	0.797485	0.229045	0.049661
15	0.079561	0.353596	0.109816	0.226056

Fonte: Do Autor.

3.3 Tratamento de imagens

Embora seja realizada a classificação de 16 classes, a base de dados com o texto presente nas imagens das CNHs só possuí os textos de 10 campos. Levando isto em consideração, foram mantidos somente os registros de identificações em que as classes (campos) estão presentes na base de dados, valores estes representados pela Tabela 6.

Tabela 6 – Campos com valores presentes na base de dados

Campo	Valor no YOLO	Significado
nome	1	NOME
rg	2	DOC. IDENTIDADE / ÓRG. EMISSOR / UF
cpf	3	CPF
dt _nas	4	DATA NASCIMENTO
filia	5	FILIAÇÃO
reg	9	Nº REGISTRO
val	10	VALIDADE
1_hab	11	1ª HABILITAÇÃO
local	13	LOCAL
dt_emi	14	DATA EMISSÃO

Fonte: Do Autor.

O YOLO gera coordenadas relativas (x_center, y_center, w e h) dos campos detectados. Entretanto, para utilizar o openCV para recortar os campos detectados, são necessárias as coordenadas geográficas da imagem (x_max, x_min, y_max e y-min) conforme Figura 13. Para obter estas coordenadas, foram gerados e utilizados os cálculos presentes no Quadro 7.

Quadro 7 – Cálculo para recortar os campos identificados na imagem de CNH.

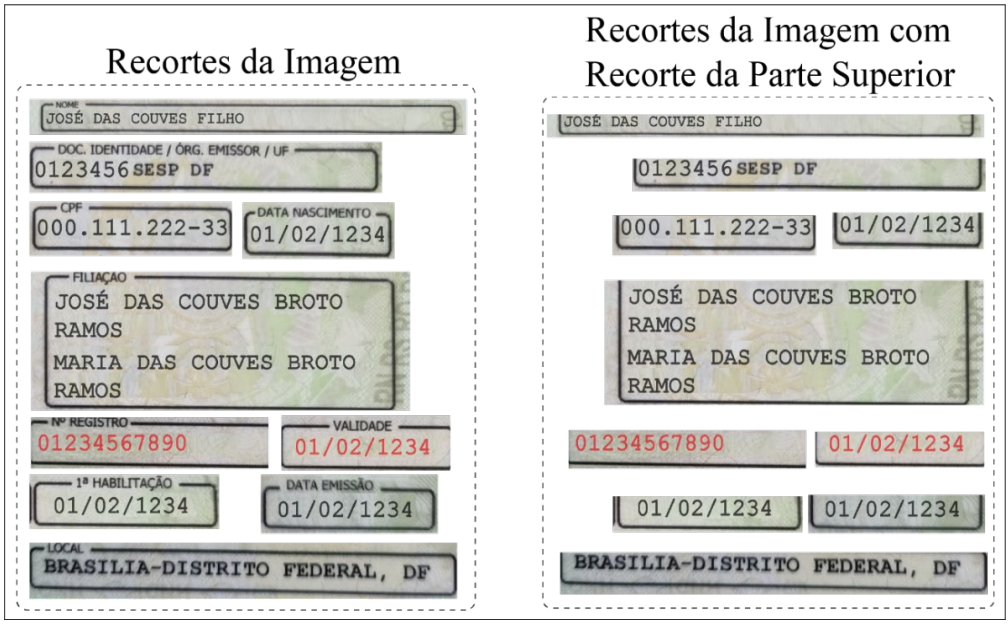
```
width = largura da imagem
height = altura da imagem
width_crop = w * width          # Largura do recorte
height_crop = h * height        # Altura do recorte
x_max = (width_crop + (x_center * width * 2))/2
x_min = (x_center * width * 2) - x_max
y_max = (height_crop + (y_center * height * 2))/2
y_min = (y_center * height * 2) - y_max
```

Fonte: Do Autor.

Após definir as coordenadas geográficas, foram realizados os recortes de forma que cada recorte possui o nome da classe pertencente acrescido do

identificador da CNH pertencente (ex: “cpf_cnh_ficticia.png”). Como pode ser observado ao lado esquerdo da Figura 19, os recortes das classes englobam os nomes dos campos, o que pode gerar uma extração de textos desnecessários. Visando solucionar este problema, nos recortes que não são da classe filiação, foi realizado um recorte de 35% da parte superior do recorte e, nos recortes da classe filiação, um recorte de 10% da parte superior. A Figura 19 realiza a comparação entre todos os recortes da imagem da CNH fictícia, com os mesmos recortes com a remoção da parte superior.

Figura 19 – Recortes da CNH fictícia com e sem remoção da parte superior

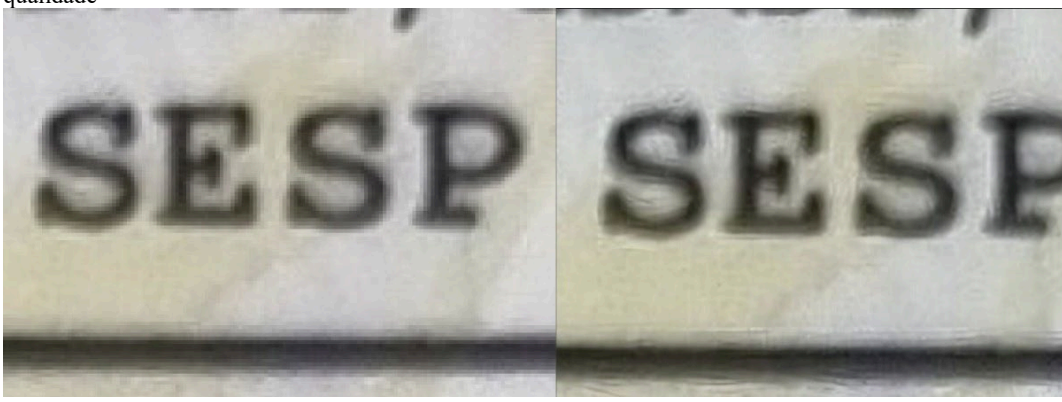


Fonte: Do Autor.

Todos os recortes foram submetidos ao modelo ESRGAN para aumentar a qualidade e em seguida salvos com o mesmo nome em uma pasta separada. Este processo é muito útil quando a imagem da CNH submetida possuí uma baixa qualidade, visto que, embora o YOLO consiga identificar bem os campos em imagens com baixa qualidade, os recortes ficam com qualidade extremamente ruins, dificultando a extração do texto presente neles. A Figura 20 exibe, da CNH fictícia, o “ÓRG. EMISSOR” DE “DOC. IDENTIDADE / ÓRG. EMISSOR / UF”, ao lado

esquerdo o recorte normal e ao lado direito o recorte que passou pelo processo de aumentar a qualidade.

Figura 20 – Texto “SESP” antes e depois de ser realizado o ganho de qualidade



Fonte: Do Autor.

3.4 Extração dos textos

Com todos os recortes criados, foram extraídos os textos deles com a biblioteca pytesseract e inseridos os textos extraídos em um dataframe com o identificador da imagem, classe a qual o texto extraído pertence, texto extraído dos recortes que passaram pelo processo de qualidade, texto extraído dos recortes normais (que não passaram pelo processo de qualidade) e o texto real que representa o real valor dos campos. Com todos os textos extraídos, os nomes das classes foram substituídos por valores que melhor representam o texto, sendo a Tabela 7 a representação dos valores antigos e os novos. A Tabela 8 representa os textos extraídos dos recortes da CNH fictícia e o texto real dos campos.

Tabela 7 – Representação dos valores substituídos na tabela com as extrações do texto

Valor Antigo	Valor Novo
nome	nome_CNH
rg	identidade_CNH
cpf	cpf_CNH
dt_nas	nascimento_CNH
filia	filiacao_CNH
reg	registro_CNH
val	validade_CNH
1_hab	pri_habilitacao_CNH
local	local_emissao_CNH
dt_emi	data_emissao_CNH

Fonte: Do Autor.

Tabela 8 – Textos extraídos da CNH fictícia em comparação com o texto real

classe	texto_qualidade	texto_normal	texto_real
data_emissao_CNH	\n\n01/02/1234,\n\x0c	\n\n01/02/1234\n\x0c	01/02/1234
local_emissao_CNH	\n\nBRASILIA-DISTRITO FEDERAL, DF\n\n \n\x0c	| BRASILIA~DISTRITO FEDERAL, DF |\n\x0c	BRASILIA-DISTRITO FEDERAL, DF
pri_habilitacao_CNH	| 01/02/1234 |\n\x0c	\\ 01/02/1234\n\x0c	01/02/1234
validade_CNH	| 01/02/1234 |\n\x0c	| 01/02/1234 |\n\x0c	01/02/1234
registro_CNH	01234567890\n\nseen\n\x0c	01234567890\n\nee\n\x0c	1234567890
filiacao_CNH	â€˜JOSE DAS COUVES BROTO\nRAMOS\nMARIA DAS COUVES BROTO\nRAMOS\n\x0c	JOSE DAS COUVES BROTO\nRAMOS\nMARIA DAS COUVES BROTO\n\n \n\nRAMOS\n\x0c	JOSE DAS COUVES BROTO RAMOS MARIA DAS COUVES BROTO

nascimento_CNH	â€”a ww EE re ee\n\n01/02/1234\n\n \n\x0c	\n\x0c	01/02/1234
cpf_CNH	000 0. . 222-33)\n\x0c	000 ai 1 - 222-33}\n\x0c	11122233
identidade_CNH	0123456SESP DF\n\n \n\x0c	\n\n0123456SESP DF\n\n \n\x0c	0123456 SESP DF
nome_CNH	JOSE DAS COUVES FILHO\n\n \n\x0c	JOSE DAS COUVES FILHO |\n\x0c	JOSE DAS COUVES FILHO

Fonte: Do Autor.

3.5 Tratamento dos textos extraídos

Pode-se observar na Tabela 8 que os textos extraídos possuem inúmeros caracteres que representam alguma formatação de texto ou foram extraídos erroneamente. Para solucionar este problema, foram realizados 8 filtros com expressões regulares nos textos extraídos, com as seguintes finalidades:

Remover todos os caracteres que não correspondem com padrão de data (\d+/\d+/\d+) nos textos extraídos pertencentes às classes de data ('data_emissao_CNH', 'nascimento_CNH', 'pri_habilitacao_CNH', 'validade_CNH');

Remover todos os caracteres não numéricos dos textos extraídos pertencentes às classes “registro_CNH” e “cpf_CNH”;

Substituir por “ ” (espaço) os caracteres “,-” presentes nos textos extraídos pertencentes à classe “filiação_CNH”;

Substituir por “ ” (espaço) os caracteres “|\n{}()'"‘#.:?@$*&” em todos os textos extraídos;

Remover todas as letras minúsculas;

Remover todos os códigos ASCII “\x00-\x08\x0b\x0c\x0e-\x1f\x7f-\xff”;

Substituir onde existe mais de um espaço consecutivo por apenas um espaço;

Remover espaço “ ” no início e final dos textos extraídos.

A Tabela 9 representa os valores extraídos da CNH fictícia após a realização dos filtros nos textos extraídos e os textos reais.

Tabela 9 – Textos tratados extraídos da CNH fictícia

classe	texto_qualidade	texto_normal	texto_real
data_emissao_CNH	01/02/1234	01/02/1234	01/02/1234
local_emissao_CNH	BRASILIA-DISTRITO FEDERAL, DF	BRASILIA~DISTRITO FEDERAL, DF	BRASILIA-DISTRITO FEDERAL, DF
pri_habilitacao_CNH	01/02/1234	01/02/1234	01/02/1234
validade_CNH	01/02/1234	01/02/1234	01/02/1234
registro_CNH	01234567890	01234567890	01234567890
filiacao_CNH	JOSE DAS COUVES BROTO RAMOS MARIA DAS COUVES BROTO RAMOS	JOSE DAS COUVES BROTO RAMOS MARIA DAS COUVES BROTO RAMOS	JOSE DAS COUVES BROTO RAMOS MARIA DAS COUVES BROTO RAMOS
nascimento_CNH	01/02/1234		01/02/1234
cpf_CNH	000022233	000122233	00011122233
identidade_CNH	0123456SESP DF	0123456SESP DF	0123456 SESP DF
nome_CNH	JOSE DAS COUVES FILHO	JOSE DAS COUVES FILHO	JOSE DAS COUVES FILHO

Fonte: Do Autor.

4 RESULTADOS OBTIDOS

Com os textos extraídos tratados, foi calculada a porcentagem de similaridade entre os textos extraídos das imagens que passaram pelo processo de ganho de qualidade e o texto real (Porcentagem Qualidade) e a porcentagem de similaridade entre os textos extraídos das imagens normais e o texto real (Porcentagem Normal). Para isto, foi utilizada biblioteca SequenceMatcher, do Python, que calcula a porcentagem de similaridade entre dois textos, verificando os caracteres presentes e a ordem deles nos 2 textos. Conforme exemplificado na Tabela 10, todos os campos da CNH fictícia estão com a porcentagem de similaridade dos textos extraídos com os textos reais.

Tabela 10 – Porcentagem de similaridade dos textos extraídos da CNH fictícia

classe	Porcentagem Qualidade	Porcentagem Normal
Data da Emissão	100.0 %	100.0 %
Local da Emissão	100.0 %	96.55 %
Data Primeira Habilitação	100.0 %	100.0 %
Validade	100.0 %	100.0 %
Registro	100.0 %	100.0 %
Filiação	100.0 %	100.0 %
Data Nascimento	100.0 %	0.0 %
CPF	80.0 %	90.0 %
Documento de Identidade	96.55 %	96.55 %
Nome	100.0 %	100.0 %
Total	97.65 %	88.31 %

Fonte: Do Autor.

A tabela de porcentagem de similaridade gerada possui uma linha para cada campo de cada CNH, resultando em um total de 1830 linhas. Para conseguir representar estes valores, foi realizada a média, por classe e total, das porcentagens de similaridade obtidas, conforme a Tabela 11.

Tabela 11 – Porcentagem de similaridade dos textos extraídos de 183 imagens de CNH

Classe	Similaridade Qualidade	Similaridade Normal
Data da Emissão	95.63 %	93.95 %
Local da Emissão	77.82 %	55.04 %
Data Primeira Habilitação	96.72 %	94.09 %
Validade	95.74 %	91.90 %
Registro	96.92 %	96.09 %
Filiação	96.48 %	92.73 %
Data Nascimento	94.88 %	95.43 %
CPF	96.80 %	52.06 %
Documento de Identidade	92.84 %	83.00 %
Nome	93.25 %	82.37 %
Total	93.71 %	83.67 %

Fonte: Do Autor.

Foi verificada a quantidade de textos extraídos com 100% de similaridade com o texto real, conforme Tabela 12. Nos textos extraídos das imagens que passaram pelo processo de aumentar a qualidade, 70.6% (1292 de 1830 campos) dos textos extraídos possuem 100% de similaridade com o texto real, enquanto nos textos extraídos das imagens que não passaram pelo processo de qualidade (Imagens Normais), 64.86% (1187 de 1830 campos) dos textos extraídos possuem 100% de similaridade. Ainda analisando os campos com 100% de similaridade, 2.18% (4 imagens) das imagens de CNH que passaram pelo processo de aumentar a qualidade possuem 100% de similaridade em todos os campos e nenhuma das imagens que não passou pelo processo de qualidade possui 100% de similaridade em todos os campos.

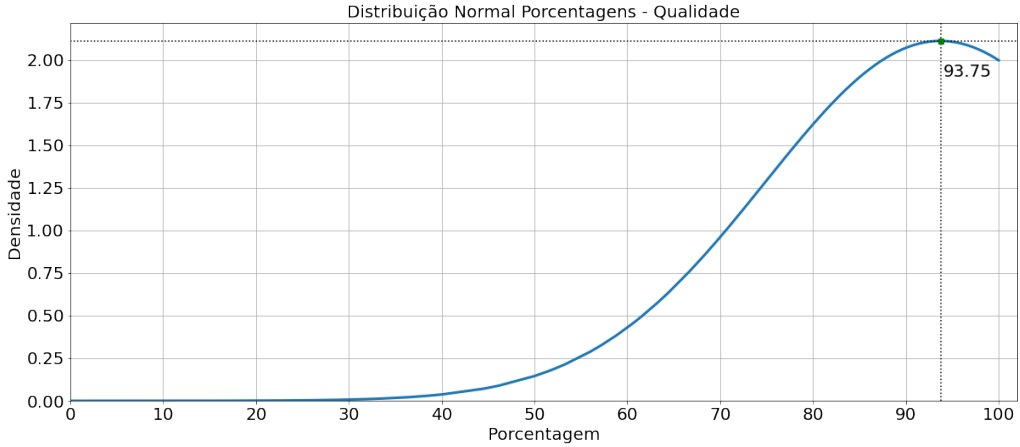
Tabela 12 – Quantidade de textos extraídos com 100% de similaridade com o texto real (Máximo de 183 por classe)

Classe	Qualidade	Normal
Data da Emissão	166	155
Local da Emissão	78	54
Data Primeira Habilitação	171	158
Validade	166	151
Registro	163	166
Filiação	91	110
Data Nascimento	136	148
CPF	137	74
Documento de Identidade	86	75
Nome	98	96
Total	1292	1187

Fonte: Do Autor.

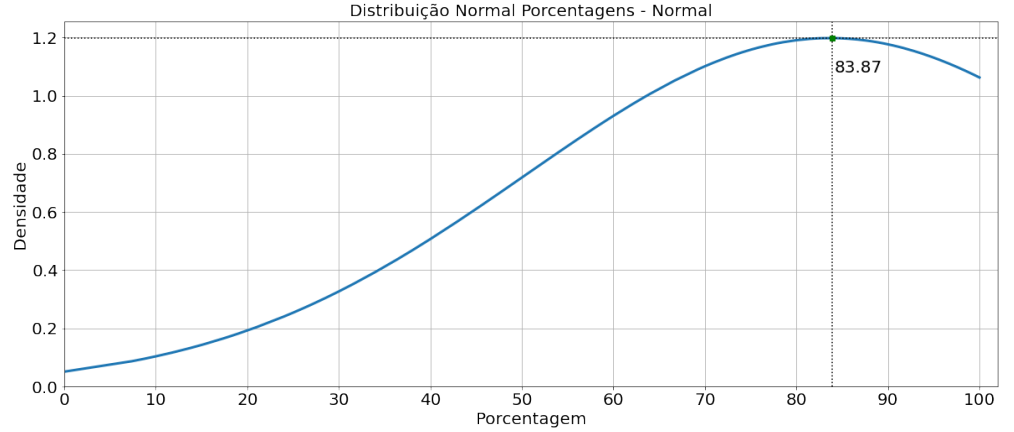
Para entender melhor os valores obtidos, tanto para as porcentagens de similaridade dos textos extraídos de imagens que passaram pelo processo de qualidade como para as porcentagens dos textos extraídos das imagens normais, foram gerados os gráficos de distribuição normal de todas as porcentagens de similaridade dos textos extraídos (Figuras 21 e 22) e destas porcentagens por classe (Figuras 23 e 24).

Figura 21: Curva normal das porcentagens de similaridade dos textos extraídos de imagens que passaram pelo processo de aumentar a qualidade



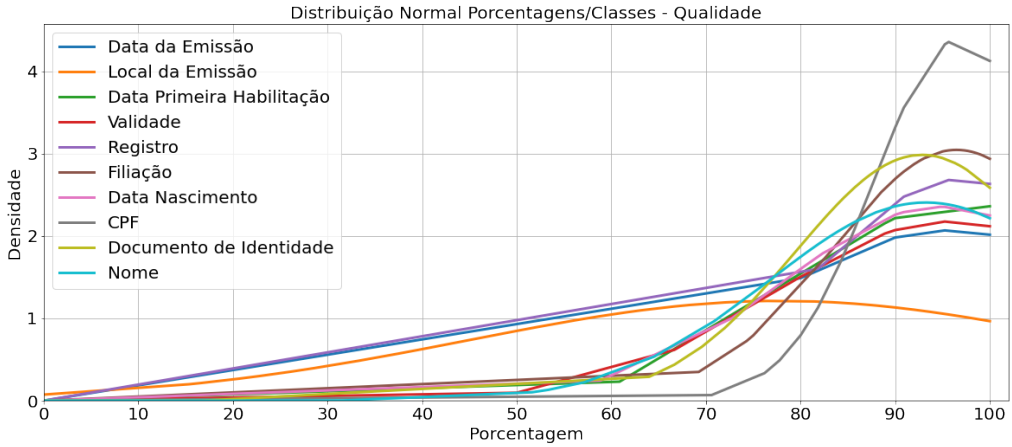
Fonte: Do Autor.

Figura 22: Curva normal das porcentagens de similaridade dos textos extraídos das imagens que não passaram pelo processo de aumentar a qualidade



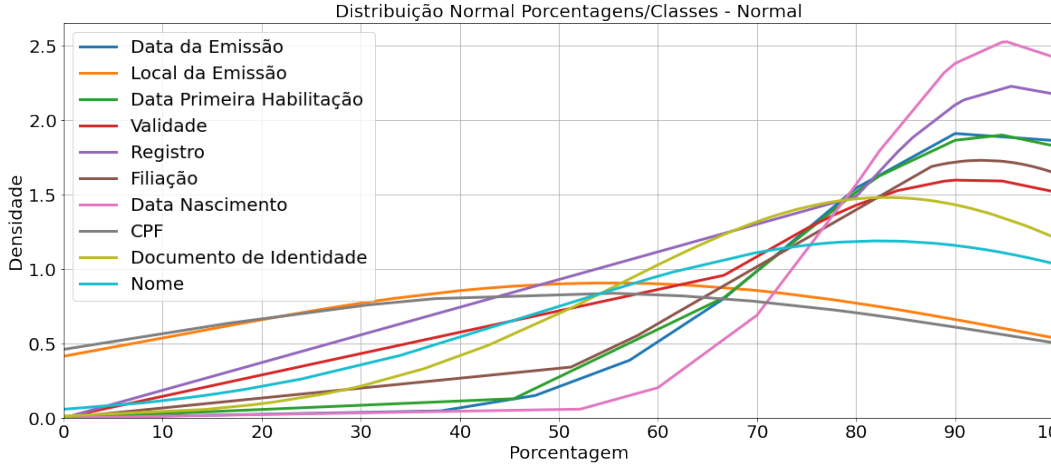
Fonte: Do Autor.

Figura 23: Curvas normais por classes das porcentagens de similaridade dos textos extraídos que passaram pelo processo de aumentar a qualidade



Fonte: Do Autor.

Figura 24: Curvas normais por classes das porcentagens de similaridade dos textos extraídos que não passaram pelo processo de aumentar a qualidade



Fonte: Do Autor.

5 CONCLUSÕES

Com o constante crescimento de plataformas digitais, também é crescente a quantidade de plataformas que solicitam o envio da foto da CNH para cadastro ou comprovação de autenticidade do usuário. Com isto, para reduzir tempo de espera até que um agente humano valide as informações e para realizar uma economia para

as plataformas digitais, o processo de automação da extração do texto dos campos da imagem da CNH vem se mostrando cada vez mais necessário.

Por se tratar de dados de cadastro ou validação, a acurácia há de ser a maior possível, evitando que dados errados sejam informados para as plataformas. O projeto elaborado buscou atender este requisito aplicando tecnologias consideradas estado da arte em visão computacional, o que gerou resultados promissores.

O presente trabalho evidencia a elevada capacidade do modelo YOLOv3 detectar os campos da CNH, demonstrando que o fator mais relevante para bons resultados, é a boa qualidade da imagem para ser realizada a extração dos textos. Como pode ser observado na última linha da Tabela 11, ao submeter as imagens ao processo de aumentar a qualidade com ESRGAN antes de realizar a extração dos textos, a porcentagem de similaridade do texto extraído com o texto real é otimizada em mais de 10%.

5.1 Proposta de trabalhos futuros

Conforme os dados da Tabela 11, o campo “Local de Emissão” é responsável pela menor porcentagem de similaridade entre o texto extraído e o texto real. Este campo representa o local onde a CNH foi emitida e seu valor é formado pelo nome do município seguido pela UF (Unidade Federativa).

Como forma de tentar contornar a baixa porcentagem de similaridade do texto extraído do campo “Local de Emissão” com o texto real, acredito que uma abordagem relevante para a pesquisa seja criar uma base de dados contendo todos os municípios brasileiros e suas respectivas UFs para, após a extração do texto do campo “Local de Emissão”, verificar a similaridade deste texto com todos os registros na base de dados de municípios e UFs e substituir o texto extraído pelo registro (município e UF) que obtiver maior similaridade com o texto extraído.

REFERÊNCIAS

COOPELAND, MICHAEL. What’s the Difference Between Artificial Intelligence, Machine Learning and Deep Learning. NVIDIA, 29, julho, 2016. Disponível em: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>. Acesso em 20, abril, 2022.

FANG, Jie; WANG, Pingbo. Application of improved YOLO V3 algorithm for target detection in echo image of sonar under reverb. In: Journal of Physics: Conference Series. IOP Publishing, 2021. p. 042048.

FRADKOV, Alexander L. Early history of machine learning. IFAC-PapersOnLine, v. 53, n. 2, p. 1385-1390, 2020.

FUKUSHIMA, Kunihiko; MIYAKE, Sei. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets. Springer, Berlin, Heidelberg, 1982. p. 267-285.

GERS, Felix A.; SCHMIDHUBER, E. LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks, v. 12, n. 6, p. 1333-1340, 2001.

HAYKIN, Simon. Redes Neurais: Princípios e Prática. 2 ed. Porto Alegre: Editora Bookman, 2007.

HINTON, Geoffrey E. et al. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

KOVÁCS, L. Z. Redes Neurais Artificiais. 3 ed. São Paulo: Editora Livraria da Física, 2006.

OH, Kyoung-Su; JUNG, Keechul. GPU implementation of neural networks. Pattern Recognition, v. 37, n. 6, p. 1311-1314, 2004.

RAPHAEL, Bertram. The thinking computer: Mind inside matter. San Francisco, 1976.

REDMON, Joseph et al. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 779-788.

REDMON, Joseph; FARHADI, Ali. YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 7263-7271.

SEVERANCE, Charles. Guido van rossum: The early years of python. Computer, v. 48, n. 2, p. 7-9, 2015.

SHAW, G. L. Donald hebb: The organization of behavior. In: Brain Theory. Springer, Berlin, Heidelberg, 1986. p. 231-233.

SMITH, Ray. Tesseract blends old and new OCR technology. Tutorial at DAS, 2016.

VEDALDI, A.; LENC, K. MatConvNet: Convolutional Neural Networks for MATLAB. In: Proceedings of the 23rd ACM international. vol. 15. p. 689-692. Out. 2015.

WANG, Xintao et al. Esrgan: Enhanced super-resolution generative adversarial networks. In: Proceedings of the European conference on computer vision (ECCV) workshops. 2018. p. 0-0.

YAMAGUCHI, Kouichi et al. A neural network for speaker-independent isolated word recognition. In: ICSLP. 1990.

ANÁLISE QUALI-QUANTITATIVA DE ARQUITETURAS DE REDES NEURAIS ARTIFICIAIS PARA GERAÇÃO DE MÚSICA POLIFÔNICA

Pedro Henrique Rodrigues Mendes

William Roberto Malvezzi

RESUMO

Cada vez mais a música é parte integrante de inúmeras atividades do cotidiano do homem moderno, desde a audição sutil e emotiva de uma trilha sonora em um longa-metragem até um som marcante o suficiente como um sinal de alerta indicando o término de uma sessão de votação em uma urna eletrônica. É possível criar novas músicas por meio da inferência do padrão aprendido das composições dos músicos compilados ao utilizar técnicas de arquiteturas de Redes Generativas Adversais (GANS), Redes Neurais Recorrentes (RNN), Unidade com Portas Recorrentes (GRU) e Memória de Longo e Curto prazo (LSTM). Os resultados da pesquisa convergem em definir quais arquiteturas de redes neurais estudadas conseguem gerar as melhores amostras dentro do estilo musical Bossa Nova proposto com a finalidade de ser um catalisador nas resoluções de problemas socioeconômicos e também gerar enriquecimento da cultura popular brasileira.

Palavras-chave: Redes Neurais Artificiais (RNA). Música. Aprendizado Profundo.

1 INTRODUÇÃO

A música como sendo uma combinação harmoniosa e expressiva de sons tem sido um elemento importante na experiência de usuários em diversas mídias consumidas na atualidade. Destarte, músicas, trilhas sonoras e efeitos sonoros são resultado do trabalho de uma equipe profissional, possuindo dois processos: técnico e criativo para compô-las. Segundo COCA et. al. (2011), observou-se que ao passar dos anos, os métodos de composição musical estão evoluindo e tornando-se uma mistura de métodos computacionais. Portanto, o profissional da área do audiovisual

deve estar aberto a dominar esses novos métodos, determinando assim um novo parâmetro de complexidade em atividades tanto técnicas quanto criativas para elaboração de material sonoros de qualidade.

A gênese musical é caracterizada por altos investimentos adiantados, baixos custos marginais e altos riscos (LEURDIJK e NIEUWENHUIS, 2012). Portanto, a composição de melodias para trilhas sonoras para mídias audiovisuais não é um recurso acessível a novos artistas emergentes para aplicarem a um produto de entretenimento. Segundo BERCHMANS (2006), a falta de investimentos na área, no período cinemanovista, contribuiu para a atuação dos próprios diretores como produtores musicais e compositores das suas próprias trilhas, dessa forma, restrições financeiras no meio artístico geram incentivo para inovação e, ao aplicar essas técnicas de aprendizado profundo em produção independente, um grande apelo ao atual crescimento da cultura Maker é induzido.

Com inteligência artificial, em 2018 estimou-se que a experimentação com machine learning está aumentando, e mostra enorme potencial oferecido por essa tecnologia nos campos da música, filmes e literatura (KULESZ, 2018). Portanto, tecnologias que envolvem o aprendizado de máquina visam consolidar um espaço dentro da indústria do entretenimento para atingir esses objetivos. Segundo CORRÊA (2008), uma das principais qualidades de modelos conexionistas é a capacidade de aprender padrões e características presentes nas melodias do conjunto de treinamentos e obter generalizações dessas características para a composição de novas melodias.

2 REVISÃO BIBLIOGRÁFICA

Será apresentada a seguinte fundamentação teórica em tópicos para execução da pesquisa com base em referências bibliográficas reconhecidas.

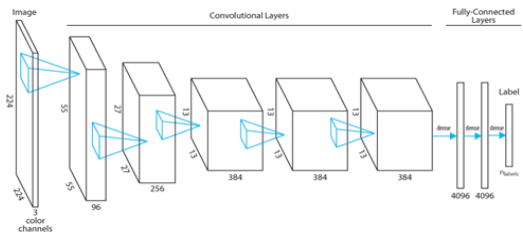
2.1 Redes Neurais Convolucionais (CNN)

O uso tradicional de rede neural artificial (ANN) para visão computacional implementa a abordagem na camada de entrada e a primeira camada escondida dimensionada ao tensor da imagem. Portanto, ao processar imagens com resolução

de 200x200x3 (200 de altura, 200 de largura e 3 canais de cores) as duas primeiras camadas da ANN teria a quantidade equivalente e relativa a 120.000 pesos. Segundo KARPATHY e JOHNSON (2018), essa grande conectividade presente entre as duas primeiras camadas é dispendiosa e esse imenso número de parâmetros certamente tende para que o modelo fique super ajustado, no inglês overfitting.

Segundo ALBAWI e MOHAMMED (2017), um dos aspectos mais benéficos de CNNs é a redução do número de parâmetros de entrada em uma ANN. Dessa maneira, a utilização de CNNs consiste em rearranjar camadas da rede disposta em três dimensões: largura, altura e profundidade que gera o volume das primeiras cinco camadas presente na Figura 1. A disposição de uma arquitetura CCN é constituída no exemplo na figura 1 que na esquerda da mesma figura é observada imagem a ser processada pela ANN, ao centro são as camadas que realizam convolução dos dados e por último na direita possui as camadas totalmente conectadas ANN capazes de sintetizar o output. Conforme PONTI e COSTA (2017), a camada convolucional irá processar a imagem e produzirá uma transformação por meio de uma combinação linear da vizinhança de 27 pixels da imagem em um único pixel de saída que resulta na construção do mapa de características do modelo.

Figura 1: A ilustração da arquitetura CNN AlexNET.



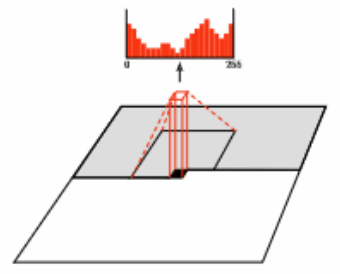
Fonte: Microsoft Machine Learning Blog (2017)

Após a camada de convolução, os resultados filtrados são compilados em um mapa de características. Esse mapa é responsável por armazenar informações de atributos dos dados em formato de vetores, formas e cores. Partes específicas da rede são ativadas ao encontrar elementos importantes dentro dos dados processados, como por exemplo vetores responsáveis pelos olhos ou orelhas de gatos ou cachorros em na foto processada. Outra função dentro do filtro de convolução é o pooling que é responsável por reduzir a escala dos mapas de características. Segundo

PONTI e COSTA (2017), essa operação por cálculos convolucional possui dois propósitos: em primeiro lugar visa reduzir o custo computacional, como a profundidade da rede costuma aumentar ao longo das camadas, é conveniente reduzir a dimensão espacial para melhor performance, em segundo, reduzindo o tamanho das imagens podemos obter um tipo de composição de banco de filtros multiresolução que processa imagens em diferentes espaços-escala.

Ao fim do processo, após criação dos mapas de características, as entradas são devidamente filtradas e processadas nas camadas totalmente conectadas, conforme no estágio mais à direita na Figura 1, que possui arquitetura de Multiple Layers Perceptrons (MLP) e tem de trabalho de realizar o output de classificação, porém, o escopo da pesquisa converge para geração de imagens, por meio da arquitetura PixelCNN. segundo OORD et al (2016), a ideia básica da arquitetura é usar conexões auto regressivas para modelar imagens pixel por pixel, decompondo a imagem original como um produto de condicionais.

Figura 2: A visualização da arquitetura PixelCNN realizando o mapeamento da vizinhança de pixel para predizer o próximo pixel, para gerar esse pixel o modelo pode apenas usar como condição o pixel anteriormente gerado.



Fonte: Oord et al (2016)

O PixelCNN é aplicável a essa pesquisa quando ao transformar as informações da música no arquivo midi em imagem e assim entramos no campo de visão computacional, portanto, a imagem será formatada em que na altura representa a escala do piano, largura progressão do tempo e no fim a profundidade lida com a ativação da nota, portanto, conforme a Figura 1 é possível reproduzir coesão e prosseguimento em uma melodia na geração de ativação e não ativação das notas em

uma imagem quando a arquitetura PixelCNN usa como parâmetro pixels gerados anteriormente.

2.2 Redes Neurais Recorrentes (RNN)

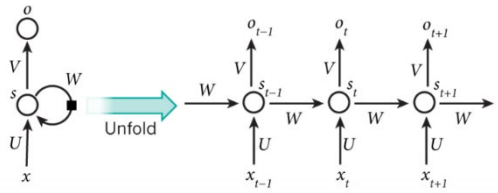
Uma ANN normalmente processa o dado de entrada em suas camadas e emite sua decisão, assim realizando um ciclo, com uso de RNN, um modelo preditivo permite armazenar informações da entrada anterior e usa-las como parâmetros novamente para novas decisões nas entradas subsequentes. Isso permite que as redes neurais realizem o processamento temporal e também aprendam sequências, como por exemplo: realizar reconhecimento, reprodução de sequência ou associação e predição temporal (BULLINARIA, 2015).

h_t=φW(〖Uh〗_(t-1) ,x_(t)) (a)

y=entrada(∑_t^N 〖Wh〗_t - θ) (b)

A equação a descreve como é possível perpetuar uma informação no passado, o sinal x_t passa pela função de ajuste de do estado oculto na figura 3, tem o somatório do sinal atual x_t com o estado anterior 〖Uh〗_(t-1), o produto da soma por fim é processado em uma função de ativação não linear φ com os parâmetros W do perceptron, resulta em novo sinal de entrada h_t do perceptron da rede com informações do sinal de entrada h_(t-1), portanto, a função básica do perceptron RNN é modificada para função b.

Figura 03: Ilustração do perceptron em uma Rede Neural Recorrente, o valor do peso W do perceptron é destrinchado (Unfold) entre os períodos de tempo: t+1 e t-1. Essa ilustração demonstra que o peso W do perceptron sempre está em alteração contínua por fatores temporais.



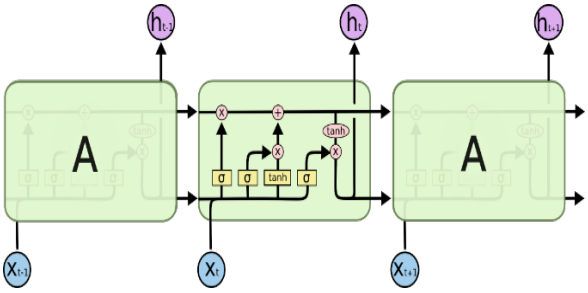
Fonte. BRITZ (2015)

Uma RNN pode aprender uma distribuição de probabilidade para uma sequência de valores, sendo treinado para prever o próximo símbolo na sequência (CHO et al, 2014). Portanto, o fato que esse tipo de arquitetura, a RNN, pode guardar informações pregressas e aplicá-las, é oportuno como método a ser aplicado nesse caso de estudo, pois a rede irá predizer a próxima nota musical com base na última nota tocada, criando uma coerência harmônica entre as notas criadas pelo modelo.

2.3 Redes Neurais de Memória de Longo e Curto prazo (LSTM)

Como apresentado, RNN armazenam feedbacks de conexões passadas recente para influenciar futuras entradas, simulando uma espécie de memória de curto prazo na rede neural, entretanto é interessante que uma rede tenha a capacidade de possuir uma memória de longo prazo, ao realizar a mudança dos pesos do estado oculto em um ritmo mais vagaroso. Segundo HOCHREITER e SCHMIDHUBER (1997), a rede neural LSTM pode aprender a conectar intervalos de tempo em mais de 1000 passos, mesmo em caso com ruído ou entradas incompressíveis sem a perda dos intervalos de tempo curtos.

Figura 04: Ilustração do funcionamento de uma rede LSTM de perpetuamento dos estados ocultos (h_(t-1)) ao longo do tempo, sinal de entrada (x_t) alimenta os retângulos amarelos que indicam uma camada da rede neural com dois diferentes tipos de funções de ativação: sigmóide (σ) e de tangente hiperbólica (tanh). Os vetores em preto indicam o fluxo percorrido pelos sinais. Os círculos na coloração rosa indicam operações aritméticas com resultados das camadas da rede neural, aplicando operações de soma (+), multiplicação (×) e tangente hiperbólica (tanh)



Fonte. OLAH (2015).

A arquitetura LSTM aplica a ideia de acumular, alterar e adicionar os valores dos estados ocultos pregressos em uma célula de memória ao longo do tempo, o perpetuamento desses valores pode ser ilustrado na figura 04 no vetor horizontal

passando no topo do diagrama, permitindo que a informação flua com a possibilidade de não ocorrer alteração do conteúdo entre os processamentos de sinais na rede neural.

$$f_t= \sigma(w_f\times[h_{(t-1)},x_t]+ b_f) \quad (c)$$

$$i_t= \sigma(w_i\times[h_{(t-1)},x_t]+ b_i) \quad (d)$$

$$c_t= \tanh(w_c\times[h_{(t-1)},x_t]+ b_c) \quad (e)$$

Para permitir que retire e adicione valores no estado escondido, LSTM utiliza a estrutura Gates, segundo OLAH (2015) esses Gates são compostos de uma camada da rede neural com função ativação do tipo sigmóide (σ) ou tangente hiperbólica (tanh). O primeiro Gate, com a sigmóide (σ) na figura 4, é conhecida como "Porta de Esquecimento", quando a saída da função c é igual a 0, apaga os valores armazenados no estado escondido (h_(t-1) multiplicando pelo valor nulo obtido. Segunda Gate é responsável determinar a inserção de novos valores no estado escondido (h_(t-1)), possui duas camadas da rede neural conforme na figura 4 e tem sua função ($i_t \times c_t$)+ h_(t-1) , a primeira função d de ativação sigmóide (σ) irá determinar se o novos valores candidatos criados pela camada de ativação na função e serão aplicados e ativação da tangente hiperbólica (tanh) cria novos valores para novo estado escondido (h_(t-1)), caso dê novamente 0 resultado da ativação sigmóide (σ), portanto, não será alocado novos valores para h_t, pois o resultado produto é igual a 0 e , portanto, sua soma com h_(t-1) no final do processo não irá oferecer alterações.

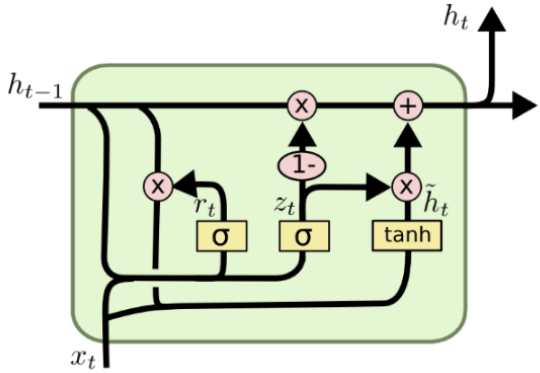
$$o_t= \sigma(w_0\times[h_{(t-1)},x_t]+ b_0) \quad (f)$$

$$h_t=o_t \times \tanh(ct) \quad (g)$$

Ao fim do processo haverá a decisão da rede com a entrada x_t em questão, após a obtenção dos resultados das Gate de esquecimento e o Gate de inserção, o sistema possui um h_(t-1) final para ser agregado na decisão. A camada da rede neural que é utilizada para a decisão é constituída pela função f, de ativação sigmóide (σ) já utilizada antes nas estruturas Gates, portanto, o sinal resultante dessa camada é multiplicado pelo produto da função tangente hiperbólica do valor h_(t-1) presente na função g, no final gerando um valor de saída do sistema h_t influenciado

por saídas anteriores para que na sequência também possa influenciar a saída h_(t+1).

Figura 7: Infográfico do funcionamento da arquitetura GRU.



Fonte: DRAKOS (2019).

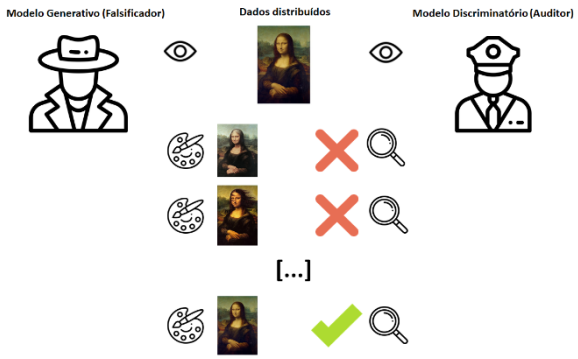
Outra arquitetura com característica recorrente mais recente é a GRU, na linguagem portuguesa Unidade com Portas Recorrentes, utilizando com base os estudos NAYEBI e VITELLI (2015), utilização da arquitetura GRU é similar na arquitetura conforme a figura 4 do LSTM em comparação a figura 7 do GRU. Ambas arquiteturas possuem portões de esquecimento (Forget Gate ou Reset Gate) e os portões de inserção (Update Gate). A diferença detectada entre as arquiteturas ocorre na GRU que a informação não é esquecida, sim obrigatoriamente incrementada no T-1, ao passo que na arquitetura LSTM o seu segundo portão controla a adição de informações ao estado T-1, sendo assim em uma caracterização abstrata. O GRU possui um T-1 que muda mais lentamente que a RNN simples e mais rapidamente que uma LSTM.

2.4 Redes Neurais Generativas Adversais (GANS)

Uso de GANS consiste no uso duas ANN multicamadas para síntese de novos dados com uso de técnicas de aprendizado supervisionado ou não supervisionado. Segundo GOODFELLOW et al. (2014) o método de redes adversárias consiste em um modelo generativo opondo-se a outra rede adversária: o modelo discriminatório que aprende a determinar qual amostra foi gerada por meio do modelo generativo e ou se é da distribuição dos dados, esse conceito é exemplificado na figura 5 que a

arquitetura GANS pode ser explicado, em um formalismo simples por meio dessa ilustração, ao realizar algo semelhante a uma relação entre um falsificador e um auditor. O falsificador tem o objetivo de criar uma obra de arte a fim de enganar o auditor com vistas a realizar um lucro fácil, que por sua vez o auditor irá verificar essa obra e julgar se a obra apresentada é realmente falsa ou supostamente verdadeira. Assim, cada um desses dois agentes irá testar a habilidade do outro, após interações da recusa do auditor, esse falsificador atinge certo grau de excelência após o repetitivo aperfeiçoamento das técnicas de síntese da obra por meio da tentativa e erro, em um determinado espaço de tempo, o auditor não saberá distinguir se a obra verificada é verdadeira ou falsa por conta desse aprendizado por reforço.

Figura 05: O modelo GANS exemplificado e ilustrado.



Fonte: própria (2019).

O método GANS é uma alternativa para criação de modelos deep generative, pode produzir algoritmos de treinamento específicos para variados tipos de modelos e otimização de algoritmos (GOODFELLOW et al., 2014).

$$E(G,D)= 1/2\ \epsilon_{x\sim pt}\ [1-D(x)]+1/2\ \epsilon_{z\sim py}\ [D(G(x))] \quad (h)$$

Por ser constituído por duas ANN, generativa G(x) e uma discriminatória D(x), essa arquitetura possui um ponto importante na sua função h de erro, capaz de gerenciar o erro das duas ANN. Ao treinar o gerador, queremos maximizar esse erro enquanto tentamos minimizá-lo para o discriminador (ROCCA, 2019).

2.5 Bossa Nova

O movimento musical bossa nova é um retrato do contexto conturbado presenciado no Brasil durante o período das décadas de 50 e 60 os programas de desenvolvimento durante a presidência de Juscelino Kubitschek (1950-1955), transformaram as paisagens urbanas pelo país e, consequentemente, geram uma agitação e concentração cultural tanto de âmbito nacional e internacional nesses centros.

A massificação e distribuição da cultura pelo uso das tecnologias de rádio e a recente televisão, importação de influências estrangeiras como o Jazz e o Blues convergiram com a poesia e ritmo do samba carioca brasileiro no estilo musical brasileiro "Bossa Nova". Segundo CALDAS (2005) afirma que a Bossa-Nova trouxe um novo ritmo de música, batidas sutis no violão, acordes, dissonantes, arranjos musicais inusitados com a mescla da improvisação do jazz e uma nova forma de interpretar o samba em um movimento que, inicialmente se constituiu como um movimento artístico-musical da zona sul carioca.

Figura 06: Da direita para esquerda em sequência na fotografia estão dispostos: Toquinho, Tom Jobim e Vinicius de Moraes em momento de ensaio.

Fonte: FONSECA (2016).

Alguns dos principais agentes culturais desse movimento foram Vinícius de Moraes, Tom Jobim, Nara Leão, João Gilberto e Toquinho e alguns deles estão presentes na Figura 6. No exterior, a Bossa Nova é o mais conhecido gênero de composição de música popular brasileira, pois, os bossa-novistas tiveram a oportunidade de se apresentar e gravar no internacionalmente, principalmente nos

Estados Unidos e com participação de outros artistas populares naquele presente momento como Frank Sinatra e Stan Getz (MARQUES, 2011).

Um ponto importante de parâmetro para criação e caracterização de uma música do estilo “Bossa Nova” deve-se no o atributo harmonia, segundo MACHADO (2009) a harmonia é um dos elementos presentes mais importantes no estilo bossa nova, a complexidade nesse fator remete a sofisticação que causa admiração de músicos e da população em geral.

2.6 Word2Vac

Segundo MIKOLOV et Al. (2013), word2vac consiste em criar um vetor de informações baseado na similaridade entre os símbolos armazenados neste vetor, portanto, o modelo irá agrupar símbolos conforme o grau de similaridade baseado na incidência desencadeada. Podemos usar o como exemplo o vetor 13 como uma simulação do modelo word2vac.

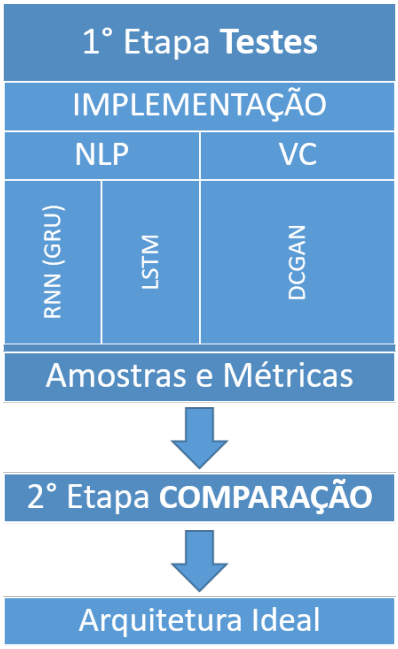
[Reino, Rei, Reinado, Rainha e Coroa] (1)

Sabe-se que o vetor 1 foi formulado com base em um hipotético corpus de textos, ao analisar esse vetor, sabemos que sempre que a palavra Reino for citada em uma sentença, logo em seguida pode ocorrer uma alta probabilidade que a palavra seguinte seja Rei ou a palavra Rainha, pois, semanticamente são palavras com alta similaridade.

3 METODOLOGIA DO TRABALHO

A pesquisa segue a metodologia quali-quantitativa para gerar uma análise de arquiteturas de ANN’s no desenvolvimento de composições de músicas polifônicas do gênero Bossa Nova. No fluxograma 1, é especificado o processo de implementação da pesquisa.

Fluxograma 01: Fluxo da metodologia da pesquisa.

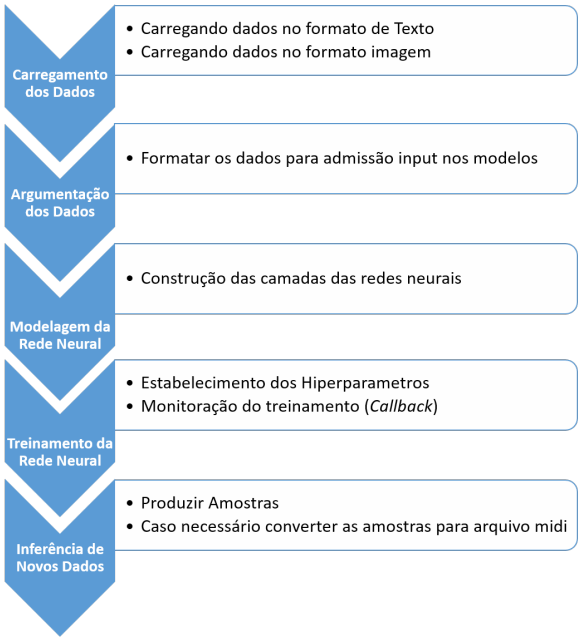


Fonte: própria (2019).

A primeira etapa presente no fluxograma 1, nomeada como Testes, possui o objetivo de construir um comparativo das arquiteturas aplicadas do referencial teórico com a finalidade de declarar o mais eficiente dentre elas com base nos resultados obtidos. Portanto, para conseguir uma comparação justa entre todas as arquiteturas, foi definida uma mesma condição para todos. Houve o emprego de um conjunto único e compartilhado de dados e também um valor "100", fixo de épocas no treinamento dos modelos.

A implementação de tais técnicas serão divididas em dois campos de atuação: (PLN) Processamento de Linguagem Natural e (VC) Visão Computacional, cada um desses campos possui uma abordagem de treinamento e manejo com os dados diferentes, porém, segue o mesmo fluxo de desenvolvimento de modelos de aprendizado de máquina presente no fluxograma 2.

Fluxograma 02: Fluxo do desenvolvimento dos modelos preditivos utilizados na pesquisa.



Fonte: própria (2019).

Com base nos resultados gerados por esses modelos sob mesmas condições, entramos na segunda etapa do fluxograma 1, nomeada como Comparação, primeiramente avalia o coeficiente lógico Loss, significa perda ou erro na língua portuguesa, registrado no treinamento da arquitetura e esse coeficiente é demonstrado pela equação i ou comumente conhecido como o cálculo de erro médio quadrático (MSE).

$$Loss(y,\hat{y}) = \sum_{n=1}^{n}(y-\hat{y})^2 \text{ (i)}$$

y é o valor real e o yˆ é o valor inferido, portanto, é calculado a diferença do quadrado desses valores que denota um índice de assertividade para essas predições.

Um outro parâmetro de comparação aplicado durante a etapa de comparação é a realização de uma análise musical, portanto, é preciso determinar atributos que sejam importantes para uma música pertencer ao estilo “Bossa Nova”, O ponto chave já discutido é a análise da harmonia presente nas amostras, neste caso uma boa forma de sintetizar uma música é utilizando a técnica de Redução Musical Harmônica (RMH), segundo SIMONETTA et Al. (2018) a técnica RMH consiste

em um sistema representado por gráfico que teoricamente representa música pela perspectiva horizontal (Contraponto) e vertical (Harmônica) simplificada sem perder informações contextuais e harmônicas. Com RMH podemos calcular similaridade harmônica com outras músicas utilizando um modelo word2vec com o fim de achar relação entre acordes em composições “Bossa Nova”, portanto, o produto de um RMH pode ser representado no vetor 12, neste exemplo foi utilizado a música Águas de Março, o conteúdo do vetor informa os principais acordes que compõem a estrutura harmônica na notação numérica romana.

['III64', 'III64', '#IV6', 'II43','IV', 'I', 'IV7'] (12)

A técnica de similaridade por distância word2vec que é utilizado em problemas de PLN também pode ser aplicado na música, ao lidar com harmonia vemos formações de acordes que ligam e dão profundidade para melodia são características plenas em músicas do gênero “Bossa Nova”. A pesquisa resulta na definição da arquitetura com a melhor performance comprovada segundo o processo de comparação desenvolvido.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Conforme a metodologia seguida, a seção de resultados seguirá uma estrutura semelhante ao processo presente no fluxograma 1. A base de dados utilizada ao longo da pesquisa é proveniente de várias fontes, portanto, o motivo é que não há uma fonte oficial ou acadêmica que centralize arquivos midi do gênero “Bossa Nova”, como normalmente acontece com compositores famosos de música clássica. Portanto, não é garantido que esses materiais sejam a mesma composição conforme seus artistas originais conceberam, mas sim versões alteradas ou regravações produzidas por terceiros. Houve uma análise e filtragem prévia de todas as músicas na construção da base de dados para verificar se é o conteúdo corresponde que os títulos propostos em cada arquivo midi.

4.1 Testes (Implementação)

Foi proposto na pesquisa a utilização de uma série de arquiteturas de redes neurais, e, por isso, o escopo da pesquisa foi dividido conforme suas similaridades,

PLN agrega as arquiteturas LSTM e GRU onde usam técnicas de linguagem natural, outro escopo é o VC com as arquiteturas GAN e CNN que envolvem a área de visão computacional. Essas implementações possuem o objetivo de gerar dados e insumos para a comparação proposta ao fim da etapa. Portanto, será discutido a seguir como e o por quê foi construído o modelo em cada arquitetura proposta:

4.1.1 PLN

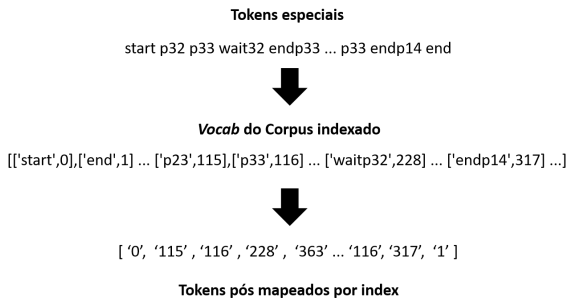
Durante a construção referencial teórico e a pesquisa de artigos acadêmicos relacionados durante a execução desse projeto, a técnica PLN foi o método com maior número de referências com a intenção de geração de música por inteligência artificial, dentro desse escopo é importante citar os trabalhos de TODD (1989), CHEN e MIIKKULAINEN (2001), YÜKSEL (2011), COCA et al. (2011), NAYEBI e VITELLI (2015) e PAYNE (2018). Uma constante encontrada dentro do trabalho desses autores é o método de usar fenômeno de gênese sequencial de notas ou acordes musicais, originalmente essa abordagem foi concebida por TODD (1989), consiste em que cada nota musical gerada possui uma dependência de memorização de notas geradas no passado para ao fim criar uma coesão musical na totalidade da sequência formada.

Conforme descrito, arquiteturas recorrentes permitem criar um estado oculto na rede neural semelhante a uma memória. A implementação do PLN foi norteada pela a pesquisa da PAYNE (2018), que utilizou de símbolos para codificar as respectivas notas e acordes do formato MIDI para texto, ao final processando-os usando a arquitetura AWD-LSTM. A autora separou a pesquisa em duas abordagens, utilizando modelos para notas e acordes, segundo PAYNE (2018), a representação com uso de acordes musicais trouxe resultados como pouca generalização, consequentemente não conseguiu lidar com diferentes durações de notas dentro do acorde e sendo mais efetivo para criar uma amostra com a característica de ritmo constante. A representação com uso de notas musicais criou amostras mais autorais, criou melodias com coerência rítmica e lidou bem com notas com longa duração. Visando o gênero em estudo, “Bossa Nova”, orientado nos resultados da pesquisa de PAYNE (2018), foi decidido utilizar ambas representações, notas e acordes

musicais, em ambos modelos, pois se houvesse a utilização de apenas uma representação, poderia haver uma remoção de características importantes ou determinantes presentes no gênero musical.

Para modelar os dados desse problema, a fim de processá-lo em uma rede neural, foram utilizadas como ferramentas a linguagem de programação python 3.6 aliados com as bibliotecas music21 de manipulação de músicas e Keras para criação de modelos de redes neurais com backend tensorflow. A biblioteca music21 foi a ferramenta que facilitou a transcrição dos arquivos MIDI para notas e acordes musicais em formato texto. A única argumentação aplicada nos dados ocorreu na tokenização e mapeamento do vocabulário do corpus das notas, a tokenização consiste em quebrar uma estrutura padrão de textos em documentos, estrofes e frases para símbolos dispostos organizado em um vetor. O mapeamento na figura 7 ilustra a ação de converter o próprio corpus ou textos futuros em uma notação numérica inteira, para realizar essa conversão, necessita a criação de um vocabulário, que se refere a uma lista com cada símbolo diferente presente neste na totalidade do corpus texto e esses símbolos são indexado em números inteiros.

Figura 7: Infográfico do mapeamento de símbolos de uma sentença, redes neurais apenas conseguem processar símbolos numéricos em suas entradas.



Fonte: própria (2019).

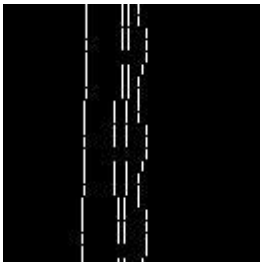
Foi realizado um teste utilizado a arquitetura com perceptrons RNN simples conforme na Figura 3, como previa na metodologia e referencial teórico, o treinamento não ocorreu da forma esperada pois o Loss do modelo não convergiu para mínimo global, embora fosse realizado alterações nos hiperparâmetros e também na base dados, que gerou amostras não próximas a uma música porque emitia apenas uma nota musical em uma indefinidamente repetição. Percebemos a

arquitetura RNN presente na equação a sempre irá guardar informações de parâmetros de entradas de inferências passadas (t-1) para atual inferência (t) e ocasiona a tendência criar loops fechados de notas musicais quando apenas duas notas se relacionam no aprendizado adquirido no treinamento. Portanto, para criar uma melodia complexa é necessário a arquitetura utilizar-se de memórias de longo prazo no modelo.

4.1.2 VC

Técnicas de aprendizado profundo atreladas ao uso de visão computacional para a resolução de problemas complexos estão no ápice na data de publicação dessa pesquisa, segundo KRIEGESKORTE (2015), o modelo convulacional AlexNet em 2012 marcou o ponto de partida da dominância de redes neurais na área de VC, as taxas de erros reduziram drasticamente em comparação aos resultados obtidos antes da implementação de arquiteturas CNN's e chegam ao ponto de disputar com a performance humana em classificação e detecção de objetos. A intenção da pesquisa é aproveitar a crescente no campo do conhecimento de VC e replicá-la em um sistema de geração de melodias. Há um reduzido número de pesquisas que buscam a síntese musical na abordagem VC, dentro dos trabalhos encontrados podemos citar HUANG et al. (2017); DONG et al. (2017) e do YANG, CHOU e YI-HSUAN (2017). A característica em comum entre essas pesquisas citadas é que se baseiam no uso arquiteturas ou técnicas já consolidadas de VC, dentre elas temos o processamento de imagens no formato de tensors e também o uso de arquiteturas de redes neurais GAN e CNN. O trabalho de DONG et al. (2017) consiste no modelo Musegan para geração de melodias com a arquitetura GANs, essa arquitetura normalmente utiliza de imagens para o processamento, para isso os autores da pesquisa modelaram o problema em pianorolls, é um modo de representação visual de uma música.

Figura 8: Exemplo de uma visualização 128x128 Pianoroll de um trecho da música Águas de Março da composição de Tom Jobim 1972.



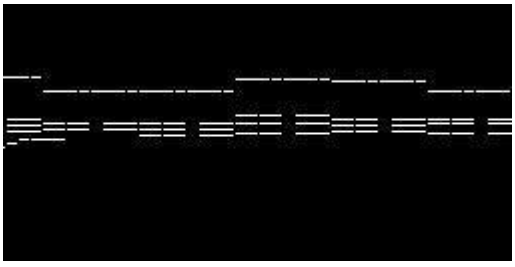
Fonte: Própria (2019).

O pianoroll é uma matriz P^(s×e), o atributo s denota a variável step que é o tempo ou compasso presente na melodia pianoroll, e a variável e indica a escala musical C maior, que também pode ser alterada para outras configurações de escalas, portanto, possuindo um total de 128 notas (de C0 para G10) na referida escala. A união desses atributos cria um mapeamento de step (tempo) e em conjunto também na escala, assim resultando em uma matriz onde a posição do valor 1 dentro dessa matriz é a coordenada para o acionamento de uma nota musical dentro do eixo da escala no determinado instante no eixo tempo, com isso é possível gravar uma música em uma forma de visual conforme na Figura 8. Uma matriz pianoroll pode possuir mais uma dimensão se for caracterizado do tipo Multitrack, portanto, terá a notação P^(s×e×t), em que t é uma faixa de instrumento presente no arquivo midi, no caso dessa pesquisa será apenas utilizada o canal do instrumento piano acústico clássico.

A pesquisa de HUANG et al. (2017) também utiliza de pianorolls como estrutura de dados na modelagem do problema, porém, ao invés da geração de melodias em si, o conteúdo da pesquisa foi base para construção do modelo COCONET que aborda a geração de novas harmonias conforme os padrões de Bach. Segundo HUANG et al. (2017) COCONET realiza a reconstrução de uma nova harmonia dentro de um pianoroll já existente por contraponto de notas pelo método convolutivo. O modelo COCONET foi conhecido pelo doodle do google em comemoração do aniversário de Bach que possibilitou a interação de todos os usuários dessa ferramenta de pesquisa na data 21 de março de 2019 a utilizarem o funcionamento modelo em questão.

Por último, utilizando CNN na síntese de melodias tem a pesquisa YANG, CHOU e YI-HSUAN (2017) com o MIDINET baseia na arquitetura GANS com o agente discriminador e gerador em arquitetura CNN em já referido pianorolls, uma diferença que o autor realizou testes utilizando a representação no campo dos acordes e das notas. Segundo YANG, CHOU e YI-HSUAN (2017), obteve bons resultados ao utilizar acordes e as amostras avaliadas por grupos de músicos e não-músicos concordaram que possuíam as qualidades superiores como: agradável e mais realístico em comparação ao modelo com notas. O modelo MIDINET conseguiu compor com característica de linearidade de tempo. Segundo YANG, CHOU e YI-HSUAN (2017), a arquitetura CNN consegue reproduzir resultados semelhantes a um modelo RNN, que utiliza recursividade de informações, provando que modelos convolucionais também conseguem lidar com fatores temporais.

Figura 9: Exemplo de uma visualização 128x254 Pianoroll de um trecho da música Águas de Março da composição de Tom Jobim 1972. Também foi translado os eixos step e escala em comparação ao

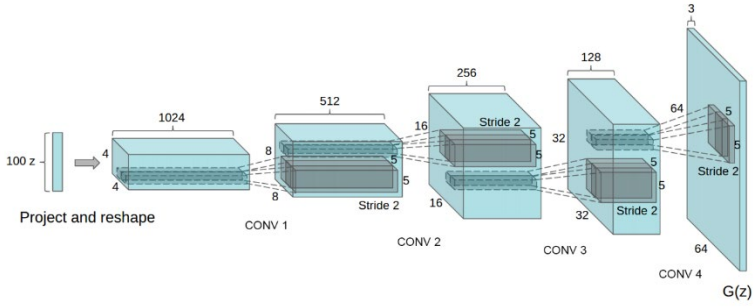


Fonte: Própria (2019).

A efetividade de utilizar a representação pianorolls como modelagem para sistemas complexos de ANN em geração ou manipulação de melodias com VC foram evidenciadas nas três pesquisas citadas anteriormente, define então a representação dos dados escolhida nesta pesquisa. Definido que a dimensão das pianorolls será P^(128×128) conforme figura 8 ou a P^(254×128) presente na figura 9 e respectivamente foram criados dois conjuntos de dados com 5.154 e 2.525 de imagens que correspondem a pianorolls de músicas de estilo bossa nova no formato midi. A biblioteca pypianoroll da linguagem python foi utilizada para converter arquivos midi do projeto para o formato pianoroll, esta biblioteca foi concebida pelos autores da pesquisa DONG et al. (2017).

A implementação do modelo GAN ocorreu com pianoroll no formato png da figura 8, pois é constatado que modelos GAN exigem alta demanda computacional e com os recursos obtidos para realização da pesquisa a decisão sensata foi processar essas imagens na dimensão 128x128. Para otimizar os recursos computacionais já obtidos, buscou-se um estudo para avaliar qual melhor arquitetura derivada no GAN seria possível de aplicar nessa pesquisa, conforme já discutido sobre tipologia de redes convolucionais em problemas de visão computacional, a pesquisa de RADFORD et al. (2015), propôs implementar nas arquiteturas GAN tradicionais ao aplicar no modelo gerador G(X) a arquitetura CNN conforme na figura 10. Segundo RADFORD et al. (2015), o uso de CNN obtém de benefícios um treinamento da rede mais estável ao manipular propriedades aritméticas de vetores presentes na amostra gerada com maior facilidade em cálculos convolucionais. Os autores RADFORD et al. (2015) nomearam essa arquitetura de DCGAN, Redes Adversarias com Convolução Profunda, analisando a figura 9 expõem que é possível dispensar o uso de camadas fully connected e camada pooling no modelo generativo, esses tipos de camadas aumentam significativamente o número de parâmetros variáveis presente no modelo e subsequentemente ampliará o poder computacional necessário para realizar o treino de uma época.

Figura 10: Ilustração de um gerador G(X) DCGAN que o input Z com distribuição dimensional 100 é processado por 4 filtros convulsionais de diferentes dimensões até projetar uma representação em uma imagem gerada 64x64.



Fonte: RADFORD et al. (2015).

4.2 Testes (Comparação)

A implementação dos modelos levantou dados estatísticos e também informações relevantes. Como demonstrado na metodologia, temos que separar os

resultados em duas análises: Análise de Performance dos modelos e a Análise Musical das amostras.

4.2.1 Análise de Performance

Antes de comparar os resultados, os modelos LSTM e GRU puderam ser implementados seguindo as mesmas condições propostas. Entretanto houve uma exceção ao implementar o DCGAN, que utiliza imagens, precisou-se de variações nas condições propostas, dentre elas aumento do coeficiente de épocas que foi mudado para "200" e foi utilizada a totalidade da base de dados. A argumentação dos dados para PLN consegue extrair bastante informação de um arquivo midi, pois como lidamos com texto, uma nota ou acorde da música vira um símbolo no nosso corpus de texto, enquanto ao transpor esse problema para o campo de VC, essa mesma nota ou acorde é transformada em uma representação em pixel que sua vez integra uma imagem 128x128 que por fim essa imagem compõe uma base de dados com outras imagens similares.

Tabela 1. Resultados da etapa Testes, coeficiente Loss. *O DCGAN é uma arquitetura que envolve o uso de duas redes distintas, portanto, será informado o Loss referente ao D(x) Discriminador e também do G(x) Gerador.

Amostra	Loss
LSTM	0.1067
GRU	0.1808
DCGAN*	$D(x)$ 0.1799 / $G(x)$ 4.1790

Fonte: Própria (2019).

A minimização do Loss indica que o sistema está conseguindo modelar o problema conforme os objetivos declarados, na tabela 1 observamos a arquitetura que mais conseguiu assimilar o problema foi o modelo LSTM no ponto de vista de Loss, indicando que convergiu mais perto para o mínimo global do problema. O modelo GRU que apresenta arquitetura semelhante ao LSTM e condições iguais de teste, mostrou um desempenho com uma margem de diferença de 0,0741 no coeficiente Loss. Essa diferença pode envolver o fato que o estado oculto da

arquitetura GRU muda mais rapidamente que a LSTM, sendo assim o modelo esquece de parâmetros ainda importantes para a inferência presente. Avaliar métricas de arquiteturas GAN é complexo, segundo BORJI (2018), é explícito que os modelos GAN são tipicamente orientados para maximizar aproximação a distribuição de dados inferida, utilizando um modelo parametrizado da distribuição real.

O ocorrido durante a implementação que as métricas de treinamento do conjunto de modelo DCGAN tendem ora para resultados do Discriminador ou do Gerador, semelhante a um cabo de guerra, Segundo BORJI (2018), é percebido pequeno esforço em avaliar modelos na arquitetura GANs, pois avaliações embasadas em quantitativa e qualitativas estão ainda faltando para esse tipo de arquitetura. Um fator importante que deve ser exposto é que um pianoroll, como na figura 7, possui uma dominância na incidência da cor preta em comparação a cor branca utilizado no acionamento das notas, em uma situação hipotética esse modelo poderia conseguir um bom índice no Loss em apenas gerar uma imagem totalmente na cor preta e isso complica mais ainda como avaliar essa arquitetura com as métricas escolhidas. Podemos ver na tabela 1 que o modelo DCGAN obteve no Discriminador um Loss de 0.1799, no decorrer do treinamento o gerador aumentava seu Loss sempre que o Discriminador diminuísse o dele.

A implementação do bloco de PLN ocorreu de formas semelhantes, a um teste empírico, foi utilizada a base dados inteira e para treinamento do modelo LSTM e que por sua vez não convergiu provável pelo imenso corpus gerado por essa base de dados e o modelo não conseguiu formar uma convergência ao possuir um grande espaço amostral no vocábulo criado. Então tentou-se reduzir essa base de dados gradualmente, de 93 arquivos midi para 15. Depois dessa redução, ambos modelos GRU e LSTM conseguiram convergir para um coeficiente Loss baixo registrado na Tabela 1. As amostras foram criadas em PLN com uma duração de 127 segundos, portanto, 500 símbolos foram gerados no output.

O DCGAN foi implementado e treinado originalmente com “100” épocas, atingindo métricas de Loss D(x) 0.2061 e G(x) 3.3668, esse Discriminador ficou com índice relativamente perto do registrado com da tabela 1, porém, as amostras

estavam fornecendo melodias com baixa qualidade. Um ponto importante é que as amostras geradas por DCGAN conseguem reproduzir 10 segundos de melodia. Seguindo a fonte do código para desenvolver um DCGAN, o autor SUMA (2019) utilizou o número de "300" épocas no problema dele, então foi atualizado para "200" épocas em aproximadamente 6 horas de treinamento continuo. O resultado foi satisfatório das amostras geradas, com um treinamento com maior quantidade de épocas do modelo seria possível que a qualidade melhorasse mais ainda, e com isso deparamos com uma grave desvantagem em poder de comparação com os modelos PLN, é exigido um alto custo computacional para treinar o modelo DCGAN e na sua saída gerar amostras de 10 segundos de duração.

4.2.2 Avaliação Musical

O atributo mais destacado e questionado pela pesquisa é a qualidade musical das amostras geradas por ANN, embora as métricas dos modelos sejam importantes, e nós dão o caminho para obter a melhor modelagem do problema, porém, nada basta ter boas métricas de loss no treinamento do modelo se ao final as amostras são algo próximo de um ruído. Foram geradas 4 amostras de cada arquitetura para realização da comparação e essas amostras estão presentes no repositório X. Para realizar o teste de similaridade é necessária uma música de parâmetro que seja indiscutivelmente do gênero "Bossa Nova", com isso foi escolhida aleatoriamente a canção Águas de Março.

Tabela 2. Resultados da análise de Similaridade Harmônica com Redução Harmônica.

Amostras	Similaridade Harmônica
LSTM	0,99998427
GRU	0.99998725
DCGAN	0,99991589

Fonte: Própria (2019).

Com o processamento dos dados para representação em RMH, o resultado foi estruturado na tabela 2, observamos que os índices estão muito próximos onde se alteram após a quinta casa após a vírgula, e também obteve um maior grau de

similaridade harmônica com uso da arquitetura GRU, com uma pequena margem em comparação ao LSTM. Para entender esse ocorrido, é preciso ouvir as amostras, portanto em todas 4 amostras LSTM tivemos incidência de cópias de músicas presentes na base de dados, indicando um claro overfitting. Enquanto nas amostras 1, 2 e 4 GRU temos uma pobre harmonia, e em grande parte do tempo temos apenas a audição da melodia da música, na amostra 3 é a única que consegue observar uma estrutura harmônica. Ao fim com as amostras DCGAN possuem desvantagem ao apenas ter 10 segundos de material sonoro, é observado também que foi gradas amostras primitivas de músicas com dissonância em certas partes, a melhor amostra observada foi a 4 que chegou mais perto de criar uma harmonia.

5 CONSIDERAÇÕES FINAIS

Um dos objetivos alcançados foi a geração de amostras musicais no gênero "Bossa Nova'' com uso de técnicas de PLN e VC conforme os resultados de Similaridade Harmônica na tabela 2 e ambas abordagens precisam de uma pesquisa mais aprofundada, pois só foram abordados aspectos já estudados de cada arquitetura envolvida, também o conteúdo das amostras geradas apontam um grande potencial futuro. Os métodos de LSTM e GRU conseguiram desempenhos perto do semelhante, porém, no atributo harmonia o LSTM é notavelmente superior aos seus rivais, embora o problema com overfitting o prejudique, muito é por consequência da base de dados ter um tamanho reduzido, possuindo apenas 15 músicas midi aleatórias, e somado com o uso técnica da Greedy Search Decoder que consiste em uma função simples argmax na camada de saída do modelo que irá inferir a presente palavra t com base na maior probabilidade dada palavra t-1 passada. Relacionando os dois pontos citados, temos uma menor variabilidade de notas e acordes no treinamento e somado a um sistema de geração de símbolos guiado na maior probabilidade, nisso temos como resultado a repetição e plágio da base dados por que o modelo não conseguiu generalizar.

DCGAN obteve resultados ainda perto do primitivo, porém, por consequência de limites computacionais enfrentados na pesquisa, esse modelo pode evoluir igual ao que foi presenciado nos testes. GRU conseguiu generalizar bem as amostras no aspecto de melodia, embora com presentes dissonâncias, a falta de harmonia em

determinados instantes das amostras não há uma explicação lógica para sua ocorrência. Por fim, os resultados convergem para melhor arquitetura sendo a LSTM, passível que com mais foco esse modelo possa produzir materiais com maior qualidade agregada dentro do gênero “Bossa Nova” e fugir do overfitting.

Como sugestão de trabalhos futuros, inicialmente faltou implementar uma nova arquitetura no estado da arte em PLN que surgiu no desenvolvimento da pesquisa que foi arquitetura Transformers, consiste no uso de ANNs encoders e decoders, um ponto importante dessa arquitetura, é que ela foi base para o modelo GPT-2 da empresa OpenAI que repercutiu no meio acadêmico e também na mídia de massas com seus feitos em geração de texto.

Segundo ponto é um estudo ampliado da aplicação de outras técnicas de Search Decoder além do método Greedy utilizado na pesquisa, portanto, ainda há outros métodos que podem ser testados na geração de músicas como o: Beam Search, Nucleus Sampling e o Top-K Sampling.

Carece na pesquisa a existência de um método estatístico para avaliação abrangente das músicas geradas por redes neurais, o trabalho de avaliar essas amostras possui aspectos tanto objetivos quanto subjetivos que normalmente necessita de um especialista na área da música para fazê-lo. É possível apontar como exemplo a métrica chamada de BLEU Score que avalia sentenças traduzidas entre idiomas por inteligência artificial, portanto, o mesmo conceito pode ser um ponto de partida para desenvolvimento de uma métrica avaliativa para músicas.

REFERÊNCIAS

ALBAWI, Saad e MOHAMMED, Tareq Abed.

Understanding of a Convolutional Neural Network – Turquia, Antalya, 2017.

BERCHMANS, Tommy. A música do Filme – Tudo o que você queria saber sobre música de cinema – São Paulo: Escrituras, 2006.

BORJI, Ali. Pros and Cons of GAN Evaluation Measures – Revista Computer Vision and Image Understanding, Volume 179, Páginas 41-65, 2018.

BULLINARIA, John A. Recurrent Neural Networks Neural Computation Lecture 12 – Escola de Ciência da computação, Universidade de Birmingham, Reino Unido, 2015.

BRITZ, Denny. Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano. Sitio WildML, 2015. Disponível em: <http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/>. Acesso em abril de 2019.

CALDAS, Waldenyr. A cultura político-musical brasileira – São Paulo: Musa, 2005.

COCA, Andrés E.; Romero, Roseli A. F. e Zhao, Liang. Generation of composed musical structures through recurrent neural networks based on chaotic inspiration – International Joint Conference on Neural Networks (IJCNN), San Jose, Califórnia, EUA, de 31 julho à 5 de agosto, 2011.

CORRÊA, Débora Cristina. Sistema baseado em redes neurais para composição musical assistida por computador – Universidade Federal de São Carlos, 2008.

CHEN, Chun-Chi J. e MIIKKULAINEN, Risto. Creating Melodies with Evolving Recurrent Neural Networks – 2001 International Joint Conference on Neural Networks, Washington, DC, 2001.

CHO, Kyunghyun; VAN MERRIËNBOER, Bart; GULCEHRE, Caglar; BOUGARES, Fethi; SCHWENK, Holger; BAHDANAU, Dzmitry e BENGIO, Yoshua. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation – 2014.

DONG, Hao-Wen; HSIAO, Wen-Yi; YANG, Li-Chia e YI-HSUAN, Yang. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment – Centro de pesquisa da tecnologia da informação e inovação, Academia Sinica, Taipei, Taiwan, novembro, 2017.

DRAKOS, Georgios. What is a Recurrent Neural Networks (RNNS) and Gated Recurrent Unit (GRUS) – Sitio towards data Science, 2019. Disponível em: < https://towardsdatascience.com/what-is-a-recurrent-nns-and-gated-recurrent-unit-grus-ea71d2a05a69>. Acesso em novembro de 2019.

FONSECA, Eder. “A Bossa Nova não tem idade” Toquinho – Cantor, compositor e violonista - Sitio Panorama Mercantil, 2016. Disponível em: <http://www.panoramamercantil.com.br/a-bossa-nova-nao-tem-idade-toquinho-cantor-compositor-e-violonista/>. Acesso em abril de 2019.

GOODFELLOW, Ian J.; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron e BENGIO, Yoshua. Generative Adversarial Nets – Département d’informatique et de recherche opérationnelle, Universidade de Montreal, Montreal, Canadá, 2014.

HUANG, Cheng-Zhi Anna; COOIJMANS, Tim; ROBERTS, Adam; COURVILLE, Aaron e ECK, Douglas. M. Counterpoint by Convolution. – 18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017.

HOCHREITER, Sepp e SCHMIDHUBER, Jurgen. Long Short-Term Memory – Neural Computation, dezembro de 1997.

LEURDIJK, Andra e NIEUWENHUIS, Ottilie. Statistical, Ecosystems and Competitiveness Analysis of the Media and ContentIndustries. The Music Industry – Comissão Europeia, Joint Research Centre, Instituto para a prospecção de estudos tecnológicos, Escritório de Publicações da União Europeia, Luxemburgo, 2012.

KARPATHY, Andrej e JOHNSON, Justin. Convolutional Neural Networks (CNNs / ConvNets) – Universidade de Stanford, Ciência da Computação turma CS231n, 2018. Disponível em: < http://cs231n.github.io/convolutional-networks/>. Acesso em setembro de 2019.

KRIEGESKORTE, Nikolaus. Deep neural networks: a new framework for modelling biological vision and brain information processing – Medical Research Council Cognition and Brain Sciences Unit, Cambridge, Reino Unido, 2015.

KULESZ, Octavio. Culture, platforms and machines: the impact of artificial intelligence on the diversity of cultural expressions – Intergovernmental Committee for the Protection and Promotion of the Diversity of Cultural Expressions, UNESCO Headquarters, Paris, França, dezembro de 2018.

OLAH, Christopher. Understanding LSTMs – Blog colah's blog, 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em abril de 2019.

MACHADO, Elton Leandro. Harmonia na bossa nova: um mapeamento da produção científica – Escola de Música e Belas Artes do Paraná. Curitiba, 2009.

MARQUES, Douglas L. e MARINO, Luciana N. Minha terra tem palmeiras Imagens do Brasil na bossa nova – Programa de Pós-Graduação em Letras Universidade Federal de Juiz de Fora, Juiz de Fora, Rio de Janeiro, Brasil, 2011.

MICROSOFT. Embarrassingly Parallel Image Classification, Using Cognitive Toolkit and TensorFlow on Azure HDInsight Spark – Sitio Microsoft Machine Learning Blog, 2017. Disponível em: <https://blogs.technet.microsoft.com/machinelearning/2017/04/12/embarrassingly-parallel-image-classification-using-cognitive-toolkit-tensorflow-on-azure-hdinsight-spark/>. Acesso em setembro de 2019.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg e DEAN, Jeffrey. Conditional Image Generation with PixelCNN Decoders – Universidade de Cornell, EUA, junho de 2016.

NAYEBI, Aran e VITELLI, Matt. GRUV: Algorithmic Music Generation using Recurrent Neural Networks – Departamento de Ciência da computação, Universidade de Stanford, Califórnia, EUA, 2015.

PAYNE, Christine. Clara: Generating Polyphonic and Multi-Instrument Music Using an AWD-LSTM Architecture – OpenAI Scholars Program, 2018.

PONTI, Moacir A. e COSTA, Gabriel B. Paranhos da. Como funciona o Deep Learning – Simpósio Brasileiro de Computação (SBC), Tópicos em Gerenciamento de Dados e Informações, 2017.

RADFORD, Alec; METZ, Luke e CHINTALA, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks – Universidade de Cornell, EUA, novembro, 2015.

ROCCA, Baptiste. Understanding Generative Adversarial Networks (Gans) – Sitio towards data Science, 2019. Disponível em: <https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29>. Acesso em abril de 2019

SIMONETTA, Federico; CARNOVALINI, Filippo; ORIO, Nicola e RODÀ, Antonio. Symbolic Music Similarity through a Graph-Based Representation – Wrexham, Reino Unido, 2018.

SUMA, Greg. Image Generator, Drawing Cartoons with Generative Adversarial Networks – Sitio towards data Science, 2019. Disponível em: < https://towardsdatascience.com/image-generator-drawing-cartoons-with-generative-adversarial-networks-45e814ca9b6b>. Acesso em novembro de 2019.

OORD, Aaron van den; KALCHBRENNER, Nal; VINYALS, Oriol; ESPEHOLT, Lasse; GRAVES, Alex e KAVUKCUOGLU, Koray. Conditional Image Generation with PixelCNN Decoders – Universidade de Cornell, EUA, junho de 2016.

YANG, Li-Chia; CHOU, Szu-Yu e YI-HSUAN, Yang. MIDINET: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation.– 18° International Society for Music Information Retrieval Conference, Suzhou, China, 2017.

TODD, Peter. M. A connectionist approach to algorithmic composition. – Computer Music Journal, volume 13 N°4, páginas 27–43, 1989.

SUMA, Greg. Image Generator, Drawing Cartoons with Generative Adversarial Networks – Sitio towards data Science, 2019. Disponível em: < https://towardsdatascience.com/image-generator-drawing-cartoons-with-generative-adversarial-networks-45e814ca9b6b>. Acesso em novembro de 2019.

ANÁLISE DA HISTÓRIA DE CRIPTOMOEDAS PARA FINS DE PREMEDITAR FUTURAS COTAÇÕES: ESTUDO DE CASO EM MACHINE LERANING E REDES NEURAIS

Robson Oliveira Batista

Ricardo José Menezes Maia

RESUMO

Com o atual cenário do mercado de criptomoedas em expansão e o alto volume de investidores, seja através de mineração, compra e venda ou até mesmo games "play-to-earn". Faz-se necessário de realizar um estudo mais aprofundado da volatilidade das moedas digitais. Uma forma de abrandar essa volatilidade, é aplicar modelos preditivos que fazem uso de inteligência artificial computacional para analisar futuros investimentos no mercado monetários de criptomoedas. O presente estudo aplica a técnica de Redes Neurais Recorrentes para realizar uma análise de aprendizado através do histórico de cotações diárias de quatro diferentes tipos de criptomoedas. Utilizou-se um sistema de API (Interface de Programação de Aplicação) para conectar à base de dados cryptocompare e coletar os dados das moedas Dogecoin, Litecoin, Ethereum e Bitcoin. Para cada moeda foram utilizados diferentes parâmetros e períodos históricos para melhor atingir a taxa de aprendizagem. Entre os 4 modelos testados, o Bitcoin apresentou os melhores resultados, com total de 2000 cotações analisadas. Apresentando o estudo preditivo com menos fidedigno, o Ethereum sofreu a análise de 1500 cotações.

Palavras-chave: Criptomoeda. Deep Learning. Cotação. Mercado Monetário. Redes Neurais. LSTM.

1 INTRODUÇÃO

Devido aos imparáveis avanços tecnológicos, o mundo está sendo guiado pelo volume crescente de dados e sua disponibilidade, porém a segurança na transação de informações sempre foi uma preocupação (CORRÊA; MARÇAL;

FLACH, 2019). As compras realizadas através da internet dependem de instituições financeiras para atuar como intermediador. Apesar de atender a maioria das necessidades do consumidor, o modelo existente ainda carece de requisitos de confiabilidade (NAKAMOTO, 2008).

De modo a resolver o problema da relação de confiança entre transações comerciais através da web, Satoshi Nakamoto criou um sistema baseado em caixa eletrônico Peer-to-Peer com tecnologia de criptografia, permitindo que consumidores e compradores tenham relação comercial sem o intermédio de uma instituição terceira. Tal tecnologia foi denominada blockchain ou cadeia de blocos (NAKAMOTO, 2008). O sistema criado por Nakamoto deu origem à primeira criptomoeda, a qual veio ser conhecida mundialmente como Bitcoin.

Apesar de trazer maior confiabilidade à mercê de suas transações, o principal problema do Bitcoin e das criptomoedas, em geral, é a sua volatilidade. Trazendo um grande impacto ao que se refere às estratégias de comércio e tomada de decisão dos investidores, tornando a moeda pouco confiável. Cria-se a necessidade de uma ferramenta capaz de trazer predições referentes ao valor cambiário do mercado de criptomoedas, com a finalidade de auxiliar investidores em futuras decisões (ALMEIDA, 2019).

Com o intuito de trazer um maior conhecimento sobre o sistema cambiário de criptomoedas, a presente pesquisa utilizou técnicas de Machine Learning, com uma análise história das principais criptomoedas emergentes no mercado atual, de modo a identificar a possibilidades de predições do mercado monetário de moedas virtuais

2 REFERENCIAL TEÓRICO

A presente seção encontra-se dividida em dois momentos. Sendo o primeiro à mercê do contexto histórico referente ao Blockchain, com a exploração do tema de criptomoeda e Bitcoin, de modo a trazer um maior conhecimento ao assunto abordado na pesquisa. No segundo período serão tratados diferentes modelos de análise preditiva e os benefícios que a aprendizagem de máquina pode trazer ao mercado financeiro de criptomoedas.

2.1 Blockchain e Criptomoedas

Em um fórum de discussões, em primeiro de novembro de 2018, Satoshi Nakamoto apresentou de forma online seu artigo que tratava de uma forma eletrônica de dinheiro. Do qual permitia transações sem intermédio de uma instituição financeira, tornando os cursos das transações mais baratas (BATISTA; ALVES, 2021). O estudo que Bitcoin: A Peer-to-Peer Eletronic Cash System, conseguiu trazer mais uma tentativa de transmissão de valores dissociados dos governos e sistemas bancários (BUENO; AGUIAR, 2020).

Através de técnica de criptomoeda blockchain, as transações bitcoins remetem a um estudo do qual somente o remetente e o receptor possuem acesso às informações, mesmo que seja extraviado a informação seu conteúdo não poderá ser acessado. Desenvolvido através de sistemas para de descentralização de dados, conseguindo compartilhar os registros difusamente entre os usuários, adicionando seus dados em um novo bloco de registros, garantindo a confiança e segurança de cada operação (BUENO; AGUIAR, 2020).

Um blockchain é um banco de dados distribuído de registros, ou razão pública, de todas as transações ou eventos digitais que foram concluídos e trocados pelos participantes. Cada transação no livro-razão público é verificada duas vezes pela maioria dos participantes do sistema. Os dados não podem ser excluídos depois de inseridos. Cada transação na arquitetura do blockchain, é um nó de computadores (AYYAPPAN, 2020).

Composto por vários nós de computador, a tecnologia blockchain mantém uma cópia do livro razão (dados de entrada e saída) entre o destinatário e seu remetente, ambos se comunicam entre si, mantendo um consenso sobre o conteúdo transferido, sem a necessidade de uma autoridade central para coordenar e validar as transações (AYYAPPAN, 2020). Os dados são blocos conectados por funções hash (funções matemáticas de fácil cálculo a partir do valor de entrada, porém quase impossível de identificar o valor inicial após alterado pela função). Toda vez que um bloco novo é conectado, ele é implementado a função hash, originando uma nova blockchain e trazendo a configuração de imutabilidade, qualquer tentativa de

modificação será identificado, alterando somente os valores finais da função hash (SOBRINHO, 2019).

O blockchain é armazenado em múltiplos servidores, computadores, de modo que cada um pode ter a cópia completa de todos os registros na rede Peer-to-Peer (P2), que estão sincronizados de modo tal que a inserção de novos registros requer a validação da rede que estão sincronizados por determinados mecanismos de consensos (SOBRINHO, 2019).

Ao longo do tempo foi introduzido diversos mecanismos de consenso, sendo o mecanismo pioneiro o "Prova de Trabalho" ou Proof of Word (PoW). Consistem em resolver problemas matemáticos através de esforço computacional, do qual comprova que a pessoa como minerador (miner), tenha realizado as exigências necessárias para inserir novos dados em uma blockchain, recebendo como gratificação por seu trabalho Bitcoins. Com o risco de evitar o monopólio da mineração de Bitcoins por parte de grandes empresas, foi criado em 2012 o Mecanismo de Consenso Prova de Participação (PoS), esse mecanismo usa como critério para você ser inserido em uma blockchain a quantidade de criptomoedas que você possui, deixando de pender o poder computacional, passando a ser remunerados por taxas de transações e não mais por recompensa de blocos.

No ano seguinte veio o advento do Mecanismo de consenso Prova de Participação Delegada, do inglês Delegated Proof of Stake (DPoS). Permite que acionistas e usuários de sistemas confiram os dados inseridos na blockchain. A ideia da criação do mecanismo Bitshares é permitir que os detentores de criptomoedas, mesmo que pequenos, possam delegar seu direito de transição para outra pessoa. Acabando com privilégio de quem retém mais criptomoedas e permitindo a inserção de blocos independente do quanto possui em sua carteira.

Em 2014 veio a emersão dos chamados contratos inteligentes, que tiveram a sua relevância com o advento do Ethereum em 2014, porém o termo foi apresentado pela primeira vez em 1994 por Nick Szabo. Deu margem para que minimize exceções maliciosas e acidentais, reduzindo a necessidade de intermediários confiáveis. Independente da plataforma que utilize tal mecanismo, recorrer à característica de oráculos. Essa característica confirma se as fontes de informações

ou dados são confiáveis e se são elegíveis para serem acessados através dos contratos inteligentes (SOBRINHO, 2019).

2.2 Redes Neurais Artificiais

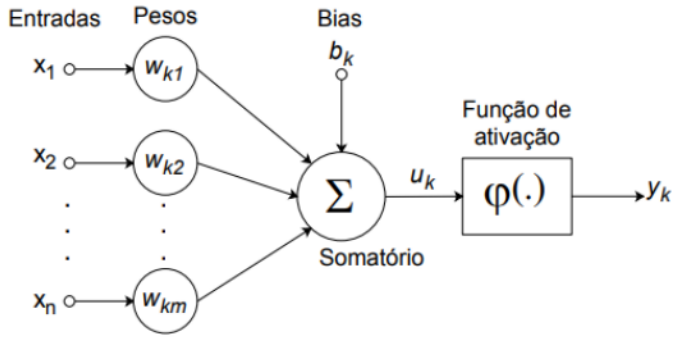
Sistema Nervoso Central, capaz de organizar vértices em camadas, como neurônios biológicos. Do qual interagem entre si de forma não linear e calculam seu próprio nível de ativação conforme vão recebendo novos sinais (COELHO; SILVA, 2017). No corpo humano, esses sinais são transmitidos através das sinapses, do qual medem a interação dos neurônios, convertendo um sinal elétrico pré-sináptico para um sinal químico e volta um sinal elétrico pós-sináptico. Resultando em uma sinapse química, podendo aumentar ou diminuir o potencial elétrico do corpo celular (GARCIA, 2021). Como no sistema biológico, as redes neurais artificiais possuem sinapses, elas codificam a influência de uma unidade sobre a outra para receber uma conexão apropriada, diminuindo e aumentando a influência do peso associado a vértices (COELHO; SILVA, 2017).

Sistema Nervoso Central, capaz de organizar vértices em camadas, como neurônios biológicos. Do qual interagem entre si de forma não linear e calculam seu próprio nível de ativação conforme vão recebendo novos sinais (COELHO; SILVA, 2017). No corpo humano, esses sinais são transmitidos através das sinapses, do qual medem a interação dos neurônios, convertendo um sinal elétrico pré-sináptico para um sinal químico e volta um sinal elétrico pós-sináptico. Resultando em uma sinapse química, podendo aumentar ou diminuir o potencial elétrico do corpo celular (GARCIA, 2021). Como no sistema biológico, a redes neurais artificiais possuem sinapses, elas codificam a influência de uma unidade sobre a outra para receber uma conexão apropriada, diminuindo e aumentando a influência do peso associado a vértices (COELHO; SILVA, 2017).

Os neurônios artificiais são capazes de alimentar a camada posterior, criando uma categoria de memórias a curto prazo, sendo a mais comum o modelo de Feedforward Neural Network. Tal tipo repassa as informações linearmente, ou seja, na mesma direção, até que chega na última camada do neurônio. Também possui diversas formas de se conectar os neurônios, uma delas é a Recurrent Neural

Networks ou Redes Neurais Recorrentes (RNN), essa técnica faz com que as informações percorrem diferentes direções, alimentando o próximo neurônio e, ao mesmo tempo se realimentando (retroalimentar), gerando a memória de curto prazo. Para um melhor entendimento seguir-se-á a premissa da Figura 1, onde os sinais de eletrônicos enviados aos axiomas de outra neurônio artificial são recebidos pela entrada. Resultando na soma dos sinais, determinado se o axônio emitirá ou não um sinal de saída para o seu sucessor, limitando a entrada e a saída da rede neural (GARCIA, 2021).

FIGURA 1 — Neurônio Artificial

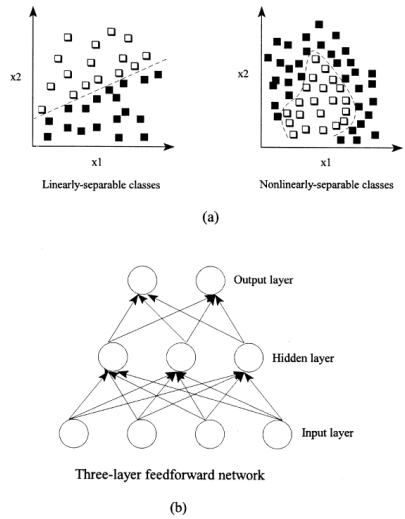


Fonte: Adaptado de Garcia (2021)

O perceptron traz a capacidade de ser treinado em um conjunto de exemplos utilizando regras de aprendizagem. Seus pesos são alterados conforme a proporção de erros calculados entre a saída e a solução para cada exemplo processado. No caso é uma função de todos os pesos em um hiperplano complexo, multidimensional e irregular, ou seja, composto por muitos picos entre alto e baixo. Para solucionar tal problema, utiliza-se uma técnica que recorre a uma busca especializada, de modo a obter o mínimo global. Tais regras só podem ser implementadas precisamente com classes lineares capazes de serem separadas em um hiperplano, conforme demonstrado na figura 02. A ilustração (a) mostra como ocorre a separação linear das classes versus a separação não linear, a fim de ilustrar o entendimento da necessidade de utilização de MLP (Perceptron multicamadas). Conforme demonstrado na ilustração (B) da figura 02, as camadas intermediárias não interagem com ambiente externo, se denominando camadas ocultas e nós ocultos. Os

neurônios ocultos processam as informações recebidas pelos nós de entrada e os transporta para a camada de saída, resolvendo o problema de solução para problemas não lineares (BASHEER; HAJMEER, 2001).

FIGURA 02 – Perceptron



Fonte: IA Basheer, M. Hajmeer / Journal of Microbiological Methods 43 (2000) 3 -31

Seguindo a mesma ideia de pensamento de Almeida Charllon eu seu trabalho "predição de séries temporais aplicadas ao mercado de criptomoedas" utilizarei redes neurais artificiais para auxiliar na realização de uma melhor predição dos valores de Bitcoin, fazendo uso de LSTM (Long Short Term Memory) para trazer um melhor conhecimento do momento que se deve realizar a compra e a venda da criptomoeda. Mediante esse pensamento não podemos deixar de falar sobre Deep Learning.

2.2.1 Deep Learning

Aprendizagem profunda, mais conhecida como Deep Learning, é uma das vertentes da inteligência artificial. Ela possibilita que uma rede através da experiência e o entenda a partir de uma hierarquia de conceitos, relacionando cada dado aprendido com o conceito mais simples. Essa técnica traz um maior sucesso para previsão do mercado de ações, podendo ser aplicado para predição de valores

de criptomoedas. Seu grande diferencial para aprendizagem de máquinas mais "rasas" é o fato de utilizar funções estáticas para obtenção e cálculos dos resultados almejados de forma direta (ALMEIDA, 2019).

2.2.2 Redes neurais recorrentes

Sendo extremamente úteis para processamento de dados sequenciais, como som, dados de séries temporais ou linguagem natural, porém a sua grande vantagem sobre as redes feedforward é o fato de possuírem um loop de feedback. Tal loop possibilita que as redes neurais recorrentes possam determinar a saída de cada passo seja alimentada de volta à rede para afetar o resultado subsequente, dispensado da necessidade de um conjunto de entradas-saída por vez e permitindo a criação de modelos dinâmicos. Ou seja, mesmo que aprendido determinado padrão durante a sua fase de treinamento, tal modelo pode sofrer alterações em sua de testes, aceitando novos exemplos com diferentes argumentos dos da fase de treinamento e, ao mesmo tempo, processar a análise preditiva com precisão (PASSOS, 2021).

Vários estudos relataram os resultados da previsão da taxa de câmbio Bitcoin usando modelos ARIMA (auto-regressivo integrado de médias móveis) clássicos e usando diferentes métodos de aprendizado de máquina, como Random Forest (RF), Logistic Regression (LR) e Linear Discriminant Analysis (LDA), e Memória de longo prazo (LSTM). Os resultados dessas análises mostraram que os modelos que contavam com o treinamento mostraram-se mais adequados para prever tanto os preços das criptomoedas quanto sua volatilidade (DERBENTSEV).

Passos explica que as redes neurais recorrentes têm grandes problemas em lembrar de dados ao longo do tempo, em 1997 Hochreiter e Schmidhuber criaram as chamadas Long Short-Term Memory (LSTM) para suprir esse problema. Essa variação de RNN permite armazenar vários estados, devido a suas células de memórias, diferente dos Padrões RNN que possuem somente 1 camada, o LSTM possui 4 camadas de redes neurais, do qual interagem de maneira particular (GARCIA, 2021). As células de memória são formadas por "gates" (portas), também conhecidas como portas de entrada e saída (input gate e output gate). Em 1999 foi adicionado o Forget gate por Gers, Schmidhuber e Cummins, esse "gate" permite

que a LSTM possa dispensar determinados dados da memória quando se mostrarem inúteis para o modelo, podendo ser de forma gradual ou instantânea. Como podemos observar na Figura 04, Onde:

Xt: Vetor de entrada

$Ct-1$: Estado antigo da célula

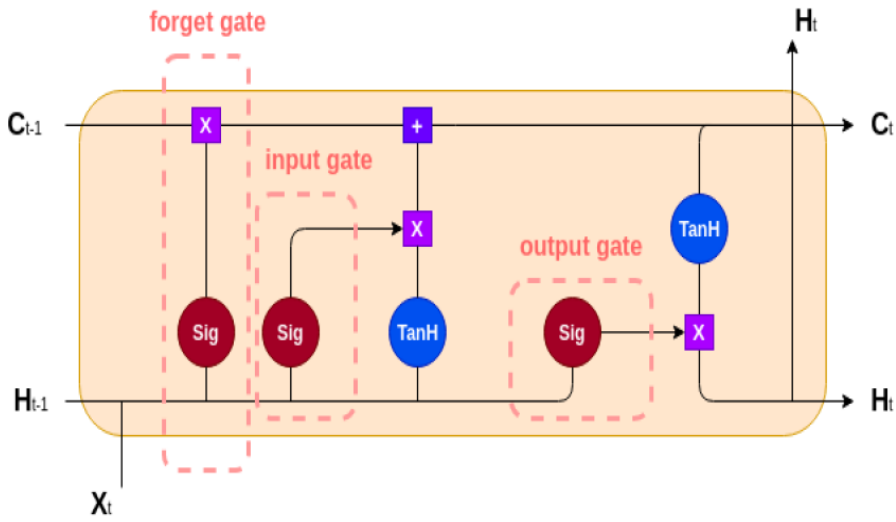
Ct: Estado atualizado da célula

$Ht-1$: Vetor de saída do último estado

Ht: Vetor de saída do estado atual

Todos os portões são ativados através de uma função sigmóide, retornando números 0 e 1. Antes de passar pelos portões, as funções de vetores de Entrada Xt e o vetor de saída do último estado $Ht-1$ são concatenados. Por meio do retorno estabelecido pela função sigmóide, será deliberado qual dado será mantido e qual descartado. Qual o resultado estive mais próximo de 0, é sinal de que esse dado será descartado e quanto mais próximo de 1, indica qual dado permanecerá. O vetor ft retornado pelo gate será multiplicado pelo vetor Antigo da Célula Ct-1, existindo a chance desse vetor ser esquecido, visto que foi multiplicado pelo vetor mais próximo de 0. Em seguida a combinação Ht-1 e Xt irão passar pelo input gate, determinando qual a função sigmóide que irá determinar, qual o valor a será atualizado. Priorizando o número mais próximo de 1. Também passando pela função hiperbólica para regular os vetores entre -1 e 1. Multiplica-se a saída da função sigmóide pela tangente hiperbólica, determinado qual é mais importante para se manter na memória. Somasse a saída do input gate ao antigo estado da célula, do qual havia sido multiplicado pela saída do forget gate, dando-lhe o resultado do novo estado de Ct (PASSOS, 2021).

Figura 04 - Gates LSTM Adaptado



Fonte: Adaptado de (PASSOS, 2021)

Para finalizar, será passado pelo gate de saída a combinação de Ht -1 e Xt, passando pela função sigmoide e multiplicando o novo estado da célula LSTM pelo retorno da função Ht-1 e Xt, resultando no Ct. Gerando um novo estado oculto de celular Ht dando-lhe a possibilidade de ser aproveitada para novas predições, passado juntamente a célula Ct para as próximas interações da RNN (PASSOS, 2021).

3 EXPERIMENTO

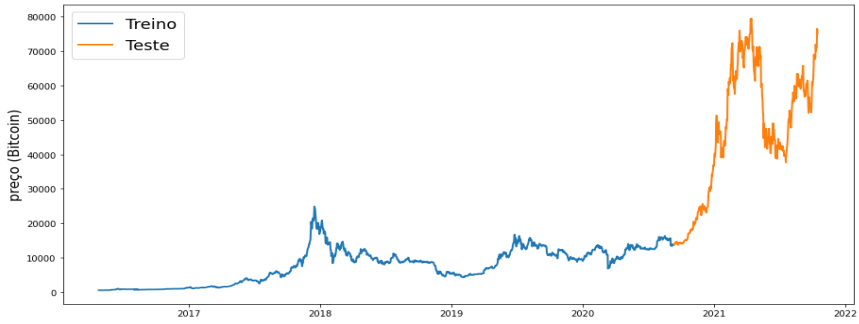
Para o presente estudo serão utilizados os dados disponíveis através da API fornecida pela Cryptocompore. O modelo utilizado realizará previsões a partir da técnica de LSTM para diferentes tipos de criptomoedas. Iniciaremos com Bitcoin (BTC), passaremos para Ethereum (ETH), em seguida Litocin (LTC) e finalizamos com Dogecoin (DOGE).

3.1 Bitcoin

Para o presente estudo serão utilizados os dados disponíveis através da API fornecida pela Cryptocompore. A massa de dados referente às datas de 25 de abril de 2016 até a cotação do dia 16 de outubro de 2021. Para rodar o modelo foi separado

20% dos dados para teste, ficando 70% para treinar o modelo, conforme explicitado na figura 05.

Figura 05 – Dados de Teste e Treno



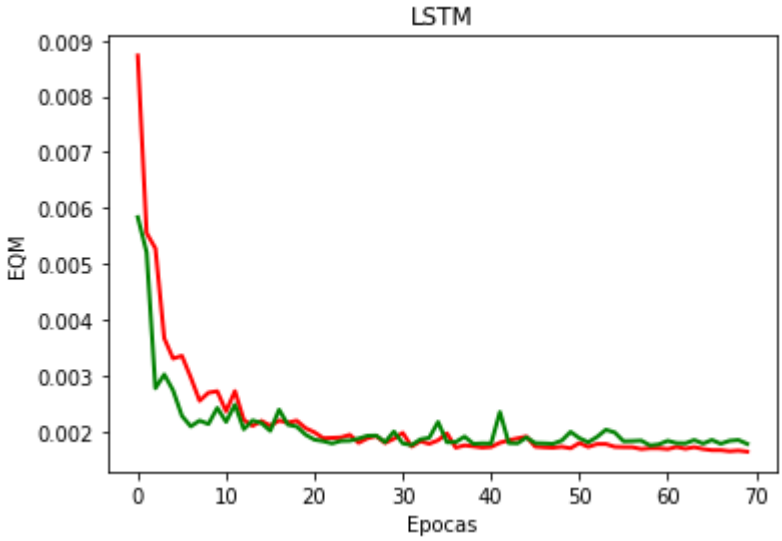
Fonte – Elaborado pelo autor do trabalho

Para o treinamento do modelo foi utilizado um total de cem (100) neurônios na camada de LSTM, sendo vinte por cento para teste, com uma camada de Dropout de 20% e cinquenta épocas. Para o processo de optimização foi usado o ADAM, trata de um algoritmo que calcula as taxas de aprendizagem adaptadas para cada parâmetro. Incorpora o conhecimento dos dados obtidos através da observação entre a interação anterior para executar um aprendizado "raro", porém predizíeis (RIBEIRO, 2020). A camada de Dropout é responsável por realizar a regularização da RNN, reduz a co-adaptações complexas entre os neurônios, forçando aprendizagem mais robustos, evitando a perda de qualquer evidência individual (DEEPLEARNINGBOOK, 2021).

O Erro quadrado médio (EQM) ou do inglês Mean Squared Erro (MSE) pretende encontrar os pesos de determinada função, fazendo a rede neural se aproximar da melhor maneira de mapear os valores de todos os exemplos utilizados. Ou seja, corrige os pesos da rede quando a saída difere do desejado (GERS; SCHMIDHUBER; CUMMINS, 2000), gerando uma análise utilizando sistema de LSTM. Disposto uma janela de 10 dias de forma aleatória e normalizado os dados para base -1, sendo o primeiro dado de entrada e os demais valores representam a alteração referente ao primeiro valor, prevendo a alteração do preço, do qual apresenta a curva de EQM (Erro Quadrático Médio). A partir da figura 06,

identificamos um ótimo casamento entre os resultados simulados obtidos pelo modelo proposto.

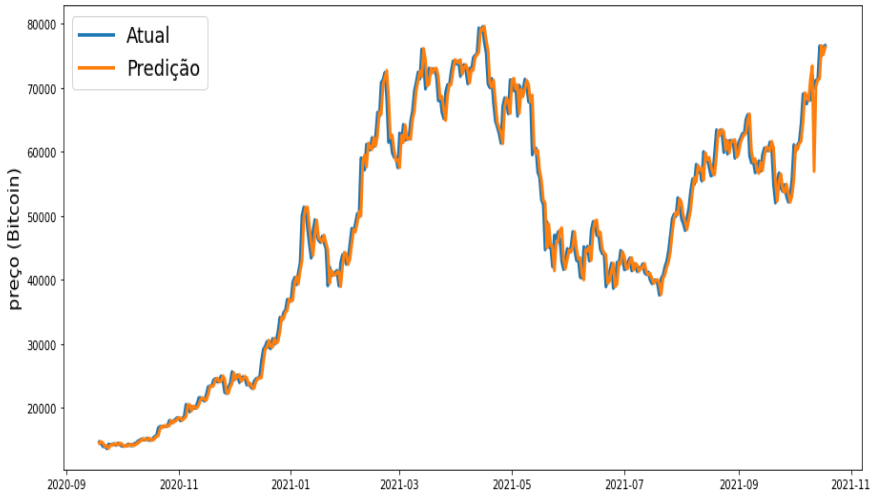
Figura 06 – LSTM (BitCoin)



Fonte – Elaborado pelo autor do trabalho

O resultado trouxe como Erro Médio Absoluto de 0.03064860892477881, R2 (Coeficiente de Determinação) de 0.7728339351674723 e Erro Médio Quadrático (MAE) de 0.0018275577050349272. Analisando o gráfico de predições comparando o Preço atual do mercado de Bitcoin e o Preço Previstos podemos identificar que gráfico de aprendizagem é similar ao gráfico de predições, conforme exposto na figura 07.

Figura 07 – Predição de Bitcoin

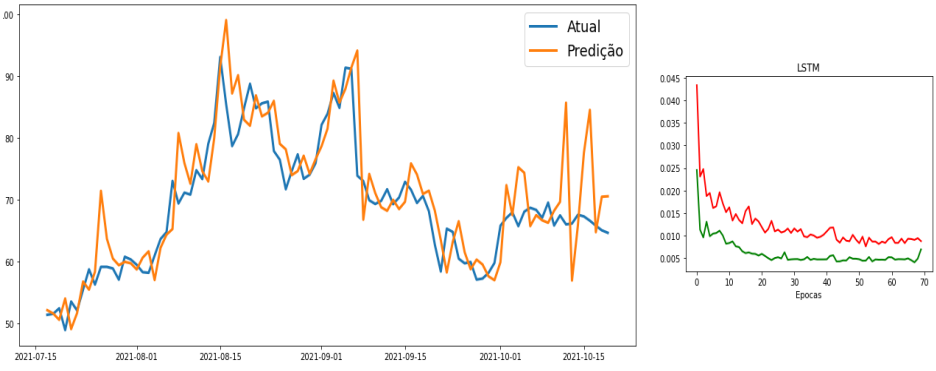


Fonte – Elaborado pelo autor do trabalho

3.2 Ethereum

Para o nosso segundo modelo, realizaremos o teste com a moeda virtual Ethereum. Usaremos os dados de 10-09-2017 a 19-10-2021, compondo um total de 1500 cotações. Utilizaremos como parâmetro 20% dos dados para treino e 80% para teste. Separado 30% dos dados para teste e 70% para treino, composto por 70 neurônios, 10 janelas, divididas em lotes de 40. Utilizaremos na camada de dropout um total de 20%.

Figura 08 — Predição Ethereum



Fonte – Elaborado pelo autor do trabalho

Analisando a figura 08, foi observado que a curva de aprendizagem do Ethereum conseguiu acompanhar a tendência das predições, porém após rodar diversas vezes o modelo, foi constatado que o problema não estava nas épocas e nem nas camadas LSTM, mas sim nos dados. Devido ao grande pico da moeda em Abril de 2021, do qual a modela valia em torno R$ 40 e passou a valer mais R$ 160 a predição apresentou uma grande falha. Por isso foi reduzido a análise de 1500 dados para as últimas 500 cotações, obtendo os dados referente a figura 08. Os dados ainda são insuficientes para realizar uma análise e a predição foi inconclusiva.

3.3 Litecoin

Para o modelo do Litecoin faz-se uma análise de quinhentos (500) dias. Respectivamente entre os dias 05 de junho de 2020 e 18 de outubro de 2021. Conforme demonstrado na figura 09.

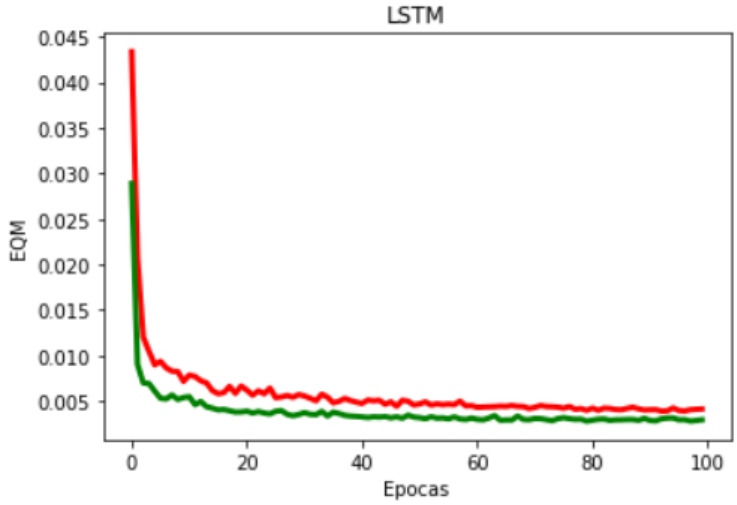
Figura 09 — Train, Test (Litecoin)

Fonte – Elaborado pelo autor do trabalho

O treinamento da LSTM foi realizado com 100 neurónios, sendo 20% dedicado a camada de Dropout. Decidir deixar esse parâmetro para treinar os demais modelos, pois foi a que se apresentou com melhor comportamento. O modelo foi treinado com 100 épocas, porém aumentamos o tamanho do lote de 32 para 40, do qual não apresentou diferença na curva de aprendizagem. Como podemos observar nas figuras 10 e 11, a curva de aprendizagem ficou distante da real, mostrando-se pouco fiel. Para haver uma melhor leitura de tais parâmetros seria necessário o

treinamento com um número maior de épocas, visto que podemos observar que o a curva de aprendizagem começa a se aproximar a partir da época 40, porém com uma taxa muito baixa.

Figura 10



Fonte – Elaborado pelo autor do trabalho

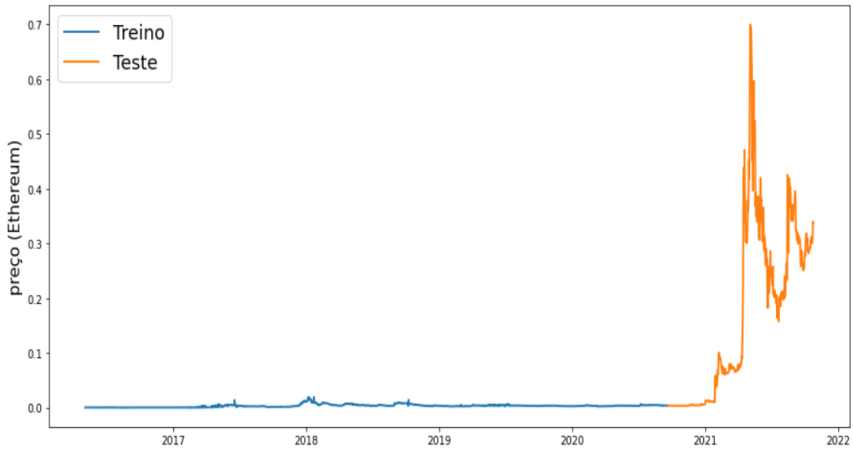
Fonte – Elaborado pelo autor do trabalho

3.4 Dogecoin

Para o último modelo foi analisada a criptomoeda Dogecoin, também conhecida como “Doge”. De acordo com o artigo publicado no site oficinadanet. Esta criptomoeda é considerada a sexta melhor criptomoeda para ser minerada. Sua criação teve origem em 2013 pelos engenheiros de software Billy Markus e Jackson Palmer. Similar ao Bitcoin e ao Ethereum, essa moeda virtual possui a sua base em tecnologia de Blockchain.

Voltando ao foco do nosso experimento, utilizamos como massa de dados as cotações entre 4 de maio de 2015 e 25 de outubro de 2021, somando um total de duas mil linhas. Para o processo de teste, foi separado vinte por centro dessa massa de dados, conforme demonstrado na figura 11.

Figura 11 – Treino e teste Dogecoin



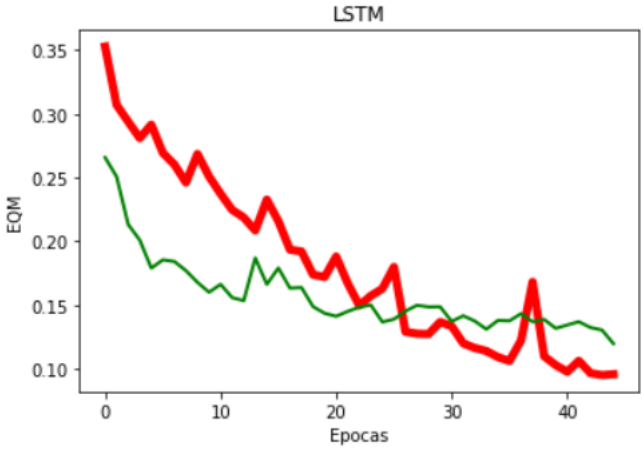
Fonte – Elaborado pelo autor do trabalho

Para realizar o processo de geração do modelo LSTM, repetimos a mesma porcentagem para camada de Dropout, com setenta (70) neurônios, quarenta e cinco (45) épocas, também foi especificado o tamanho do lote de forma aleatório em trinta e cinco (35) e finalizado com processo de otimização ADAM (mesmo usado nos testes anteriores).

Os resultados do modelo em LSTM apresentou 0.11958943883590076 para a média de erro quadrático, com R2 score de 0.7265463587015863 e o Erro médio

absoluto de 0.12575665229672536. Podemos observar na figura 12 que as perdas utilizadas no treino (Gráfico em vermelho) e as perdas utilizadas para validação do modelo começam a se alinharem às épocas 20 e 30. Isso ajuda a demonstrar a curva de aprendizagem que o modelo necessita para começar a aprender as predições futuras.

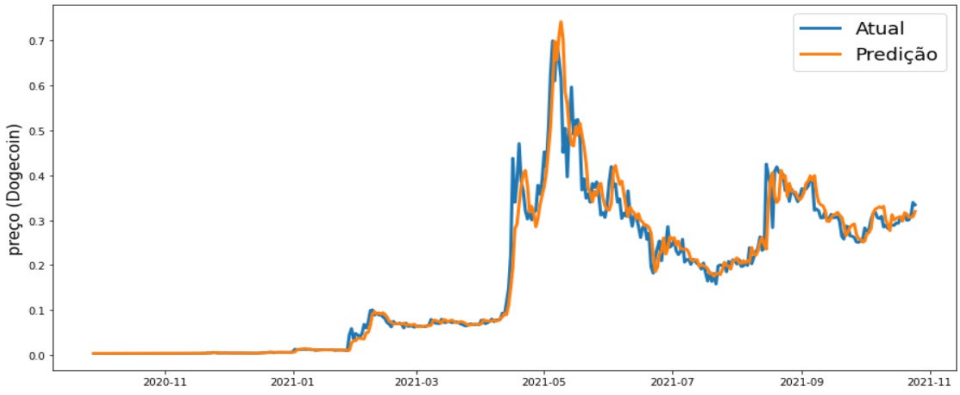
Figura 12 – Perdas Dogecoin



Fonte – Elaborado pelo autor do trabalho

Finalizando nosso estudo, observamos na figura 13 que o modelo conseguiu acompanhar as previsões, mesmo que de forma não precisa, o gráfico de predições nos mostra a tendência das curvas de valorização e decremento da criptomoeda.

Figura 13 — Predição de Dogecoin



Fonte – Elaborado pelo autor do trabalho

4 CONCLUSÃO

Identifiquei com base no referencial teórico que o sistema de redes neurais é uma forte ferramenta para estudos preditivos, do qual no presente estudo foi possível realizar com excelência a predição do mercado de câmbio de criptomoedas do tipo Bitcoin. Também pude notar em minhas pesquisas que vários autores utilizaram o sistema para realizar estudos similares para outras áreas ou categorias de mercados, como bolsa de valores.

Creio que essa ferramenta traz grande força e auxílio para investidores no mercado de criptomoeda, visto que se mostrou tão fidedigna aos dados atuais. Todavia, creio que somente essa categoria de análise não basta para realizar grandes investimos, pois o estudo cria especulação de valores futuros com base na curva de aprendizagem e o mercado de criptomoedas sofrem outros fatores externos que possam interferir abruptamente em seu valor.

Como exposto na Figura 05, do final de 2020 até o primeiro trimestre de 2021 foi onde a moeda sofreu maior crescimento e logo em seguida ela entrou em constante volatilidade. Creio que movido por fatores sociais, políticos e econômicos as criptomoedas têm seu valor alterado, como no mercado monetário. Torna-se necessário a verificação do mercado com outras ferramentas de predição para trazer uma melhor correlação entres os períodos de queda e baixa.

4.1 Trabalhos futuros

Identificamos que se consegue realizar um estudo de redes neurais para predizer preço de criptomoedas através do histórico de resultados do mercado monetário, porém não se conseguiu identificar o motivo de tais predições. De modo a trazer resultados mais confiáveis, sugiro para trabalhos futuros:

- Realizar um estudo de “sentimento” utilizando redes neurais para identificar fatores sociais que possam interferir no resultado da predição. Gerar um modelo e comparar os resultados junto ao LSTM.

- Faz uso de estudos de outros modelos de análise temporal como regressão linear para comparar resultados e uso de técnica para apontar se deve investir ou não na criptomoeda, como Radom Florests e Árvore de Decisão.

REFERÊNCIAS

As 10 melhores criptomoedas para minerar em casa em 2021. Disponível em: https://www.oficinadanet.com.br/criptomoedas/37225-criptomoedas-minerar-em-casa. Acessado em: 24 de nov. de 2021.

ALMEIDA, Charllon Lobo. Predição de séries temporais aplicada ao mercado de criptomoedas. 2019. 32 f. Monografia (Graduação em Ciência da Computação) - Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, 2019. Acessado em: 24 de nov. de 2021.

AYYAPPAN, Vasanthy. Blockchain for Data Science and its Business Applications.

BASHEER, Imad A.; HAJMEER, Maha. Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods, v. 43, n. 1, p. 3-31, 2000.

BATISTA, D. T.; ALVES, C. F. Análise do impacto do Bitcoin na eficiência de uma carteira diversificada para investidores brasileiros. RBGN - Revista Brasileira de Gestão de Negócios, [S. l.], v. 23, n. 2, p. 353–369, 2021. DOI: 10.7819/rbgn.v23i2.4098. Disponível em: https://rbgn.fecap.br/RBGN/article/view/4098. Acesso em: 27 setembro. 2021.

BUENO, Thiago Augusto; AGUIAR, Julio Cesar. Análise monetária do bitcoin. NOMOS, v. 40, n. 11, p. 59-73, jan/jun2020.

COELHO, Orlando Bisacchi; SILVEIRA, Ismar. Deep learning applied to learning analytics and educational data mining: A systematic literature review. In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). 2017. p. 143.

Como Funciona o Dropout? Disponível em: <https://www.deeplearningbook.com.br/capitulo-23-como-funciona-o-dropout/>. Acessado em 16 de outubro de 2021.

CORRÊA, Amanda Beatriz Nasatto; MARÇAL, Ronan Reis; FLACH, Leonardo. Previsibilidade no mercado de criptomoedas: uma modelagem autoregressiva com dados em painel do market cap. Revista de Contabilidade e Gestão Contemporânea, v. 2, n. 1, p. 51-62.

DERBENTSEV, Vasily et al. Forecasting cryptocurrency prices time series using machine learning approach. In: SHS Web of Conferences. EDP Sciences, 2019. p. 02001.

GARCIA, Vinicius de Souza Fialho. Séries temporais para predição de finanças no contexto de criptomoedas. 2021. 58 f. Monografia (Graduação em Engenharia de Computação) - Instituto de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade, 2021.

GERS, Felix A.; SCHMIDHUBER, Jürgen; CUMMINS, Fred. Learning to forget: Continual prediction with LSTM. Neural computation, v. 12, n. 10, p. 2451-2471, 2000.

MARETTI, Roberto Bruno Lemes. Simulação de negociações em instrumentos do índice IBOVESPA utilizando machine learning. 2019. 95 f. Dissertação (mestrado em Engenharia Elétrica e Computação) - Universidade Presbiteriana Mackenzie, São Paulo, 2019.

NAKAMOTO, Satoshi; A peer-to-peer electronic cash system. Bitcoin. –URL: https://bitcoin. org/bitcoin. pdf, v. 4, 2008.

LUIZ OTAVIO PASSOS. Uso de redes neurais recorrentes para previsão na Bovespa. 2021. 82 f. - Instituto de Ciência e Tecnologia, São José dos Campos, 2021

RIBEIRO, Athyrson Machado et al. Um Estudo Comparativo Entre Cinco Métodos de Otimização Aplicados Em Uma RNC Voltada ao Diagnóstico do Glaucoma. Revista de Sistemas e Computação-RSC, v. 10, n. 1, 2020.

SANTOS, J. A. A.; SPANCERSKI, J. S. S. Aplicação de redes neurais recorrentes na previsão de geração eólica. Revista Cereus, v. 13, n. 1, p. 217-227, 2021.

SOBRINHO, R. P. et al. Tecnologia Blockchain: inovação em Pagamentos por Serviços Ambientais. Estudo.

IDENTIFICAR AUTORIDADES POR MEIO DE RECONHECIMENTO FACIAL: USO DE TECNOLOGIA DE VISÃO COMPUTACIONAL COMO ALTERNATIVA PARA ANTIGO PROCESSO DE FOTOGRAMAS (CARÔMETRO)

Walner de Oliveira Pessoa

William Roberto Malvezzi

RESUMO

Dentro do contexto dos profissionais que atuam na atividade de Relações Governamentais, o reconhecimento de parlamentares em ambientes físicos é uma tarefa essencial. Esses profissionais exerce uma atividade que faz chegar aos parlamentares os interesses das categorias que os lobistas representam, ou seja, um dos processos que asseguram a democracia para nosso país. A forma tradicional utilizada de reconhecimento é feita por fotogramas impressos que o profissional tem que andar em sua mão para identificar (reconhecer) a autoridade parlamentar. Uma solução para essa tarefa seria utilizar técnicas de visão computacional para auxiliar a identificar, de maneira rápida e precisa, uma autoridade em um evento que circulam dezenas ou centenas de pessoas. Os estudos sobre visão computacional são recentes, nos anos 70 dois trabalha foram precursores - The Psychology of Computer Vision e A framework for representing knowledge. A partir de 2012 com a vitória do concurso de reconhecimento de imagem de computador ImageNet, Alex Krizhevsky da Universidade de Toronto com uma rede neural chamada AlexNet abriu as fronteiras para avanços de modelos de redes neurais e de hardware com arquiteturas em NVIDIA GPU (hoje o GPU está disponível para qualquer usuário da plataforma Colab da Google). Inovações como essas foram determinantes para tornar acessível para qualquer pessoa desenvolver soluções de reconhecimento facial em seus computadores domésticos. Este estudo técnico apresenta o uso de bibliotecas como Dlib e OpenCV que simplificam o desenvolvimento de aplicações para treinar um modelo com imagens de deputados federais, por exemplo, e oferecer uma ferramenta para que lobistas deixem de utilizar os fotogramas com a imagem e nome dos parlamentares (“carômetro”) impressos em papel e comecem a utilizar soluções tecnológicas de Deep Learning ao alcance de leigos.

Palavras-chave: Detecção facial, Inteligência Artificial, Deep Learning, OpenCV, Dlib, Encoding, Face Embeddings, Método CNN, Python, Método HOG, Lobby, Rotatividade, Defesa de interesse.

1 INTRODUÇÃO

No Brasil, segundo o site Congresso em Foco, existem 96 mil profissionais que atuam na atividade de Relações Governamentais (RelGovs). Essa classe tem o desafio de buscar um relacionamento junto às autoridades dos três Poderes da República diante de um cenário com elevado índice de rotatividade de cargos federal, estadual e municipal (LOPEZ,2020). Para identificar seus interlocutores, os RelGovs usam uma forma arcaica de fotograma no papel, chamado no jargão profissional de carômetro, para reconhecimento facial dentro de suas atividades profissionais diárias.

Os parlamentares têm mandatos de no mínimo de 4 anos, ou seja, após o primeiro ano os profissionais de Relações Governamentais já poderiam reconhecer essas autoridades com mais facilidades, porém pense na elevada rotatividade de autoridades governamentais dos três entes federados mudando constantemente. É um grande número, correto? É possível obter uma nova estratégia de negócio ao se aplicar as tecnologias de visão computacional em empresas de RelGov? Será que as soluções com reconhecimento facial usando redes neurais descrita nesse estudo técnico poderão proporcionar um diferencial de mercado para as empresas que aplicarem essa solução? Se o tempo é uma variável cada vez mais escassa, é preciso investir em processos ágeis e precisos na identificação de pessoas públicas para atividades de defesa de interesse.

O assunto de reconhecimento facial por meio de técnicas de aprendizagem de máquina pode parecer um assunto acessível para aplicar em problemas corporativos apenas para empresas com Facebook, Google, Amazon e outras gigantes que possuem infraestrutura com grande poder computacional. Entretanto, empresas de pequeno e médio porte poderiam sim desenvolver soluções de visão computacional com zero custo de infraestrutura. Primeiro, porque os códigos para elaborar os modelos que treinem imagens do seu ramo de negócio são códigos abertos e de fácil acesso em fórum da internet, segundo porque existem possibilidades de uso de

infraestrutura com arquiteturas robusta gratuitas para qualquer um, como é o caso das plataformas: Colab da Goggle, Kaggle e outros, todos oferecendo processamento GPU gratuito.

O objetivo desse trabalho é colaborar com os profissionais de RelGov para substituírem seus métodos tradicionais de identificar as autoridades por meio do "carômetro" por uma forma tecnológica simples de reconhecimento facial utilizando bibliotecas como OpenCV, Dlib e modelagem em Deep Learning. Além do que, estão disponíveis para todos códigos e infraestrutura que poderá tornar possível para profissionais de outras áreas fora da TI replicar modelos aparentemente complexo de inteligência artificial para seus interesses profissionais.

Será que você consegue utilizar as técnicas apresentadas nesse estudo técnico para sua realidade? É possível apenas trocar o banco de imagens e treinar o modelo para seu próprio uso? As respostas para essas questões você irá encontrar aqui.

2 REVISÃO BIBLIOGRÁFICA

A fundamentação teórica para desenvolvimento desse projeto terá como alicerce as referências bibliográficas referentes à visão computacional e à Inteligência Artificial.

2.1 RelGov e grande rotatividade de cargos

Na prática, sabe-se pouco a cerca das nomeações políticas para administrar um governo sobre e o que influencia na permanência desses nomeados em seus cargos nos três entes federados, apesar da relevância dos conhecimentos que existem sobre assuntos de relações institucionais. Existe uma grande discussão sobre a rotatividade (LOPEZ,2020).

A rotatividade no governo Lula em 2003 já alcançou os limites de 80% no ministério da Educação (MEC) e permanece acima de 70% para os Ministérios da Educação, Cultura e Saúde nesse ano (LOPEZ, 2014).

2.2 Computação Cognitiva

Computação cognitiva está relacionada à sistemas que aprendem com referência a um objetivo bem definido, e esse aprendizado se faz de forma escalonada. (AKABANE, 2018)

A computação cognitiva se refere a computadores executarem ou emularem ações baseadas em aspectos cognitivos como seres humanos fazem (NEVES, 2018)

Eles não são elaborados explicitamente por desenvolvedores, e aprendem devido uma série de interações, armazenando seus resultados por meio de interação com o contexto para o qual foi elaborada

2.3 Machine Learning

Proporcionar aos computadores a capacidade da ação de aprender não requer necessariamente um algoritmo com estruturas de decisões tradicionais, Machine Learning usa um processo de treinamento para encontrar padrões. Já em 1959, Arthur Samuel, pioneiro no estudo de ML, definia dessa forma: Machine Learning é um campo de estudo que dá aos computadores a capacidade de aprender sem ser explicitamente ser programado.

O aprendizado de máquina se refere à capacidade dos sistemas computacionais de melhorar o seu desempenho no processo de disponibilização e exposição dos dados sem que precisem seguir instruções explicitamente programadas. Tem no seu núcleo o processo de descoberta automática de padrões, da estrutura e da natureza dos dados (AKABANE, 2018).

Em Machine Learning, os modelos treinados são aplicados diariamente como ferramenta de tomada de decisões em várias áreas como diagnóstico médico, negociação de ações, na previsão de carga de energia por exemplo (BISPO, 2019). As máquinas reconstroem um padrão e procuram reproduzi-lo, e, essa imitação, pode ser de forma supervisionada ou não supervisionada.

2.3.1 Machine Learnig Supervisionado

Desenvolver um modelo por meio de evidências históricas que produzirão previsões é o objetivo do aprendizado supervisionado. É um modelo que possui conhecimento, para a fase de treinamento, tanto os dados de entrada quanto os dados de saída, pois é assim que sua metodologia funciona. Uma vez treinado, o modelo gera previsões razoáveis quando apresentados dados novos (BISPO, 2019).

A aprendizagem indutiva de funções determinísticas agrega vários modelos como as árvores de decisão, a regressão linear, a regressão logística, as redes neurais, que agrega as funções não lineares complexas e as máquinas de vetores de suporte (SVM), modelo que melhorar a performance de generalização do classificador (RUSSELL, 2004).

2.3.2 Algoritmo de classificação

Suport Vector Machines (SVM) é uma técnica de Machine Learning de classificação. Os resultados da aplicação dessa técnica são empregados na categorização de textos e na análise de imagens. As SVMs utilizam os princípios da teoria de aprendizado estatístico, possuem uma boa generalização e possuem uma capacidade de prever corretamente a classe de novos dados do mesmo domínio em que aprendizado ocorreu (LORENA, 2007).

Outro algoritmo de Classificação, definido como uma heurística de clustering, o K-Means tem como meta minimizar a distância dos seus elementos em análise e o seu conjunto de k centroides, de forma que os cálculos buscam encontra a menor distância dos seus pontos para o centroide mais próximo dele. Esse algoritmo é eficiente quando os cluster são isolados. Usa-se a função de erro quadrático para essa minimização da variação de distância ente os N dados comparando ao dentro de cada cluster. (JAIN, 1999)

2.3.3 Machine Learnig Não-Supervisionado

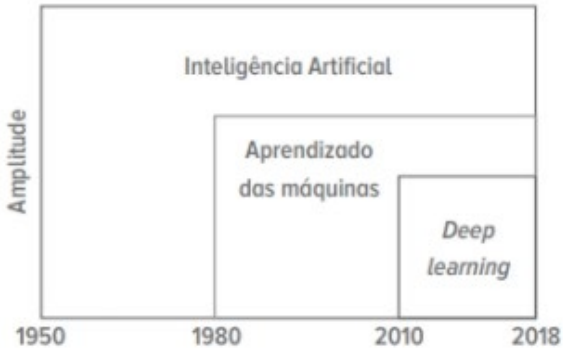
Quando o desafio é distinguir inúmeras categorias em um grupo de objetos, o aprendizado é conhecido como o de formação não supervisionada de agrupamentos. Modelo cujo rótulos de categorias não são apresentados nesse tipo de problema (NORVIG, 2013).

Um dos modelos de aprendizado não supervisionado emprega a técnica de encontrar clustering (agrupamentos). Seu objetivo é encontrar padrões nos dados de entrada por meio de análise exploratória. Um dos modelos de algoritmo de aprendizagem não supervisionada é o Medias-k (K-means) que separa a base de dados em uma quantidade k de classes (BISPO, 2019).

2.4 Deep Learning

Deep Learning é o campo que utiliza em seu modelo vários níveis de representação e abstração para compreender os dados, como imagens, áudio e texto. São modelos formados por múltiplas camadas por meio de funções não linear. Subcampo de Machine Learning, Deep Learning, tem inspiração na arquitetura de redes neurais do cérebro humano, apropriando-se do termo redes neurais artificiais (AKABANE, 2018).

Quadro 1. Linha do tempo das tecnologias



Fonte: AKABANE, 2018

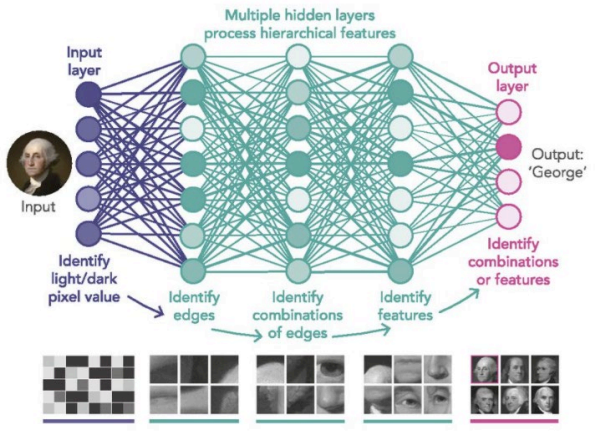
Essa tecnologia busca resolver problemas específicos da indústria e cada desenvolvimento é único. Não dá para treinar um modelo usado para verificar maçãs

poderes, por exemplo, para identificar patologia em uma estrutura de um edifício na construção civil.

O objetivo principal da Deep Learning é a produção de modelos para aprendizado das características (features) de dados de entrada e interpretação deles em grande escala. Elas são aplicadas com frequência para o campo da visão computacional, como o caso de reconhecimento facial, processamento de linguagem natural, reconhecimento de objetos. Deep Learning é um ramo dos estudos de redes neurais com foco em modelagem de dados com alto nível de abstração. Nesse modelo são utilizados métodos de transformação não-lineares e tem como suporte a geração de múltiplas representações para o aprendizado dos dados. (DA CUNHA, 2017)

Métodos de Deep Learning utilizam múltiplos níveis de representação, adquiridos por meio da combinação de módulos simples, entretanto não lineares. Uma imagem, que dá entrada na rede como uma matriz de valores de pixel, por exemplo, e as características aprendidas na primeira camada representa, normalmente, a identificação de bordas em posições específicas na imagem. A função de detectar motivos dentro da imagem identificando arranjos particulares de bordas acontece na segunda camada. A terceira camada pode montar assuntos em combinações maiores que tem relações com partes de objetos correlacionados, e assim, a função de detectar objetos como combinações dessas partes ficariam a cargo das camadas subsequentes. O fato dessas camadas de recursos não serem projetadas por humanos destaca-se como a característica fundamental do Deep Learning (LECUN, 2015).

Figura 1: Redes Neurais Deep Learning



Fonte: LEONEL, 2017

2.5 Imagem digital

Os computadores percebem uma imagem digital como sendo uma matriz matemática (arranjos numéricos), os quais representam os pixels da imagem, eles são seus pontos elementares.

A intensidade de cada pixel, considerando uma imagem em escala de cinza, é composta por 256 níveis, valores entre 0 a 255, cada nível é composto de 8 bits, ou seja, os pixels são números digitais compostos de bits (GONZALEZ, 2000).

O pixel é associado a valor entre 0 a 255 que referencia a intensidade de brilho, ou seja, uma imagem preto e branco é formado por uma matriz unidimensional, já o caso de fotos coloridas sua matriz tem 3 dimensões, cada dimensão represente um canal de RGB. Ou seja, para o contexto do assunto visão computacional a imagem digital é representada como uma matriz bidimensional contendo números inteiros que representa a medidas discretas da energia eletromagnética originada da área observada. O valor de 255 é para branco e o valor 0 para preto, toda a escala de cinza está entre esses 256 valores (8 bits – 2 ˆ 8).

Esse número de bits que representa cada pixel em espaço RGB é a profundidade de pixel, uma imagem RGB possui 3 camadas, vermelha, verde e azul, cada pixel de cores RGB tem uma profundidade de 24 bits. Na tabela abaixo é possível verificar os valores válidos para cada componente RGB em uma cor segura.

O vermelho mais puro pode ser representado por R = 255(FF), G = 00 (00) e B = 00 (00) em decimal ou FF0000 em notação hexadecimal (GONZALEZ, 2000).

Figura 2: Valores válidos para RBG

Sistema numérico	Equivalentes em cores					
Hexadecimal	00	33	66	99	CC	FF
Decimal	0	51	102	153	204	255

Fonte: GONZALEZ, 2000

2.6 Pré-processamento e tratamento de imagens

A fase de captura de uma imagem pode principalmente porque ela já está no formato digital e essa etapa engloba um pré-processamento como o redimensionamento de imagens. Alguns processos de tratar uma imagem tem o objetivo que o resultado final seja mais apropriado em comparação com o original, que é o caso do realce de imagens. (GONZALEZ, 2000).

Muitas imagens precisam passar por um processo de melhoria de qualidade para que o treinamento do modelo tenha um resultado superior, principalmente na fase de treinamento do modelo. Essa fase nada mais é do que aplicar normalização dos dados, como estamos falando de imagens de rosto, os dados de input são fotos de rosto humano e esses dados, assim como qualquer outro, precisam ser normalizados.

Alguns desses tratamento de imagem incluem normalização de brilho e contraste, alinhamento da face e enquadramento/recorte do rosto. Todas essas etapas de higiene dos dados de entrada são necessárias para o conjunto de dados que irão treinar o modelo de reconhecimento facial.

Ao realizar este processo, você terá maior precisão de seus modelos de reconhecimento de rosto uma etapa indispensável como pré-processamento.

2.7 Detecção de rosto (quadrado no rosto)

A detecção de rosto não é o reconhecimento facial, ela é uma etapa que se utiliza de vários métodos matemáticos para esse objetivo.

Um desses métodos é a detecção de bordas, detector de Sobel. Ele é um filtro baseado no gradiente da função de luminosidade da imagem, onde os pixels mais próximos do centro devem apresentar uma maior influência sobre o mesmo (NASCIMENTO, 2005). O operador de Sobel é definido com valores maiores na região central e costuma ser aproximado por operadores 3x3 (NASCIMENTO, 2005, Apud MARQUES; VIERA, 1999). Com esse detector consegue um desempenho de face (olho, nariz, boca) como também todo o contorno da face.

Figura 3: Detector de Sobel

Fonte: NASCIMENTO, 2005

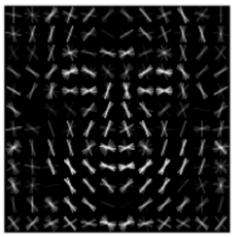
O Histograma de Gradientes Orientados (HOG) é outro método eficiente de extrair característica das cores de pixel para elaborar um classificador de reconhecimento de objeto. Pode-se entender o funcionamento do HOG com o conhecimento dos vetores de gradiente de imagem. A distribuição de histogramas de direções de gradientes (gradientes orientados) é usada como recursos desse método. Os Gradientes (derivadas do eixo x e do eixo y) de uma imagem são essenciais porque em torno das bordas e cantos de uma imagem, que são as regiões de mudanças intensa de intensidade, a magnitude dos gradientes é grande. São nas bordas e nos cantos que contêm muito mais informações sobre a forma do objeto do que regiões planas (MALLICK, 2016).

A magnitude e a direção do gradiente usando provem da fórmula abaixo, mesma para cálculo no Detector de Sobel:

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

Cálculo da hipotenusa dos gradientes X e Y e ângulo de Teta.

Figura 4: Padrão de rosto HOG gerado a partir de muitas imagens de rosto



Fonte: GEITGEY. 2014

Todo processo de reconhecimento facial necessita explorar e detectar pontos de referências do rosto como, nariz, sobrancelhas, olhos, linhas do queixo para delimitar matematicamente a área que ocupa o rosto. Nesse contexto surgiu a ideia de LandMarks.

Outro algoritmo para detecção de rosto chama-se Face Landmark Estimation (estimativa de ponto de referência de face). Dentre os vários métodos têm a abordagem por Vahid Kazemi e Josephine Sullivan em 2014. O fundamento é utilizar 68 pontos específicos (landmarks) encontrado em todas as faces como o queixo, a extremidade dos olhos, as bordas das sobrancelhas etc. (GEITGEY, 2014).

Figura 5: Os 68 landmarks by Brandon Amos

Fonte: GEITGEY, 2014

A partir dessa proposta de landmark detection, Davis King lançou a versão19.7 de dlib - e dentro das notas de lançamento encontra-se o novo detector que usa 5 pontos de detecção LandMarks para referência facial (ROSEBROCK, 2018)

Outra forma de detecção de pontos no rosto foi uma contribuição de Harris que tem estudos para identificar esquinas em uma imagem (Harris Corner Detector) (HARRIS, 1988). A ideia simples em uma forma matemática de encontrar esquinas de imagens, basicamente encontrou-se a diferença de intensidade para um deslocamento de (u,v) em todas as direções. Isso é expresso da seguinte forma (OpenCV, 2020).

Figura 6: Fórmula HOG

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} [\underbrace{I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

Fonte: OpenCV, 2020

2.8 OpenCV

OpenCV ou Open Source Computer Vision Library trata-se de uma biblioteca de código aberto totalmente livre ao uso acadêmico e comercial, que possui várias centenas de algoritmos de visão computacional. A API OpenCV 2.x é essencialmente desenvolvida em C ++.

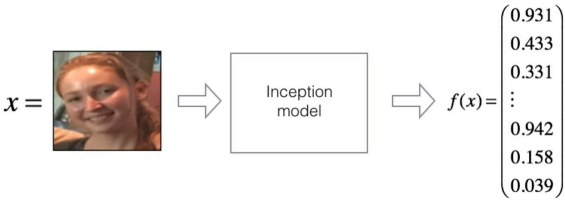
O OpenCV está organizado em módulos, ou seja, os grupos de pacotes incluem muitas bibliotecas como, Processamento de imagem (imgproc), estrutura de recursos 2D (features2d), detecção de objeto (objdetect) dentre outros. (OPENCV, 2020)

2.9 Face Embeddings

Uma das maneiras mais confiáveis de se ter as informações mais importantes para identificar um rosto único é utilizar a Deep Convolutional NetWork (Rede neural Convolucionais Profundas) para chegar a uma identificação de uma imagem com 128 medidas para cada face. Foi a Google em 2015 que aplicou esse algoritmo milhões de vezes para milhões de pessoas de pessoas diferentes - Facenet. O artigo

chama-se: FaceNet: A Unified Embedding for Face Recognition and Clustering. Desta forma, para dez imagens diferentes da mesma pessoa a matriz dessas 128 medidas terão a mesma medida, a esse processo chama-se embedding. (GEITGEY, 2014).

Figura 7: Embedding 128-D



Fonte: SAINI, 2019

3 METODOLOGIA DO TRABALHO

Esse é um estudo técnico que busca alcançar uma melhor qualidade nos resultados e obter menor quantidade de processos possível dentro de um pipeline, abordando as etapas desde a construção do banco de imagens até a implementação e execução do código, veja a seguir as etapas.

Figura 8: Fluxo do trabalho

ETAPA	DESCRIÇÃO
1ª	Levantamento bibliográfico de tutorias
2ª	Escolha do método reconhecimento facial
3ª	Infraestrutura e pré-processamento
4ª	Tratamento das imagens
5ª	Organizando dos dados (dataset)
6ª	Codificando os rostos (Encoding the faces)
7ª	Código reconhecimento facial
8ª	Classificação modelo simples k-NN
9ª	Elaborar caixas delimitadoras
10ª	Construção de um app (MVP)

Fonte: produzido pelo autor

A primeira etapa, foi composta por um levantamento bibliográfico na WEB sobre quais tutoriais que aplicam o reconhecimento facial de forma pragmática e com argumento de autoridade nos modelos empregados. O objetivo é encontrar uma

fonte de conteúdo que construa o passo-a-passo com base em referências sólidas que tratam sobre visão computacional. Escolher um tutorial requer muita pesquisa para obter um benchmak adequado para alcançar um resultado que atenda o objetivo desse estudo técnico.

A segunda etapa, define-se como a fase da escolha do método de reconhecimento facial propriamente dito, de forma que ele possa ser aplicado num protótipo e torne possível ser um instrumento de trabalho para os profissionais de relações governamentais no seu dia-a-dia.

A terceira etapa, foi responsável pela adaptação e desenvolvimento da infraestrutura e pré-processamento para executar o código. Criar um conjunto de dados de imagens, isso inclui escolher a quantidade e a qualidade de imagens por cada pessoa que será treinada no modelo, assim como de que forma capturar as fotos na Internet.

A quarta etapa é a fase de tratamento das imagens, alinhamento facial. Imagens enquadradas e centralizadas é a garantia de uma entrada apropriada para uma rede neural.

A quinta etapa tem o foco na organizando dos dados (dataset) em diretórios que contêm o nome das pessoas que serão treinadas no modelo. Cada pessoa terá seu próprio diretório (imagePaths). O total serão 27 diretórios sendo 4 arquivos no diretório raiz e 4 diretórios de nível superior.

A sexta etapa é a codificando os rostos (Encoding the faces), fase de transformar os rostos, que são imagens, em vetores numéricos (embeddings de 128-d), isso será aplicado para o conjunto de fotos de treinamento, de forma para esse conjunto de dados ficar pronto e disponível no momento de execução do código, pois não será treinada uma rede do zero e sim será utilizada uma rede pré-treinada que já foi treinada com um conjunto de dados de 3 milhões de imagens. Ou seja, codificar os rostos em nosso conjunto de dados (encode_faces.py). Nessa fase são necessários alguns argumentos: o caminho do diretório aonde encontra-se o arquivo contendo as faces codificadas, o caminho para gerar o arquivo que conterá a codificação das faces (facial encodings) e a especificação do método de detecção do rosto.

A sétima etapa já se constitui do código para o trabalho de reconhecimento em si, pois os rostos do conjunto de dados já estão vetorizados (embeddings de 128-d). Essa fase será inserida uma nova imagem para que o modelo possa reconhecer dentre as imagens treinadas e disponíveis para o código. Nessa etapa também será necessário inserir os argumentos para com o caminho/diretório de onde encontra-se o arquivo com as faces codificadas (embeddings de 128-d), a imagem que será verificada o reconhecimento facial e especificar o método de detecção do rosto.

A oitava etapa é classificação final da face com modelo simples k-NN + votos para fazer processo de classificação em si e obter a informação do rosto que tem mais proximidade (faces treinadas) com o rosto que está sendo analisado. Aqui são carregadas as imagens codificadas de rostos (encoding face) e seus respectivos nomes, pois em uma foto podem existir mais de um rosto identificado com o conjunto de dados treinados.

Nona etapa é a forma gráfica de elaborar caixas delimitadoras e inserir os nomes rotulados.

A décima etapa, envolve a construção de um app dentro do conceito de MVP para demonstração do código em uma situação real.

De forma geral as dez etapas abrangem os seguintes assuntos: Módulo Pesquisar Soluções, Módulo Conjunto De Dados, Módulo Pré-Treino, Módulo Reconhecimento Facial e Módulo Colocar Em Produção

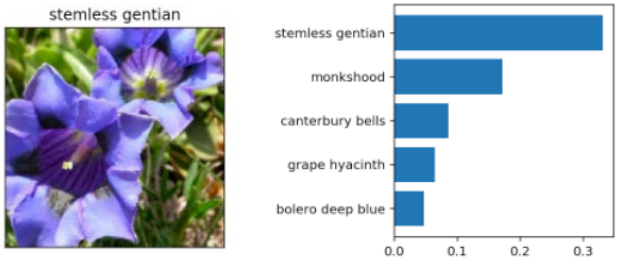
4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

4.1 Levantamento bibliográfico de tutoriais.

O conhecimento sobre reconhecimento de imagens não era um assunto desconhecido, o pesquisador cursou dois módulos no portal da Udacity: PyTorch Scholarship Challenge from Facebook e Nanodegree Deep Learning with Pytorch. Os cursos foram uma oportunidade de bolsa de estudo. O primeiro curso (fase 1) teve uma abordagem introdutória, as empresas ainda oferecem vagas gratuitas por meio de bolsa todos os anos no link: https://www.udacity.com/scholarships/facebook-pytorch-scholarship. O segundo

curso (fase2), Nanodegree em Deep Learning foi a fase subsequente para os alunos que tiveram boa classificação na primeira fase. Na última tarefa dessa fase houve uma atividade para fazer classificação de imagens de flores por meio de redes neurais. A tarefa era utilizar um modelo já treinado para fazer previsões de flores apresentando um gráfico contendo o ranking das cinco espécies com mais alta probabilidade. Todas as fontes dos códigos da Udacity estão no Github, local que contém uma ampla oportunidade para aprender sobre visão computacional. Essa tarefa específica pode ser encontrada no link: https://github.com/udacity

Figura 9: Fluxo do trabalho



Fonte: Udacity

https://github.com/udacity/pytorch_challenge/blob/master/Image%20Classifier%20Project.ipynb

Como o objetivo desse projeto estava focado em criar um protótipo para reconhecimento de faces humanas, foram pesquisados diversos conteúdos disponíveis na Internet, entretanto um site se destacou na pesquisa: o site Pyimagesearch, um ambiente que se compromete usar visão computacional e Deep Learning de maneira bem aprofundada, com uma didática adequada e com referências de autoridades nos assuntos abordados. O idealizador desse site é Adrian Rosebrock, PhD em Ciência da Computação pela University of Maryland, Baltimore County (UMBC) em 2014. Ele atualmente publica toda semana novos tutoriais. O site tem mais de cinco anos que ajuda desenvolvedores no assunto de Visão Computacional, Deep Learning e OpenCV. Todos os códigos desse experimento foram retirados do Pyimagesearch alguns ajustes e adaptações foram adicionados pelo pesquisador. O conteúdo de estudo e aprofundamento dos assuntos dos tutoriais estão na referência bibliográfica. No total foram mais de 5 tutoriais estudados para

conseguir o melhor resultado e a duração do experimento foi mais de 7 meses, de julho de 2019 até fevereiro de 2020.

4.2 Escolha do método de reconhecimento facial.

O critério para escolher dentre os mais de 350 tutoriais de acesso gratuitos para soluções de problemas do mundo real foi o fato do protótipo precisar de uma forma de usar o reconhecimento facial tanto para fotos estáticas quanto para vídeos. O site tem os assuntos categorizados por catorze tópicos:

Tabela1: Tópicos de assuntos

Image Processing	Optical Character Recognition (OCR)
Machine Learning	Keras and TensorFlow
Deep Learning	Embedded/IoT and Computer Vision
Raspberry Pi	Face Applications
OpenCV Tutorials	Object Tracking
Object Detection	Medical Computer Vision
Interviews	
dlib	

Fonte: Pyimagesearch.com

Como a pesquisa é acadêmica, encontrar bibliotecas de código aberto é uma alternativa muito promissora, desta forma, a pesquisa dentro do site foi com os termos OpenCV e reconhecimento facial.

Dentre as opções que surgiram na pesquisa foi encontrado um tutorial que utilizava como recurso de entrada tanto fotos quanto vídeos. Essas variedades de opções nas entradas dos dados são fundamentais para criação de um protótipo mais completo de forma que os profissionais de Relações governamentais possam uma solução tecnológica mais robusta em seu ambiente de trabalho.

Os nomes dos tutoriais encontrados estão abaixo, sendo o mais importante o primeiro, os demais auxiliaram no estudo e em códigos complementares:

Tabela2: Tópicos de assuntos

Face recognition with OpenCV, Python, and deep learning. 2018.
Face Alignment with OpenCV and Python. 2017
How to (quickly) build a deep learning image dataset. 2018.
OpenCV Face Recognition 2018.
Facial landmarks with dlib, OpenCV, and Python. 2017.
(Faster) Facial landmark detector with dlib. 2018.

Fonte: Pyimagesearch.com

4.3 Infraestrutura e pré-processamento.

O tutorial de reconhecimento facial usará tecnologia para identificar fotos e vídeo em tempo real a partir de algumas bibliotecas de OpenCV e Python.

A infraestrutura para rodas os códigos precisam de instalar várias bibliotecas, o ideal é criar um ambiente virtual instalando via comando pip os conteúdos: Python, OpenCV, dlib, face_recognition, imutils, requests.

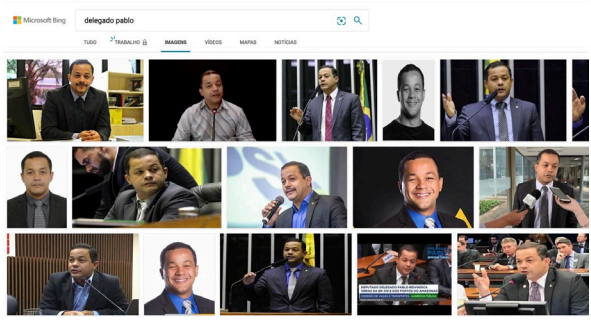
Para qualquer treinamento de faces o primeiro desafio é construir um conjunto de dados. A resolução usual para imagens de entrada no processo de reconhecimento facial deve estar no limite entre 64×64 até 224×224 pixel. O tamanho da imagem que ela ocupa do disco não importa tanto para o resultado final o que importa é a resolução (ROSEBROCK, 2018).

Como mencionado, é necessário montar um conjunto de dados de imagens para poder treinar o modelo. Para coletar uma quantidade de fotos de forma automática será necessário um método para essa automação. Foi encontrado dois métodos, um captura imagens por meio do Google Images, o código é disponibilizado pela Kaggle, de propriedade da Fast.Ai, que está disponível com nome Creating your own dataset from Google Images (HOWARD, 2018), o outro é encontrado nos tutoriais da Pyimagesearch. Ele utiliza a API do Bing, chama-se How to (quickly) build a deep learning image dataset (ROSEBROCK, 2018). O

objetivo não é detalhar esses códigos e sim usá-lo como ferramenta para capturar uma quantidade suficiente de fotos para treinar os rostos desse protótipo.

Foi escolhido arbitrariamente usar imagens de deputados pois eles têm suas imagens públicas. Como 513 deputados é um número muito grande para essa fase do projeto, escolheu-se trabalhar 20% desse número, a quantidade de 106 deputados é razoável para alcançar um produto significativo. Sobre a quantidade de fotos o próprio Rosebrock sugere a número entre 20 a 200 fotos, considerando que o treinamento melhora a medida em que se consegue mais imagens. Um ponto importante é que as condições de luz, e ângulo de visão do ambiente onde as imagens foram capturadas na internet deve ser semelhante ao ambiente de onde o protótipo será implantado (ROSEBROCK, 2018).

Figura 10: Pesquisa por imagens



Fonte: Microsoft Bing

Será utilizado o método de captura de imagem do Bing por fazer parte dos tutoriais do PyImageSearch. Antes de rodar o código de captura é necessário habilitar o serviço gratuito da Microsoft para obter a chave da API e inserir no código na variável API_KEY:

```
28    API_KEY = "95c7796a38844d59a5e151c25625c65e"
29    MAX_RESULTS = 20
30    GROUP_SIZE = 50
```

Com essa etapa prontas, roda-se o código de captura imagem pelo API Bing:

python search_bing_api-TCC.py --query " delegado pablo" --output dataset/ delegado_pablo

Os argumentos são: --query que é o assunto de pesquisa e –output o diretório para onde irão as imagens baixadas. Após o resultado do comando de captura é fundamental verificar e eliminar manualmente todas as fotos que não fazem parte do assunto procurado, assim como as fotos que estejam na posição de perfil e as imagens escuras demais.

4.4 Tratamento das imagens.

Ainda fazendo parte do processo de pré-processamento, o tratamento das imagens antes do treinamento é a parte mais demorada e essencial para obter melhores resultados. Após algumas pesquisas no portal Pyimagesearch foi encontrado um tutorial do Rosebrock que apresenta uma alternativa para alinhar um rosto usando OpenCV, Python e marcas faciais (landmarks). O algoritmo de alinhamento de face é baseado no Capítulo 8 de Mastering OpenCV with Practical Computer Vision Projects (ROSEBROCK, 2018, Apud, BAGGIO, 2012).

É preciso especificar os argumentos de quais imagens serão o conjunto de entrada e o tipo de método de predição, que nesse caso será shape_predictor_68_face_landmarks.dat. O objetivo é gerar imagem de saída que contenham: imagem centrada, olhos na linha horizontal, dimensões das imagens similares.

Para executar esse código algumas mudanças foram implementas pelo pesquisador no código original. Foi adicionado um argumento para que o programa lesse todos os arquivos de uma pasta (--dataset) e uma pasta de saída (--writing), segue linha de comando:

```
16  # construct the argument parser and parse the arguments
17  ap = argparse.ArgumentParser()
18  ap.add_argument("-i", "--dataset", required=True,
19      help="path to input directory of faces + images")
20  ap.add_argument("-p", "--shape-predictor", required=True,
21      help="path to facial landmark predictor")
22  ap.add_argument("-w", "--writing", required=True, # p/Walner
23      help="path to output image") # adicionado p/Walner
24  args = vars(ap.parse_args())
```

Python align_faces.py --shape-predictor shape_predictor_68_face_landmarks.dat --dataset dataset --writing output/file

Uma característica dos códigos do PyImageSearch é usar recurso de argparse para inserir argumentos para o código na própria linha de comando

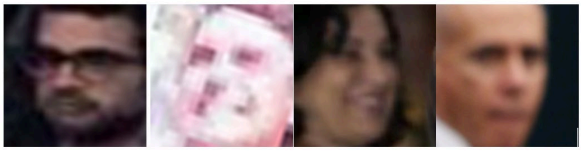
O resultado das imagens de saída são imagens que recortam a figura original aproveitando apenas com o rosto de forma que ele fique alinhado horizontalmente.

Figura 11: Alinhamento de imagem

Fonte: montada pelo autor/ Microsoft Bing

Esse algoritmo identifica, recorta e alinha todos os rostos encontrados em cada foto, de forma que haverá muitos rostos que precisam ser apagados manualmente poia eles não são alvo da pessoa que está sendo conteúdo de treinamento.

Figura 12: Imagens descartadas



Fonte: montada pelo autor/ Microsoft Bing

4.5 Organização dos dados (dataset).

A forma que o reconhecimento facial insere o label (tags) para identificar a imagem vem do nome dos diretórios, ou seja, o nome digitado na pasta será o nome apresentado na legenda da identificação do rosto.

Depois de seguir o pipeline de pré-processamento e tratamento, o resultado final é um grupo de 106 pastas com o nome de cada deputado.

As 106 pastas precisam estar dentro do diretório dataset, além dessa, outras pastas precisam ser criadas:

Os arquivos com os códigos principais deverão estar no diretório raiz:

- search_bing_api.py: para capturar conjunto de imagens por meio da API do Bing
- align_faces.py: para cortar e alinhar as faces.
- encode_faces.py: para gerar as codificações das imagens (vetores de 128-d)
- encodings.pickle: arquivo que armazenará as saídas do código encode_faces.py.
- recognize_faces_image.py: para reconhecer as faces em uma única imagem

4.6 Rede pré-treinada & Encoding the faces

Importante ressaltar que as imagens coletadas nas fases anteriores não farão o treinamento a partir do zero. Ela irão incorporar da dlib (open source maching learning library) na mesma arquitetura do modelo ResNet-34 que alcança uma precisão muito competitiva (HE, 2016) mas com menos camadas e o número de filtros reduzidos em 50% (ROSEBROCK, 2018), ou seja, a rede foi treinada para criar embeddings de 128-d em um conjunto de dados de aproximadamente 3 milhões de imagens.

Figura 13: Valor da precisão

Precisão de 99,38%

Fonte: montada pelo autor

A rede foi treinada pelo próprio criador do dlib, Davis King em um conjunto de dados de imagens fornecida pela Labeled Faces in the Wild - LFW, a rede alcançou uma precisão de 99,38% (KING, 2017). O resultado está listado no site http://vis-www.cs. umass.edu/ lfw/results.html no método de número 90 dessa listagem, com data de fevereiro de 2017. O processo que acontece nesse algoritmo usa essa rede pré-treinada de forma que na sequência utiliza ela para construir encapsulamento de 128-d, ou seja, um vetor de "features" de saída, uma lista de 128 números com valor real de forma a quantificar cada rosto (ROSEBROCK, 2018). Com esse experimento terá 106 deputados o total de fotos será de 3618 rostos em nosso conjunto de dados, uma média de 34 rostos, contendo 11 fotos de mínimo e 70 de máximo nas pastas.

Figura 14: Embeddings de 128-d

Fonte: derivada a partir de Geitgey,2016/Microsoft Bing

Para rodar esse código digita-se a linha de comando abaixo:

python encode_faces-TCC.py --dataset dataset --encodings encodings.pickle

Conforme explicado o método Arparse indica alguns argumento: dataset – local onde estarão as 106 pastas com as faces dos deputados; encodings – apresenta o nome do arquivo, encodings.pickle, onde está gravada as imagens vetorizadas (128-d); e detection-method – alternativa de escolher um entre dois métodos de detecção de rosto, HOG ou CNN o default é CNN.

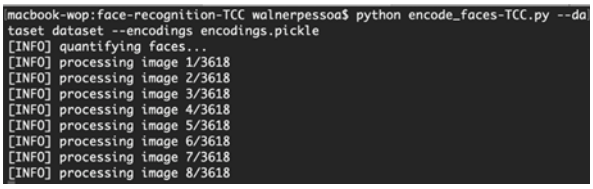
```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
    help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
    help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
    help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())
```

A variável imagePaths guardará uma lista contendo os diretórios do conjunto de dados de entrada. Outras duas variáveis criadas knownEncodings e knownNames conterão as codificações faciais (128-d) e os nomes correspondentes para cada face.

O código fará um loop com 3618 repetições, onde em cada uma das 106 pastas será considerado que todas as faces pertencem a uma mesma pessoa.

Um dicionário contendo duas chaves (encodings & names) será a forma de acesso desses 106 deputados, ele será utilizado para identificação no reconhecimento facial.

Figura 15: Comando do Terminal



Fonte: elaborada pelo autor

No Macbook Pro 2,7 Ghz, processador Intel Core I5, Dual Cores, RAM 16 GB, o processo levou 60 minutos para o programa fazer o código das faces (128-d) das 3618 imagens.

4.7 Código de reconhecimento facial

Reconhecimento facial é uma sequência de soluções formando um pipeline com métodos que estão nas bibliotecas importadas para o algoritmo (GEITGEY, 2016):

Figura 16: Pipeline de reconhecimento facial:

Encontrar a face
Analisar as características
Comparar com as faces conhecidas
Buscar precisão

Fonte: elaborada pelo autor

Segue detalhamento que Geitgey considera:

1. identificar todos os rostos dentro da imagem analisada – a detecção de rostos neste algoritmo acessa o método Histograma de gradientes orientados - HOG;

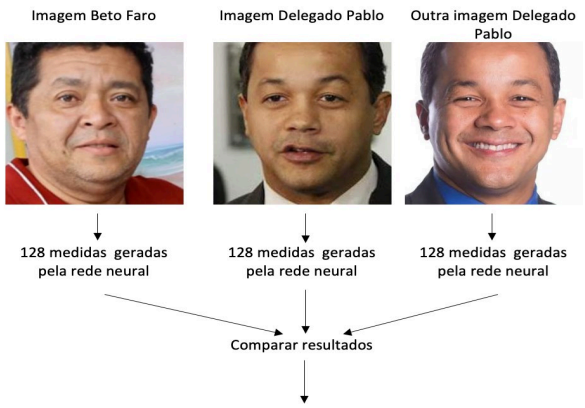
2. cerificar que cada face está no padrão de um rosto humano. O algoritmo face landmark busca 68 pontos de referências para girar, dimensionar e distorcer a imagem de modo que os olhos e a boca estejam centralizados da melhor forma possível;

3. medir as características da face que está em análise e encontrar a codificação de 128-d (encoding faces) para poder comparar os vetores de características únicas das medidas do rosto de forma para encontrar as semelhanças no rosto analisado com o conjunto de dados que já está em 128-d (encodings.pickle). Essa análise ocorre pela técnica A Single “triplet” training step (Um único passo de treinamento triplo).

4. comparar as características exclusivas desse rosto usando algoritmo básico de classificação com o conjunto de dados dos rostos que foi treinado para determinar o nome da pessoa.

O arquivo recognis_faces_image.py contém a parte que irá fazer o reconhecimento

Figura 17: A Single “triplet” training step



Ajustar a rede neural suavemente para que as medidas das duas fotos do deputado Delegado Pablo fiquem mais próximas e as medidas do Deputado Belo Faro fiquem mais longe.

Fonte: derivada a partir de Geitgey,2016/ Microsoft Bing

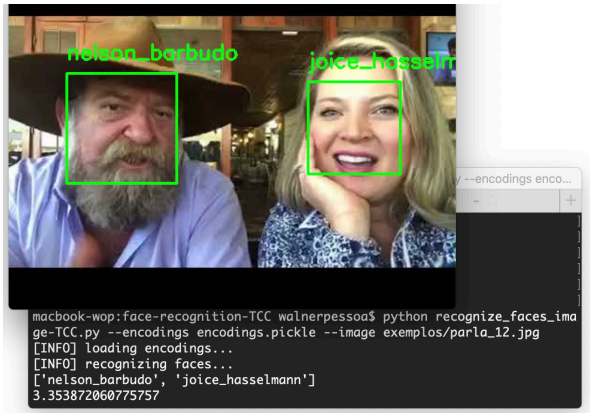
A biblioteca dlib, mantida por Davis King, é utilizada para construir os empacotamentos (embeddings) de rosto. A biblioteca face_recognition, criada por Adam Geitgey, está relacionada com as funcionalidades de reconhecimento facial do dlib.

```
5   import face_recognition
6   import argparse
7   import pickle
8   import cv2
9   import requests ######## walner
10  import time
```

O comando terminal para executar o código precisa incluir os argumentos com o caminho/diretório de onde encontra-se o arquivo com as faces codificadas (embeddings de 128-d), a imagem que será verificada o reconhecimento facial e especificar o método de detecção do rosto:

python recognize_faces_image-TCC.py --encodings encodings.pickle --image exemplos/parla_12.jpg

Figura 18: Imagem do teste de execução



Fonte: elaborada pelo autor/ Microsoft Bing

Duas informações foram implementadas para aperfeiçoar o código, elas são as últimas duas linhas da imagem acima: um array para guardar todos os nomes identificados e uma variável para contabilizar o tempo gasto para o processo de identificação.

4.8 Classificação modelo simples k-NN

Para buscar a classificação final da face esse código utiliza o modelo k-NN + quantidade de votos.

O método compare_face utiliza a distância euclidiana para gerar uma lista de valores boleanos para cada imagem dentro do conjunto de dados, ou seja, um array de 3618 valores. Essa distância tem uma medida de tolerância para ajustar o grau de exigência ou flexibilidade do sistema (ROSEBROCK, 2018).

```
41  for encoding in encodings:
42
43      matches = face_recognition.compare_faces(data["encodings"],
44          encoding, 0.4)
45      name = "Unknown"
```

Uma lista de correspondências irá fazer o trabalho de calcular o número de "votos" para cada nome, agregar os votos e identificar o nome do deputado com o maior número de votos correspondentes. Isso ocorre em conjunto com a variável dicionário de contagens que identificará a mais alta pontuação de votos para a chave nome. A quantidade de imagens por deputados dará o limite de votos, por isso é importante a quantidade de fotos ser similar entre o conjunto de dados (ROSEBROCK, 2018).

4.9 Elaborar caixas delimitadoras

As caixas delimitadoras e os nomes rotulados de cada pessoa precisa ser desenhado na imagem analisada de saída visualizar os rostos identificados e respectivos nomes com um box verde RGB (0,255,0).

```
72  for ((top, right, bottom, left), name) in zip(boxes, names):
73      cv2.rectangle(image, (left, top), (right, bottom), (0, 255, 0), 2)
74      y = top - 15 if top - 15 > 15 else top + 15
75      cv2.putText(image, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
76          0.75, (0, 255, 0), 2)
77  cv2.imshow("Image", image)
78  cv2.imwrite("saida_imagem.jpg",image)
79
```

4.10 Construção de um app (MVP)

O mínimo produto viável (MVP) é um produto, ou APP, que agrega todas as características fundamentais e necessárias para ser considerado como um produto

mínimo aceitável, ou seja, funcionalidades suficientes que provem o valor desse produto mínimo (VIDAL, 2018).

Para fazer esse essa APP, parte do projeto contou com a parceria de um ex-aluno, Pedro Reis, amigo de sala do pesquisador. Ambos se formaram no curso de Ciências da Computação do Ceub em 2018. Todo a parte do design, escolha de cores e a criação da logomarca foi feito pelo pesquisador e o deploy, incluindo a montagem do código em React e a parte de hospedagem e geração de um arquivo executável para Android foram responsabilidade do convidado.

O APP foi batizado de CARÔMETRO360 por tentar resgatar a forma coloquial de se referir a função dos profissionais de RelGov para identificar por fotogramas uma autoridade. E por ser uma proposta com ampla funcionalidade acrescentou-se o número 360.

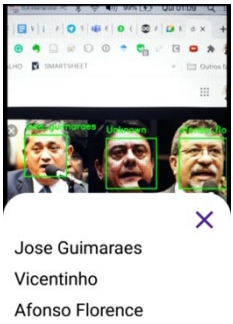
A solução foi testada de forma local no computador do pesquisador e a próxima etapa seria colocar o sistema em produção.

A App foi desenvolvido em React e a arquitetura cliente/servidor foi escolhida como solução ideal. O React funciona em estrutura de árvore similar ao Html. Os códigos desenvolvidos em Python (recognize_faces_video-TCC.py) e o arquivo contendo o treinamento com as imagens do deputados (encodings.pickle) com os 3618 vetores 128-d, estão no servidor.

O App faz as requisições usando o método Post do protocolo da camada de aplicação e estabelece contato com o Servido Nodes, o host cliente envia no body do Post o arquivo de imagem que a câmera do smartphone capturou. Do lado do host servidor, o código Python gera duas saídas: um arquivo (jpg) com o link da imagem e uma variável array com os nomes identificados na foto do usuário. Na fase seguinte o APP usa o método Get para pegar a foto (link) e gera visualização no smartphone. Essa descrição acontece para cada requisição do usuário.

O protótipo roda apenas em smartphone Android e abaixo segue algumas telas do APP e parte do código React que solicita a requisição para o servidor. O Servidor está hospedado na DigitalOcean com IP 68.183.124.14:5000.

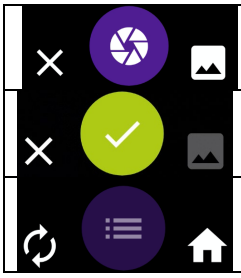
Figura 19: Imagem App Carômetro360



Fonte: elaborada pelo autor

O funcionamento do APP é intuitivo. Após clicar no botão Começar é solicitado permissão para uso da câmera para o Sistema Operacional do smartphone, depois basta o usuário clicar uma vez no botão com desenho do obturador roxo e confirmar no botão verde. O botão com três traços oferece a alternativa de visualizar a lista de nomes identificados.

Figura 20: Funcionalidades Carômetro360

	Acionar o disparo da foto ou galeria de fotos.
	Confirmar escolha da foto
	Visualizar a lista de nomes das autoridades.

Fonte: elaborada pelo autor

5 CONSIDERAÇÕES FINAIS

O campo profissional de Relações Governamentais tem desafio de reconhecer as autoridades num ambiente com tanta rotatividade em cargos públicos e a tecnologia de visão computacional apresenta-se como uma ferramenta confiável com uma precisão de 99,38% (KING, 2017).

Por ser um campo de aplicação específica, a área de visão computacional pode produzir importantes impactos práticos na sociedade. Isso significa que para treinar um modelo para reconhecimento de padrões nas diversas áreas profissionais é preciso delimitar um escopo restrito, como foi o caso desse estudo técnico. Dito de forma oposta, não é possível fazer um reconhecimento de padrões generalizado para todas as coisas.

Esse estudo técnico convida os engenheiros, que tem em sua gênese solucionar problemas, para uma reflexão sobre produzir soluções relevantes para os diversos campos da sociedade. Apesar das técnicas de Deep Learning serem complexas, elas estão acessíveis aos interessados que possuem conhecimento básico em Ciência de Dados. Além da oferta de estudos, tutoriais e curso, a infraestrutura de hardware com poder de velocidade de processamento estão acessíveis nos serviços de nuvens (Google, Microsoft, Amazon). Isso proporciona uma concreta possibilidade de solução dos problemas de reconhecimento de padrões no mundo real pelos engenheiros.

REFERÊNCIAS

AKABANE, G. K. Gestão estratégica das tecnologias cognitivas: Conceitos, metodologias e aplicações. São Paulo: Érica, 2018.

BISPO, Fabrício Machado da Silva; Maikon Lucian Lenz; Pedro Henrique Chagas Freitas; Sidney C. Inteligência artificial. Grupo A, 2019.

BRAGA, A. de P.; CARVALHO, ACPLF; LUDERMIR, Teresa B. Redes neurais artificiais. Teoria e Aplicaçoes, 2000.

DA CUNHA, KELVIN BATISTA. Reconhecimento e detecção de logotipos a partir de redes neurais convolucionais profundas, 2017.

DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). IEEE, 2005. p. 886-893.

GEITGEY, Adam, Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. 2016. Disponível em < https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>, acesso em 10 outubro.2020.

GONZALEZ, Rafael C.; WOODS, Richard E. Processamento de imagens digitais. Editora Blucher, 2000.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In Fourth Alvey Vision Conference, Manchester, UK, pp. 1988.

HE, Kaiming et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

HOWARD, Jeremy, Creating your own dataset from Google Images, 2018. Disponível em < https://www.kaggle.com/init27/fastai-v3-lesson-2>, acesso em 10 outubro.2020.

JAIN, A.K., MURTY, M.N. & FLYNN, P.J. Data Clustering: A Review, ACM Computing Surveys, vol. 31, no. 3, pp 264-323,1999.

KING, Davis E. Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research, v. 10, p. 1755-1758, 2009.

KING, Davis, High Quality Face Recognition with Deep Metric Learning. 2017. Disponível em <http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html> acesso em novembro.2020.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. Communications of the ACM, v. 60, n. 6, p. 84-90, 2017.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. nature, v. 521, n. 7553, p. 436-444, 2015.

Leonel, Jorge S. Deep Learning. Disponível em < https://deeplearningbrasil.Word press .com/>, Acesso em: 10 outubro .2017.

LOPEZ, Felix Garcia; BUGARIN, Maurício; BUGARIN, Karina. Rotatividade nos cargos de confiança da administração federal brasileira (1999-2013). 2014.

LOPEZ, Felix; SILVA, Thiago. O Carrossel burocrático nos cargos de confiança: análise de sobrevivência dos cargos de direção e assessoramento superior do executivo federal brasileiro (1999-2017). 2020.

LORENA, Ana Carolina; DE CARVALHO, André CPLF. Uma introdução às support vector machines. Revista de Informática Teórica e Aplicada, 2007, 14.2: 43-67.

MALLICK, Satya, Histogram of Oriented Gradients, 2016. Disponível em < https:// www.learnopencv.com/histogram-oforiented -gradients/>, acesso em 10 outubro. 2020.

MINSKY, Marvin. A framework for representing knowledge. MIT-AI Laboratory Memo 306. Massachusetts Institute of Technology, 1974.

NASCIMENTO, Andréia Vieira do. Detecção de faces humanas em imagens digitais: um algoritmo baseado em lógica nebulosa. 2005. Tese de Doutorado. Universidade de São Paulo.

NEVES, B. C. Computação cognitiva: novas perspectiva para a ciência da informação. InfoHome. São Paulo, p. 1- 3. dez, 2018. Disponível em: https://www.ofaj.com.br/colunas_conteudo.php?cod=1100. Acesso em: 10 jul. 2018.

NORVIG, Peter. Inteligência Artificial. Grupo GEN, 2013.

OpenCV - Documentação oficial, 2020. Disponível em < https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html >, acesso em 10 outubro. 2020.

ROSEBROCK, Adrian, (Faster) Facial landmark detector with dlib. 2018. DIsponível em <https://www.pyimagesearch.com/2018/04/02/faster-facial-landmark-detector-with-dlib/>, acesso em 10 outubro.2020.

ROSEBROCK, Adrian, Face Alignment with OpenCV and Python. 2017. Disponível em < https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/> acesso em 10 outubro.2020.

ROSEBROCK, Adrian, Face recognition with OpenCV, Python, and deep learning. 2018. Disponível em < https://www.pyimagesearch.com/2018/04/02/faster-facial-landmark-detector-with-dlib/>, acesso em 10 outubro.2020.

ROSEBROCK, Adrian, Facial landmarks with dlib, OpenCV, and Python. 2017. Disponível em < https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>, acesso em 10 outubro.2020.

ROSEBROCK, Adrian, Histogram of Oriented Gradients and Object Detection. 2014. Disponível em < https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/>, acesso em 10 outubro.2020.

ROSEBROCK, Adrian. How to (quickly) build a deep learning image dataset. 2018. Disponível em < https://www.pyimage search.com/2018/04/09/how-to-quickly-build-a-deep-learning-image-dataset/>

ROSEBROCK, Adrian, OpenCV Face Recognition 2018. Disponível em < https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/ acesso em 10 outubro.2020.

RUSSELL, Stuart J.; NORVIG, Peter. Inteligência artificial. Elsevier, 2004.

SAINI Mohit, Train FaceNet with triplet loss for real time face recognition…,2019. Disponível em < https://mc.ai/train-facenet-with-triplet-loss-for-real-time-face-recognition/>, acesso em 10 outubro. 2020.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 815-823.

TRASK Andrew W. Grokking Deep Learning, pgs.11 – 17. MEAP Edition, 2017.

VIDAL, Ricardo da Rocha Duarte Lira. Proposta de uma metodologia de avaliação de investimentos em ativos de ti no setor público. 2018.

WINSTON, P. H., The Psychology of Computer Vision, McGraw Hill, 1975.

CLASSIFICAÇÃO E MAPEAMENTO DO USO E COBERTURA DO SOLO DO DISTRITO FEDERAL UTILIZANDO TRÊS DIFERENTES ALGORITMOS DE APRENDIZAGEM DE MÁQUINA

João Paulo Fernandes Márcico Ribeiro

Ivandro Ribeiro

RESUMO

O Cerrado brasileiro abrange boa parte do território nacional, diferenciando-se ape-nas em sua composição florestal. Devido às aceleradas mudanças do uso do solo no bioma, torna-se imprescindível o monitoramento da área a fim de garantir sua conservação. Para isso, a combinação do sensoriamento remoto com algoritmos de aprendizagem de máquinas facilita as diversas análises e estudos. O objetivo deste trabalho é aplicar três algoritmos de classificação de uma imagem de satélite para avaliar a acurácia dos algoritmos no Distrito Federal, inserido no bioma Cerrado. Foram utilizados o algoritmo de Máximo Verossimilhança (MAXVER), Máquina de Vetor de Suporte (SVM) e Floresta Aleatória. Todos os algoritmos apresentaram con-fusões nas classificações de áreas urbanas, agropecuária e savanas. O resultado do algoritmo de Floresta Aleatória apresentou melhor acurácia global com 82,5% e maior valor do índice Kappa, sendo de 0,766. Outros estudos encontraram dificul-dades semelhantes com a classificação de áreas agrícolas, savana e áreas urba-nas, por conta das complexidades e padrões espectrais no ambiente que se mos-tram semelhantes no período da seca. Mapeamentos mais complexos desse bioma tiveram melhores resultados usando o algoritmo de Floresta Aleatória.

Palavras-chave: Sensoriamento Remoto. Aprendizagem de máquina. Máximo Verossimilhança. Máquina de vetor de suporte. Floresta Aleatória.

1 INTRODUÇÃO

O bioma Cerrado, conhecido também como a savana tropical brasileira, é o segundo maior bioma na América do Sul com mais de dois milhões de quilômetros

quadrados. Por abranger boa parte do território brasileiro, possui diferentes fitofisionomias que se relacionam com o tipo de clima e solo. O bioma tem sido ameaçado pela constante conversão das vegetações nativas para outros tipos de uso do solo como áreas de agriculturas e pecuária (SANO et al., 2019; STRASSBURG et al., 2017; MITTERMEIER et al., 2017).

Atualmente existem diversos algoritmos de classificação de imagem que podem ser aplicados para diversas situações (PEDRINI; SCHWARTZ, 2008). O avanço das linguagens de programação combinado com os diversos algoritmos de aprendizagem de máquinas torna cada vez mais fácil a aplicação no sensoriamento remoto, permitindo a utilização em grandes volumes de dados, já que são feitos imageamentos com grandes frequências temporais e estudos em diferentes escalas como grandes cidades, biomas ou até mesmo continentes (SAWYER, 2002).

O objetivo do presente trabalho foi aplicar três tipos de algoritmos de aprendizado de máquina para classificação de uma imagem de satélite, observando qualitativamente os resultados quanto à acurácia.

Para alcançar esses objetivos, procedeu-se da seguinte maneira: inicialmente realizou-se o download de uma imagem de satélite com as devidas correções atmosféricas (visando a não interferência no resultado da classificação), dando sequência com a coleta de amostras por meio de softwares de geoprocessamento e posterior aplicação dos métodos de classificação utilizando o algoritmo de Máximo Verossimilhança (MAXVER), Máquina de Vetor de Suporte (SVM) e Floresta Aleatória, finalizando com a avaliação da exatidão global, acurácia do usuário e do produtor além do índice Kappa.

Assim sendo, este estudo foi estruturado nas seguintes seções: Na seção dois apresentam-se uma revisão bibliográfica sobre sensoriamento remoto, história da série de satélites Landsat, características do Landsat 8, correções atmosférica e reflectância ao topo da atmosfera, características da áreas de estudo, explicação sobre os algoritmos de aprendizagem de máquina utilizados no estudo, a acurácia, os softwares e linguagens de programação utilizados e por fim alguns trabalhos publicados no meio cientifico sobre o assunto. Na seção três mostra-se a metodologia utilizada, em seguida na seção quatro, o desenvolvimento do trabalho.

Na seção cinco mostra-se a descrição dos resultados obtidos e na secção 6 apresentam-se os principais pontos inferidos e os próximos passos para a complementação do trabalho acadêmico. Por fim, na seção 7 mostram-se oportunidades e sugestões para continuidade do trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Sensoriamento remoto

Sensoriamento remoto pode ser definido como técnica de obtenção de imagens terrestres, sem que haja um contato físico de qualquer espécie entre o sensor e o objeto analisado. Para enquadrar-se de fato na definição do termo, o princípio básico é que o sensor necessita estar numa distância remota do objeto e seguir os seguintes preceitos:

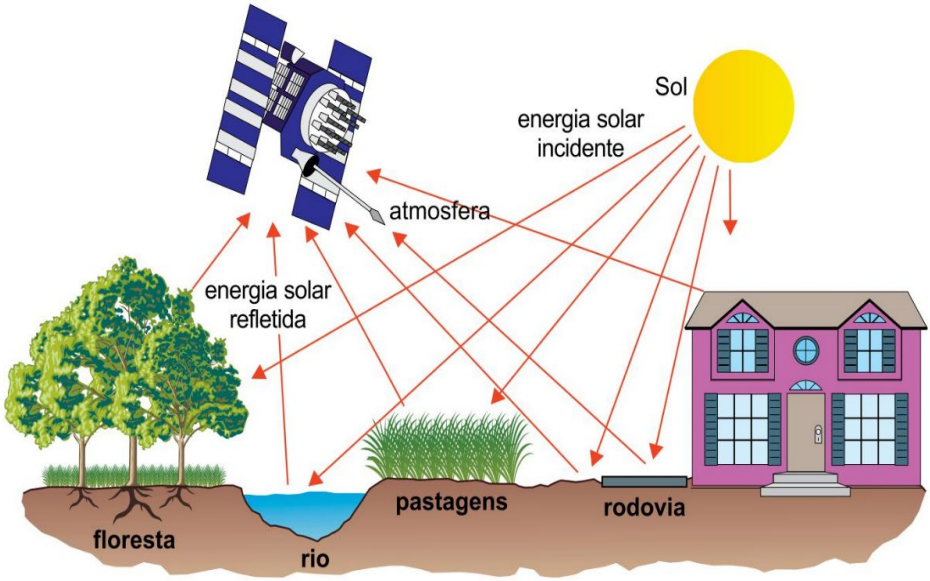
Exigência: não haver matéria entre o objeto e o sensor

Consequência: os dados podem ser transferidos pelo espaço vazio

Processo: a comunicação entre o objeto e sensor é feito por meio da radiação eletromagnética, que é a única forma de transporte de energia pelo espaço

Desta maneira, pode-se definir sensoriamento remoto unindo todos os preceitos, conforme a seguinte definição: Sensoriamento remoto é uma ciência que visa o desenvolvimento da obtenção de imagens da superfície terrestre por meio da detecção e medição quantitativa das respostas das interações da radiação eletromagnética com os materiais terrestres (MENESES; ALMEIDA, 2012). Na figura 1 observa-se um esquema ilustrativo que elucida o funcionamento do sensoriamento remoto, no qual a fonte energética ilustrada como o sol emite a energia solar que interage com as diversas feições no solo como água, floresta, rodovia, pastagens e atmosfera. A energia solar refletida é captada pelos sensores acoplados no satélite, que armazenam os dados e depois são transferidos e ajustados para uso público ou privado.

Figura 1. Figura ilustrativa exemplificando o sensoriamento remoto.



Fonte: IBGE, Atlas Geográfico Escolar (https://www.indagacao.com.br/2018/02/unesp-2017-2-fase-questao-08-o-sensoriamento-remoto-e-a-tecnica-que-permite.html)

O termo sensoriamento remoto começou a ser utilizado na década de 60 por Evelyn L. Pruit. Nesse período existia uma intensa corrida espacial entre os Estados Unidos e a antiga União Soviética, países que, durante a guerra fria, investiram fortemente em tecnologias que facilitassem a instalação de armas em pontos estratégicos e obter informações de maneira mais furtivas, como espionagem por meio de satélites.

Os primeiros satélites que se enquadravam nessa definição foram os satélites meteorológicos. O satélite TIROS – 1 (Television IR Operational Satellite) foi lançado pelo os Estados Unidos em 1960, no qual fazia imagens das coberturas de nuvens e também de algumas feições terrestres. A partir do TIROS – 1, foram aprimorados vários sensores e outras maneiras de fazer a aquisição dos dados, como por meio de satélites tripulados Mercury, Gemini e Apollo, no qual faziam imagens da terra usando câmeras fotográficas manuais.

Com o avanço tecnológico decorrente da corrida espacial, os sensores foram ainda mais aprimorados junto com as plataformas que eram acoplados. Sensores históricos e relevantes como o Landsat 1, denominado inicialmente como ERTS-1,

podia obter imagens multiespectrais com faixas no visível e infravermelho próximo com resolução espacial de 76 metros além de imagens termais, ou seja, muito além do que as câmeras fotográficas convencionais daquela época. Esses dados permitiram análises mais elaboradas das feições terrestres e inspiraram diversos países como Canadá, China, Japão e países europeus no desenvolvimento de sensores e satélites.

Esse avanço proporcionou a evolução dos diversos sensores atuais que são acoplados em satélites particulares com altas resoluções espaciais, como por exemplo as imagens da Planet que possuem 3 metros de resolução espacial ou também sensores hiperespectrais com as imagens disponibilizadas em sites de domínio público, no qual possuem centenas de bandas espectrais permitindo análises específicas do comportamento espectral terrestre.

2.1.1 Breve história do programa Landsat – Land Remote Sensing Satellite

A série Landsat teve início na década de 60 junto com diversos sensores e avanços durante a corrida espacial. O objetivo do programa era dedicado ao imageamento dos recursos naturais terrestres, no qual originou-se o nome inicial do projeto em ERTS – 1 (Earth Technology Satellite).

A partir deste primeiro satélite, foram construídos mais 7 satélites com adaptações e melhorias nos sensores. O Landsat 2 e o Landsat 3 foram lançados na década de 70 com os mesmos sensores do Landsat 1. O sensor RBV (Return Beam Vidicom) possuía resolução espacial de 80 metros e três faixas de bandas, variando da região do verde/azul, verde/vermelho e vermelho/infravermelho próximo. Apenas no LANDSAT 3 houve uma pequena melhoria, acoplando um sensor chamado RCA (Radio Corporation of America) no qual fazia imagens com uma banda pancromática específica com resolução espacial de 30 metros.

O Landsat 4, lançado em 1982, teve o sensor adaptado para mais suporte as pesquisas temáticas dos recursos naturais terrestres. Os sensores MSS (MultiSpectral Scanner) e TM (Thematic Mapper), permitiam o melhor imageamento da superfície terrestre, sendo o primeiro com separação de bandas nas

faixas do verde, vermelho, infravermelho próximo e infravermelho termal e o segundo com faixas no azul, verde, vermelho, infravermelho próximo, infravermelho médio e infravermelho termal. Essas características tiveram impacto mundial no desenvolvimento de sensores mais robustos e também na qualidade de imagens e estudos espaciais utilizando as imagens de sensoriamento remoto.

Em continuidade a série, o Landsat 5 foi enviado ao espaço com os mesmos sensores, sendo que o sensor TM manteve-se ativo até o ano de 2011, ao passo que o sensor MS ficou desligado por um tempo. Após superar a vida útil, foram desenvolvidos o Landsat 6 com o sensor ETM (Enhanced Thematic Mapper), que possuía inovação com a banda pancromática de 15 metros de resolução espacial, porém teve problemas no lançamento ao espaço.

O Landsat 7 teve uma melhoria no sensor, no qual foi denominado ETM + (Enhanced Thematic Mapper Plus). O sensor possibilitava melhores acurácias nos intervalos espectrais, além de possuir mais bandas como a banda termal e a pancromática, permitindo imagens coloridas terrestres de até 15 metros de resolução. Contudo, por falha no hardware, o sensor apresentou problemas de acurácia e correções de linha, sendo necessário utilizar algumas técnicas de interpolação e correções ao adquirir a imagem para análises.

Por fim, foi lançado o Landsat 8 com o sensor OLI (Operational Land Imager) e TIRS (Thermal Infrared Sensor). O sensor OLI é uma continuação das séries ETM com diversas melhorias como aumento das bandas espectrais e banda específica para detecção de nuvens.

As principais características dos sensores acoplados nos satélites podem ser observadas de maneira resumida na tabela 1.

Tabela 11. Características dos satélites da série Landsat. O L 7, embora esteja ativo*, não há mais recebimento de dados para o Brasil desde 2003.

Satélite	L 1	L 2	L 3	L 4	L 5	L 6	L 7	L 8
Lançamento	1972	1975	1978	1982	1984	1993	1999	2013
Situação	Inativo	Inativo	Inativo	Inativo	Inativo	Inativo	Ativo*	Ativo
Vida (Projetado)	1 ano	1 ano	1 ano	3 anos	3 anos	-	5 anos	5 anos
Sensores	RBV e MSS	RBV e MSS	RBV e MSS	MSS e TM	MSS e TM	ETM	ETM+	OLI e TIRS

Fonte: adaptado de EMBRAPA (https://www.embrapa.br/satelites-de-monitoramento/missoes/landsat)

2.1.2 Landsat 8

O Landsat 8 foi lançado em 2013 carregando os sensores OLI e TIRS. O satélite possui uma órbita sol síncrona, ou seja, acompanha a rotina do sol e permite que as imagens tenham uma iluminação consistente da superfície. O satélite está numa altitude aproximada de 705 km da terra. Tem capacidade de fazer 740 cenas durante o dia e completa uma volta na terra a cada 99 minutos.

O sensor OLI possui melhorias em relação aos sensores das series anteriores, principalmente relacionado a minimização de ruídos e aumento nas quantidades de bandas espectrais. Ao todo possui 9 bandas variando desde a faixa do azul costal até o infravermelho. Possui também uma banda específica para análises de nuvem e uma banda pancromática com resolução de 15 metros. Todas as demais bandas possuem 30 metros de resolução. O sensor TIRS possui 2 bandas espectrais com resolução de 100 metros. O sensor tem como objetivo medir a temperatura da superfície. Na tabela 2 pode – se observar as principais características do sensor OLI e TIRS acoplado no Landsat 8.

Tabela 12. Características das bandas espectrais dos sensores OLI[1] e TIRS [2] do LANDSAT 8.

Banda	Nome	Comprimento de onda (nm)	Res. Espacial
BANDA 01[1]	Costal Aerossol	430-450	30m
BANDA 02[1]	Azul	450-510	30m
BANDA 03[1]	Verde	530-590	30m
BANDA 04[1]	Vermelho	640-670	30m
BANDA 05[1]	Infravermelho Próximo	850-880	30m
BANDA 06[1]	Infravermelho de ondas curtas 1	1570-1650	30m
BANDA 07[1]	Infravermelho de ondas curtas 2	2110-2290	30m
BANDA 08[1]	Pancromática	500-680	15m
BANDA 09[1]	Cirrus	1360-1380	30m
BANDA 10[2]	Infravermelho de ondas longas 1	10300 - 11300	100m
BANDA 11[2]	Infravermelho de ondas longas 2	11500 - 12500	100m

Fonte: Elaborado pelo autor.

2.1.3 Radiância, Reflectância e Topo da Atmosfera

Existem diversos pontos que podem trazer ruídos para os dados obtidos por meio de sensores orbitais. Muitas vezes os próprios ruídos podem ser dados úteis para análises, mas na maioria das vezes é necessário aplicar alguns modelos algoritmos para minimizar a ação desses ruídos nos dados. Para isso, existem técnicas de correções como a correção atmosférica.

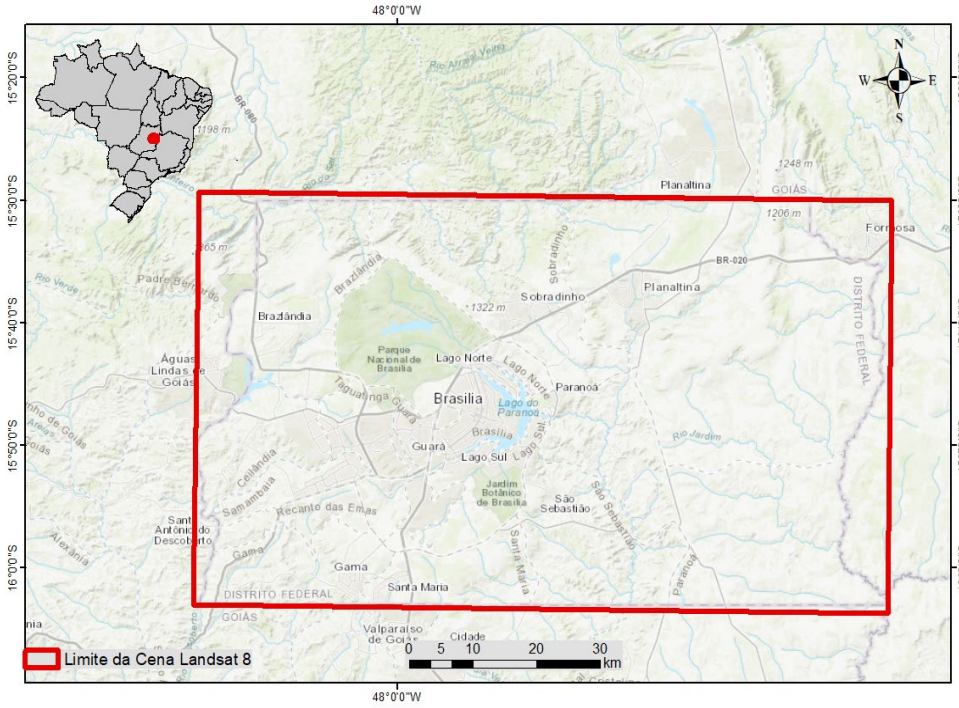
Segundo Meneses e Almeida (2012), a atmosfera pode afetar a radiância do da imagem em qualquer ponto por dois fatores: agindo como um refletor, por conter algumas partículas suspensas no ar ou até mesmo pelas características químicas que compõem, ou agindo como um absorvedor, atenuando a energia que ilumina o alvo.

A aplicação dos algoritmos de correção atmosférica ou minimização de ruídos vão depender do objetivo de utilização da imagem. Quando precisa-se fazer análises mais específicas dos alvos que estão sendo estudados, recomenda-se uma rigorosa análise do estado atmosférico durante o período de imageamento. Para alguns estudos, basta a aplicação de algoritmos já validados e aceitos pela literatura como a aplicação de métodos de correção atmosférica como TOA Reflectance ou Reflectância do Topo da Atmosfera, proposto por Chander, Markham e Helder (2009).

2.2 Área de estudo: Características do uso e cobertura da terra do Distrito Federal

A região escolhida para esse estudo foi a cena da coleção de imagens Landsat 8, abrangendo o território do Distrito Federal (DF) que possui 5.760,784 km² (IBGE, 2021) e localiza-se no centro do país (figura 2).

Figura 2. Limite da imagem de satélite Landsat 8 utilizada no estudo, localizado na região do Distrito Federal.



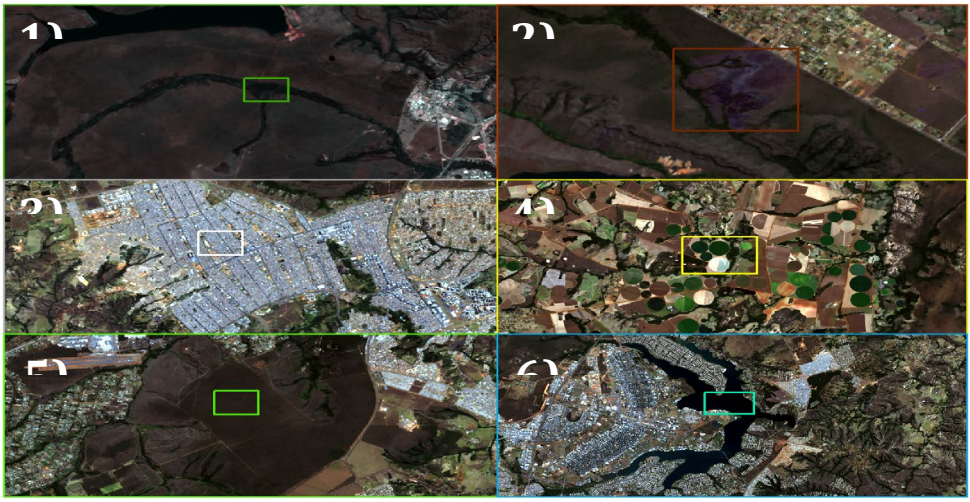
Fonte: Elaborado pelo autor.

O Distrito Federal apresenta diferentes fitofisionomias do cerrado que podem ser observadas principalmente nas áreas protegidas como a Reserva Ecológica de Águas Emendadas, Estação Ecológica do Jardim Botânico de Brasília e Parque Nacional de Brasília. A região do DF encontra-se também em três grandes bacias hidrográficas brasileira, sendo elas a bacia do São Francisco, do Paraná e do Amazonas. É uma região rica em nascentes e drenagens, com vegetações de mata ciliares, mas sem apresentar grandes rios, apenas grandes reservatórios como o Descoberto, Paranoá e Santa Maria que são utilizados para consumo doméstico e industrial de água. (ARAÚJO FILHO, 2005). Embora há um crescimento acelerado das áreas urbanas, já existem diversas áreas urbanas consolidadas como o Plano Piloto, Ceilândia, Taguatinga e Planaltina.

Uma importante característica a ser citada é a diferença de arborização nas diversas regiões do DF, sendo o Plano Piloto a região mais arborizada, ao passo que

as regiões do entorno possuem maior densidade de áreas edificadas sem grandes quantidades de árvores. Outro destaque é a região do PAD-DF (Plano de Assentamento Dirigido do Distrito Federal), localizado em um extenso platô ao leste, abrigando inúmeras áreas produtivas com grandes quantidades de pivôs centrais irrigados, áreas com vegetação reflorestada e também áreas destinadas à pecuária. Na figura 3 é possível observar as diferentes características do bioma Cerrado na região do Distrito Federal.

Figura 3. Mosaico com as diferentes características da área de estudo. No quadro número 1 mostra-se uma área de floresta densa, comum em matas de galeria, no quadro número 2 mostra-se uma cicatriz de queimada, no quadro número 3 observa-se uma área urbana densa, no quadro número 4 uma área de uso agropecuário, no quadro número 5 uma área de savana e por fim, no quadro número 6, uma área de corpo hídrico.



Fonte: Elaborado pelo autor.

Diante de todas essas características, o Distrito Federal evidencia-se como uma boa área de estudo pois abriga diferentes tipos de uso do solo em um território pequeno quando comparado aos demais estados do Brasil, podendo ser um excelente piloto para estudo de classificação do uso e cobertura do solo do bioma Cerrado, com a possibilidade do estudo ser replicado em outras regiões.

2.3 Aprendizagem de Máquina

O aprendizado de máquina é uma área da Inteligência artificial que tem como objetivo de desenvolver técnicas computacionais sobre o aprendizado e aplicar e

adquirir o conhecimento de forma automática. Segundo Mohri, Rostamizadeh e Talwalkar (2018) pode-se definir como métodos computacionais que utilizam por meio de experiências passadas, sendo basicamente adquiridas de conjuntos de dados como imagens ou planilhas, de forma que melhora performances, classificações, acurácias ou até mesmo geram previsões.

Existem inúmeros algoritmos e modelos para os diversos setores do aprendizado de máquina. Os algoritmos classificadores permitem observar padrões, com base em uma certa característica e agrupa-los em classes.

É comum observar dois tipos de algoritmos classificadores, sendo um não supervisionado, que segundo Faceli et al. (2021) pode se entender como o os algoritmos observando os padrões por meio de estatísticas e realizando agrupamentos, sumarizações ou associações e o aprendizado supervisionado, no qual o há um conjunto de dados já agrupados ou classificados, que são utilizados na entrada do algoritmo, treinando o modelo e gerando o resultado conforme essa classificação.

Esses algoritmos classificadores tornam-se muito útil para aplicações em imagens de satélite, no qual existem padrões espectrais específicos que podem ser classificados de maneira automatizada.

2.4 Algoritmo Máximo Verossimilhança

O algoritmo de classificação supervisionado por Máxima Verossimilhança é amplamente utilizado em imagens de sensoriamento remoto. O algoritmo analisa a ponderação das distâncias entre as médias dos valores dos pixels das classes, por meio de parâmetros estatísticos, considerando que todas as bandas têm distribuição gaussiana, calculando a chance de um determinado pixel pertencer a uma classe específica (INPE, 2008). Como trabalha com a alta probabilidade de um determinado valor do pixel ser destinado a uma classe, sugere-se que as amostras de treinamento tenham um número alto de quantidade de pixels para aumentar a eficácia do algoritmo (MENESES; ALMEIDA, 2012)

2.5 Algoritmo Máquina de Vetor de Suporte

O algoritmo classificador de Máquina de Vetor de Suporte (Support Vector Machine - SVM) é definido como uma técnica computacional de aprendizado que tem como objetivo reconhecer padrões nas amostras e separa-las em classes, otimizando o cálculo das margens (o que define a separação das classes) de maneira linearmente. O algoritmo, que foi implementado por meio da teoria estatística de aprendizagem de Vapnik (1995), funciona como uma máquina linear, na qual constrói um vetor na superfície de decisão, de forma que esse vetor possua a melhor distancia de separação entre as classes.

Máquina de Vetor de Suporte é um ótimo classificador considerado por muitos pesquisadores como uma boa opção de algoritmo para aplicação no sensoriamento remoto, observado em vários estudos (BROWN; LEWIS; GUNN, 2000; MELGANI; BRUZZONE, 2004).

2.6 Algoritmo Floresta Aleatória

O Random Forest, também conhecido Floresta Aleatória, é um algoritmo de aprendizagem de máquina supervisionado como os demais algoritmos utilizados nesse estudo, que pode ser empregado em classificações ou regressões. O algoritmo funciona como uma combinação das árvores de decisões com a aleatoriedade, com o objetivo de melhorar a precisão.

O classificador utiliza como princípio a árvore de decisão, que segundo Prates (2018) é uma ferramenta de apoio a tomada de decisão que usa um gráfico no qual faz uma analogia visual a uma árvore, sendo que cada ramo da árvore indica as condições e probabilidades para se chegar em cada resultado.

No algoritmo de floresta aleatória, são utilizados um conjunto de árvores de decisão com a combinação da aleatoriedade, no qual são repetidas várias vezes para o treinamento do classificador. As várias árvores de decisão vão construindo um conjunto de algoritmos de treinamento com diversas amostras aleatórias, na qual cada árvore seleciona um pixel e classifica de acordo com a classe temática. Como são vários classificadores, há uma maior probabilidade de aumentar a acurácia do

resultado. As classes mais votadas das árvores são retomadas então para o classificador. (HAN; KAMBER; PEI, 2011).

2.7 Análises de acurácia e índice kappa

Acurácia, segundo Mikhail e Ackermann (1976) é o grau de proximidade de uma estimativa com o seu parâmetro verdadeiro. A acurácia pode ser observada de três maneiras, como a Acurácia Global, Acurácia do Usuário e Acurácia do Produtor.

A acurácia do produtor (equação 1) são as porções de classes reais que foram corretamente atribuídas as classes corretas pelos classificadores. A acurácia do usuário (equação 2) basicamente diz ao usuário com que frequência a classe classificada estará presente na classe real. A acurácia global (equação 3) refere-se a medida percentual da quantidade de amostras corretamente classificadas em relação ao total de amostras disponíveis (MAPBIOMAS,2022).

Segundo Congalton e Green (2019) a matriz de confusão é uma maneira de comparar entre a verdade do campo e o resultado da classificação. A matriz de confusão, exemplificada na tabela 3, mostra como são dispostas as classes. Na tabela os resultados do total de colunas ∑(a), ∑(q), ∑(g) corresponderia as acurácias do produtor, os resultados do total de linhas, ∑*(a), ∑*(q), ∑*(g) correspondem a acurácia do usuário e ∑ (total) a acurácia global.

Tabela 13. Tabela de confusão modelo para comparação dos resultados da classificação.

	Matriz de confusão - Exemplo				
	Classes	Classe 1(a)	Classe 2(q)	Classe 3(g)	Total linhas
Previsto	*Classe 1 (a)	aa	aq	ag	∑*(a)
	*Classe 2 (q)	aq	qq	qg	∑*(q)
	*Classe 3 (g)	ga	gq	gg	∑*(g)
	Total colunas	∑(a)	∑(q)	∑(g)	∑(total)

Fonte: Adaptado Congalton e Green (2019).

Equação 1. Acurácia do produtor

$$acuráciadoprodutor = \frac{n_{aa,aq,gaouaq,qqgqouaq,qg,\gg}}{\sum(a,q,g)}$$

No qual aa, aq ga correspondem as classes combinadas observadas na tabela 3 e ∑ (a, q, g) a soma da coluna.

Equação 2. Acurácia do usuário.

$$acuráciadousuário = \frac{n_{aa,aq,gaouaq,qqgqouaq,qg,\gg}}{\sum(a,* q,* g)}$$

No qual aa, aq ga correspondem as classes combinadas observadas na tabela 3 e ∑ (*a, *q, *g) a soma das linhas.

Equação 3. Acurácia global.

$$exatidãoglobal = \frac{\sum_{i=1}^{k} n_{aaqq,\gg}}{\sum total}$$

No qual aa, qq, gg correspondem as classes combinadas observadas na tabela 3 e ∑(total) a soma de todos os valores da tabela.

Outro importante índice para mensurar a qualidade dos resultados classificadores é índice Kappa. O índice Kappa é um índice proposto por Cohen (1960) que avalia o nível de concordância ou reprodutibilidade entre conjuntos de dados. Já exatidão global é a estimativa de acerto de todos os indicadores. A equação do índice Kappa pode ser observada na equação 4.

Equação 4. Índice de Kappa.

$$índicekappa = \frac{P(O) - P(E)}{1 - P(E)}$$

No qual P(O) é a proporção observada das concordâncias, P(E), a proporção observada das concordâncias.

A avaliação dos valores resultantes da aplicação estatística do índice Kappa foi avaliado de acordo classificação sugerida por Landis e Koch (1977). Os parâmetros da classificação encontram-se na tabela 4.

Tabela 14. Tabela com a classificação do índice Kappa para cada intervalo de valor.

Índice kappa (K)	Qualidade
K = 0,2	Ruim
0,2 - 0,4	Razoável
0,4 - 0,6	Bom
0,6 - 0,8	Muito Bom
0,8 - 1	Excelente

Fonte: Adaptado de Landis e Kock (1977).

2.8 Software ArcMap – ArcGIS

O ArcGIS é um conjunto de softwares de sistemas de informação geográfica que funciona de maneira integrada. O software é desenvolvido pela empresa ESRI (Environmental Systems Research Institute). Os softwares presentes no conjunto permitem fazer análises espaciais, por meio de ferramentas no software ArcMap, manipulação e organização de bases de dados, como no software ArcCatalog, e visitar diversas ferramentas úteis, por meio do software ArcToolBox para manipulação e modelagem do dado.

A arquivo shapefile é o formato mais comum utilizado no software, contendo informações de arquivo vetoriais e tabelas de atributo. Outro arquivo comum são os de formato raster como tiff, grid, jpeg, que geralmente são dados matriciais como imagens de satélite, cartas topográficas ou imagens

2.9 Linguagem de programação R

Existem diversos softwares e pacotes estatísticos para análise de dados. A maioria desses softwares possuem custos de aquisição ou pacotes que geralmente são elevados. Por isso, observa-se cada vez mais a busca, incentivos e aprimoramento no uso de softwares livres (CHAMBERS, 2008).

Dentre os ambientes de domínios públicos, o ambiente R tem ganhado cada vez mais notoriedade no âmbito mundial. Pode-se definir o R como uma linguagem de alto nível e um ambiente para análises de dados e geração de gráficos (CRAWLEY, 2007).

Atualmente existem diversos pacotes e bibliotecas que permitem utilizar modelos complexos de aprendizagem de máquina, redes neurais ou para a manipulação de dados de maneira mais simplificada. Nesse estudo utilizou as seguintes bibliotecas:

“e1071”: pacote com diversas funções estatísticas, análises de classes, classificador “naive Bayes” e outros algoritmos de aprendizagem de máquina.

“Raster”: possui funções para criação, leitura, manipulação e escrita de dados matriciais no formato raster. Possui algumas funções ligadas ao geoprocessamento, como leitura de dados vetorizados no formato “shapefile”

“caret”: abreviação para o nome "Classification And REgression Training” (Classificação e Regressão para Treinamento). Possui diversas funções estatísticas e também auxílios na manipulação de dados e pré-processamento.

randomForest: possui o algoritmo de classificação e regressão baseado em floresta de árvores usando aleatoriedades, baseado na publicação de Breiman (2001)

RStoolbox: pacote que oferece diversas funções para processamento de imagens de sensoriamento remoto. Possui funções para pré-processamentos, analises, visualização gráfica e classificadores.

2.10 Trabalhos correlatos

A classificação de imagens de satélite é o processo de associar os pixels da imagem a determinadas classes que podem representar objetos do mundo real. A prática é amplamente utilizada no meio cientifico principalmente para estudos relacionados ao uso e cobertura do solo e dinâmica do uso da terra.

No estudo de Alencar et al., (2020) no qual realizou-se uma classificação do uso e cobertura do solo para o bioma Cerrado utilizando os algoritmos de floresta aleatória em imagens de satélite Landsat para os anos de 1985 a 2017, foram identificados desafios como a sazonalidade e heterogeneidade dos tipos de vegetação. Para isso foram utilizadas amostras de referências de outras versões de mapeamento do cerrado como MAPBIOMAS e TerraClass. O trabalho permitiu observar as mudanças do uso de solo, no qual houve grande perda da vegetação para

atividades antrópicas. Os resultados tiveram uma acurácia global de 87%, considerando as amostras das coleções anteriores do MAPBIOMAS e 71% considerando o TerraClass como referência. Isso demonstra a possibilidade de extrair amostras em mapas de referência para a classificação de grandes intervalos temporais e grandes áreas.

Embora utilizado em uma área de vegetação completamente diferente do bioma Cerrado, Deilmai, Ahmad e Zabihi (2014) realizou a comparação entre dois algoritmos de classificação usados nesse estudo, a máquina de vetor de suporte e máximo verossimilhança. Para a classificação foram utilizadas amostras coletadas na própria imagem Landsat, no qual foi utilizada para classificação. Com algumas classes semelhantes utilizadas neste trabalho, como área urbana, floresta e corpo hídrico, o algoritmo de máximo verossimilhança obteve 78,33% de acurácia global e 0.65 de índice Kappa, sendo enquadrado como bom, ao passo que o algoritmo de máquina de suporte de vetor obteve acurácia global superior a 90% e índice Kappa de 0.86, demonstrando a potencialidade do algoritmo nas classificações de áreas complexas, com exceção do algoritmo de máximo verossimilhança, no qual foi salientado que não recomenda-se a aplicação em áreas muito complexas e com grandes padrões espectrais, já que exige-se maior coleta de amostras.

Carvalho et al. (2019) utilizou o algoritmo de floresta aleatória para classificar a vegetação do Parque Nacional da Chapada das Mesas utilizando imagens de alta resolução espacial. O parque localiza-se no bioma Cerrado e apresentam os mesmos desafios de sazonalidade e heterogeneidade da vegetação, porém sem áreas edificadas e com predominância de áreas naturais. Foram encontrados valores de acurácia global de 71% e 0.66 de índice Kappa, no qual segundo o autor pode ser considerado bom tendo em vista a heterogeneidade da área.

No estudo de Cho, et al. (2021) foram aplicados o algoritmo de floresta aleatória, utilizando imagem de satélite Landsat 8 para três municípios no estado do Tocantins. Os municípios, inseridos no bioma Cerrado, mostraram ser útil para o estudo por conta da importância na conservação das áreas naturais para biodiversidade e por ser parte da fronteira da expansão agropecuária no Brasil. Foram coletadas amostras para sete classes, como áreas naturais, urbanizadas, solo

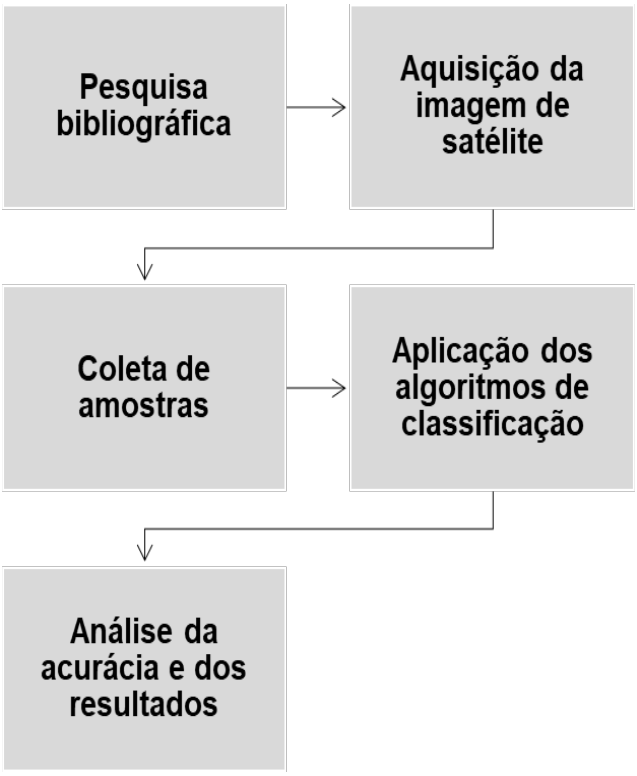
exposto e agricultura. Foram encontradas acurácias globais de 84% e índice Kappa de 0,64, ou seja, “muito bom” de acordo com Landis e Koch (1977). Outro ponto importante observado no estudo foi o padrão encontrado de uma subestimação na classe de agricultura e de uso e cobertura do solo, no qual as áreas agrícolas foram incorporadas equivocadamente a classes de solo exposto, vegetação natural e pastagem.

Os mapeamentos de savana sempre foram um desafio, destacados em diversos estudos como em Alencar et al. (2020), Pinto et al. (2019) Carvalho et al. (2019) que salientam acerca da heterogeneidade e sazonalidade das áreas de Cerrado que aumentam a chance de confusões nas classificações devido a grande quantidade de padrões espectrais.

3. METODOLOGIA DO TRABALHO

Este trabalho foi realizado utilizando a metodologia de pesquisa aplicada descritiva. Foram coletados dados, aplicados algoritmos classificadores e o resultado foi analisado e interpretado. Essa abordagem tem como objetivo observar os algoritmos de aprendizagem de máquina aplicados classificação de imagem e suas acurácias. Assim, a elaboração do trabalho foi dividida nas seguintes etapas apresentadas na figura 4.

Figura 4. Etapas do método aplicado no trabalho



Fonte: Elaborado pelo autor.

Primeira Etapa: Revisão bibliográfica sobre aplicação de algoritmos de aprendizagem de máquina para classificação de imagens de satélite, observando quais algoritmos são mais empregados e seus desafios.

Segunda Etapa: Início do processo de aquisição da imagem de satélite, obtida por meio da plataforma Google Earth Engine.

Terceira Etapa: Coleta de amostras para treinamento e teste de acurácia com base em interpretação visual da imagem, utilizando o software ArcGIS.

Quarta Etapa: Aplicação dos algoritmos de classificação, utilizando a linguagem de programação R.

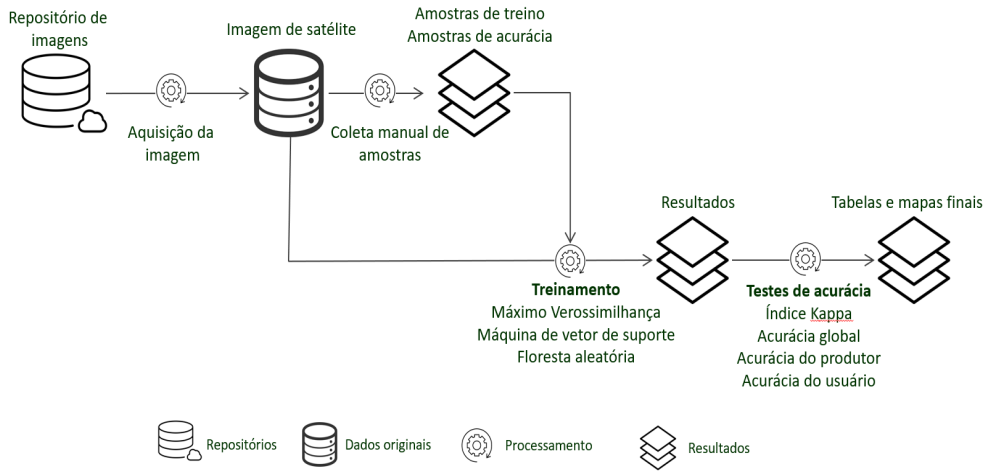
Quinta Etapa: Observação visual dos mapas resultantes da classificação e aplicação dos testes de acurácia para mensuração da qualidade dos resultados. Os

resultados foram comparados observando os principais erros de classificação entre as classes, observados nas matrizes de confusão.

4. DESENVOLVIMENTO DO PROJETO

Os tópicos a seguir mostram de maneira mais detalhada as etapas utilizadas neste trabalho. A figura 5 mostra o diagrama das etapas.

Figura 5. Diagrama com as etapas de processos aplicados no trabalho.



Fonte: Elaborado pelo autor.

4.1 Aquisição da imagem

A imagem de satélite utilizada no estudo é proveniente do sensor OLI abrigado no satélite LANDSAT 8. A imagem foi obtida por meio da plataforma Google Earth Engine, repositório que permite acesso a milhares de imagens de satélites. A escolha da imagem foi determinada pela porcentagem de quantidade de nuvens, sendo considerado o máximo de 0 a 10%, no dia 19 do mês de julho de 2019, referente ao período de seca na região.

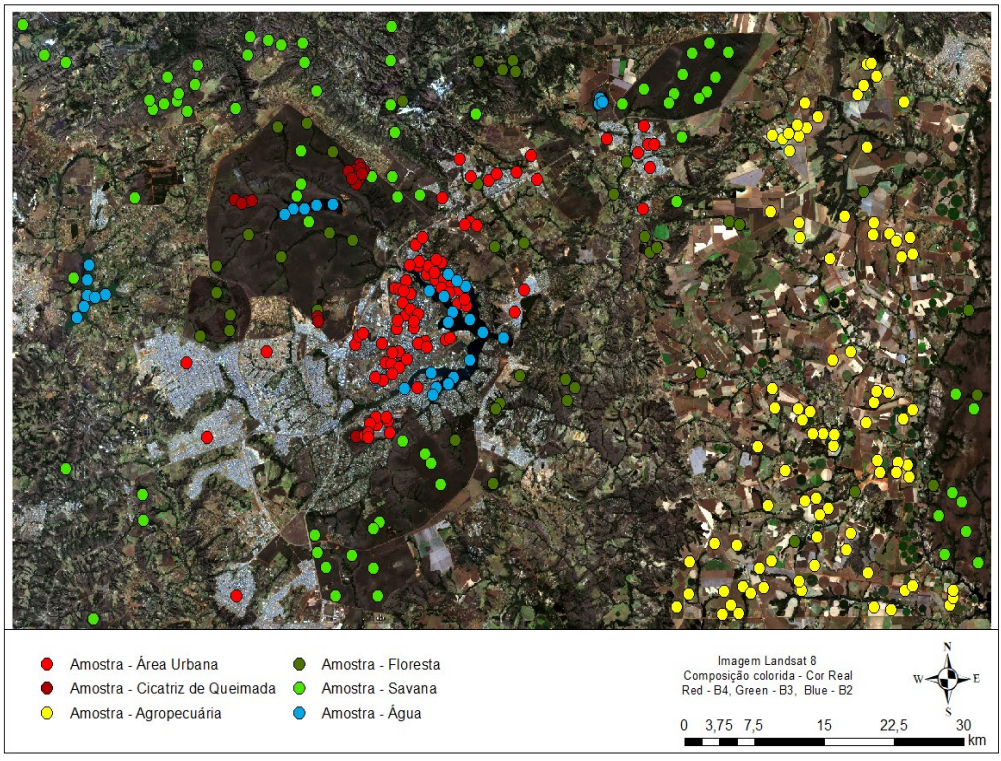
A imagem Landsat 8 possui 30 metros de resolução espacial e suas características espectrais estão listadas na tabela 1. A imagem utilizada no estudo já está calibrada e corrigida atmosfericamente ao topo da atmosfera.

Foram escolhidas para análise as bandas do 1 ao 7 por serem bandas que conseguem pegar as diferentes assinaturas espectrais como vegetação, água, áreas urbanas, áreas de agropecuárias e queimadas.

4.2 Coleta de amostras

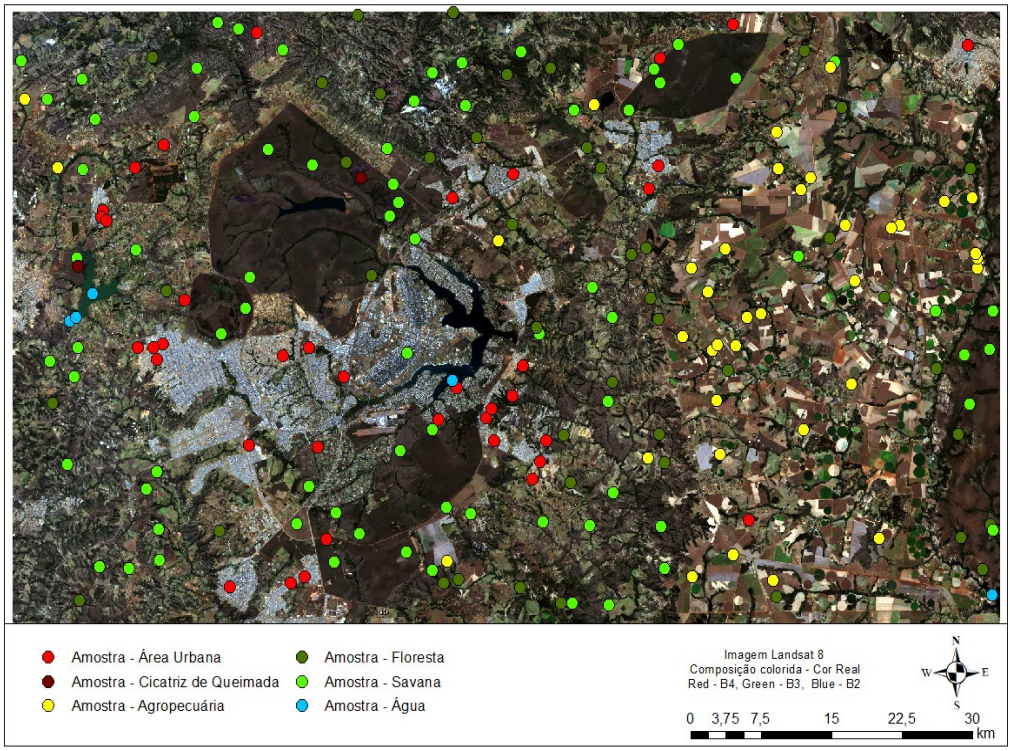
Foram gerados 340 pontos aleatórios (por meio do software ARCMAP) no qual foram classificados manualmente de acordo com a interpretação visual da imagem. Os pontos foram divididos nas classes de Floresta, Savana, Área Urbana, Água, Agropecuária e Queimada para as amostras de treino. Também foram gerados 200 novos pontos aleatórios para as amostras de acurácia. Todos os pontos foram classificados pela mesma pessoa, no caso o autor do trabalho e no mesmo dia. Na figura 6 pode-se observar a disposição espacial das amostras coletadas para treino e na figura 7 as amostras de acurácia.

Figura 6. Amostras coletadas para a aplicação dos algoritmos.



Fonte: Elaborado pelo autor.

Figura 7. Amostras coletadas para verificação da acurácia dos algoritmos.



Fonte: Elaborado pelo autor.

4.3 Aplicação dos algoritmos utilizando a linguagem R

Primeiramente, foi utilizado o pacote “raster” para carregar as imagens bandas da imagem de satélite. Por meio da função “brick” as bandas foram empilhadas para que pudessem ser trabalhadas de maneira integral (Tabela 5). Foram carregados os pontos por meio da função “shapefile” e em seguida foi aplicado a função “crs” para certificar que as amostras e a imagem de satélite estavam na mesma projeção geográfica (tabela 5).

Tabela 15. Tabela com os códigos para carregamento da imagem, pontos e comparação da projeção geográfica.

```
library(raster) # carregando a biblioteca
img <- brick("/LANDSAT8BSB.tif") #carregando as bandas e empilhando
shp <- shapefile("training.shp") # carregando pontos para treino
compareCRS(img,shp) # Comparando projeção geográfica
```

Fonte: Elaborado pelo autor.

A aplicação do algoritmo de máximo verossimilhança foi feita por meio do pacote “RStoolbox”, por meio da função “superClass”. O algoritmo de máquina de vetor de suporte utilizado foi obtido por meio da biblioteca “e1071”, utilizando a função “tune”. O algoritmo de floresta aleatória foi utilizado por meio da biblioteca “randomForest” e a função “tuneRF” (tabela 6).

Tabela 16. Tabela com os códigos para aplicação dos algoritmos classificadores.

```
library("RStoolbox") #biblioteca para máximo verossimilhança
MLC<-superClass(img,shp, responseCol="ID", model = "mlc", tuneLenght=1,trainPartition = 0.7) #aplicando algoritmo
library(e1071) #biblioteca para máquina de vetor de suporte
svmgs <- tune(svm, train.x = smp[-ncol(smp)], train.y = smp$cl, type = "C-classification", kernel = "radial", scale = TRUE, ranges = list(gamma = gammas, cost = costs), tunecontrol = tune.control(cross = 5) #aplicando algoritmo
library(randomForest) #biblioteca para floresta aleatória
rfmodel <- tuneRF(x = smp[-ncol(smp)], y = smp$cl, sampsize = smp.size, strata = smp$cl, ntree = 250, importance = TRUE) #aplicando algoritmo
```

Fonte: Elaborado pelo autor.

4.4 Teste de Acurácia

No teste de acurácia foram utilizadas as bibliotecas “raster” para carregar os resultados classificados e a nuvem de pontos interpretadas para a acurácia. Os pontos foram utilizados para extraírem as informações das imagens já classificadas e assim poder comparar com os pontos da amostra de acurácia. Na tabela 7 observa-se o processo de carregamento das imagens resultados e extração. Após a extração, foi

utilizado a função “confusionMatrix” da biblioteca “caret”. A função fornece as matrizes de confusão e também o resultado do índice kappa.

Tabela 17. Tabela com os códigos para testes de acurácia.

library(raster) # biblioteca para carregar as imagens e os pontos de validação
library(caret) # biblioteca para a função da matriz de confusão e índice kappa
compareCRS(img,shp) # Comparando projeção geográfica
img.classifiedSVM <- raster("SVM_classification.tif") #carregando imagem, nesse caso a máquina de vetor de suporte
shp.valid<-shapefile("acuracia.shp") # carregando pontos para teste de acurácia
predictedSVM<-as.factor(extract(img.classifiedSVM, shp.valid)) #extraindo os dados da imagem por meio dos pontos
SVMACC<-caret::confusionMatrix(reference,predictedSVM) #aplicação da função de matriz de confusão e índice kappa

Fonte: Elaborado pelo autor.

5. ANÁLISE DOS RESULTADOS

Na tabela 8 constam os valores organizados na matriz de confusão para o algoritmo de Máximo Verossimilhança, seguido do mapa na figura 8.

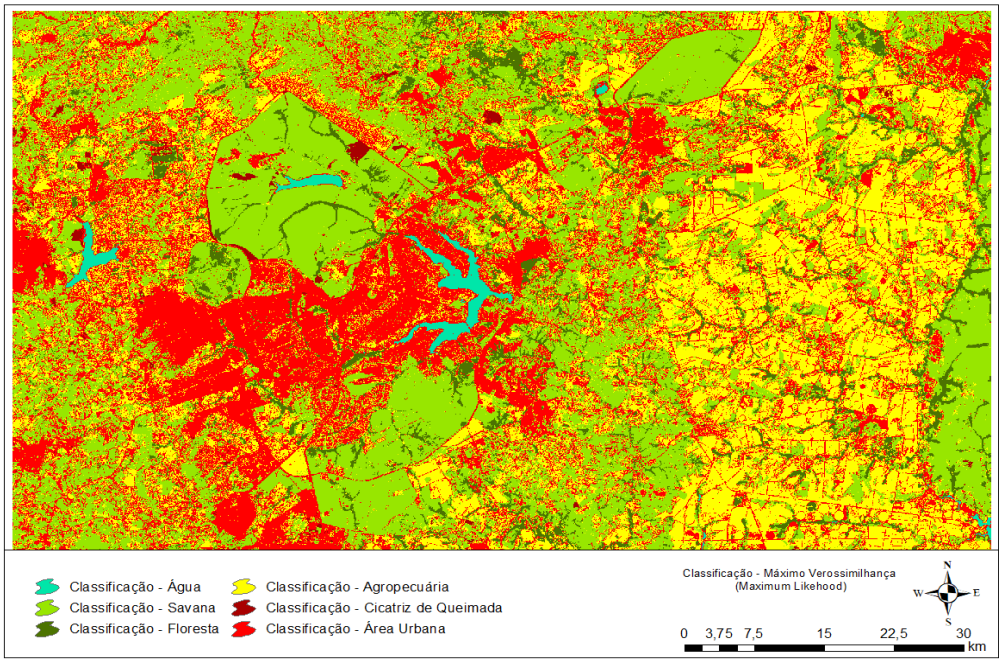
Tabela 18. Tabela com a matriz de confusão para o algoritmo de classificação de Máximo Verossimilhança.

Matriz de confusão Máximo Verossimilhança

Dados classificados	Classes	Área Urbana	Queimada	Agropecuária	Floresta	Savana	Água	Soma linhas
	Área Urbana	37	0	1	0	0	0	38
	Queimada	0	2	0	0	0	0	2
	Agropecuária	5	0	30	0	2	0	37
	Floresta	2	0	2	26	14	0	44
	Savana	10	0	11	2	51	0	74
	Água	0	0	0	0	0	5	5
	Soma colunas	54	2	44	28	67	5	200

Fonte: Elaborado pelo autor.

Figura 8. Mapa de classificação do algoritmo de classificação verossimilhança.



Fonte: Elaborado pelo autor.

Observa-se na tabela 8, classificações equivocadas nas áreas urbanas, confundindo principalmente com áreas de agropecuária e savana. Algumas regiões arborizadas como a região do Plano Piloto e Lago Norte mostraram-se predominantemente classificado como área urbana, apesar de ser uma região com significativa arborização. As áreas de corpos hídricos e áreas queimadas foram bem classificadas, sem nenhuma confusão com outra classe. Observaram - se grandes áreas de manchas urbanas classificadas nas regiões de relevo ondulado, principalmente nas sombras de relevo na parte norte da imagem e também em áreas de savana seca, como em algumas regiões do parque nacional. Algumas regiões com matas de galeria menos densas não foram classificadas como floresta densa e sim como savana. Algumas manchas urbanas foram encontradas em áreas de unidades de conservação, no qual apresentam predominantemente savana e florestas densas. As áreas de agropecuária foram bem mapeadas na porção leste, contudo também houveram confusões com as áreas urbanas, florestas e também com áreas de savana mais secas.

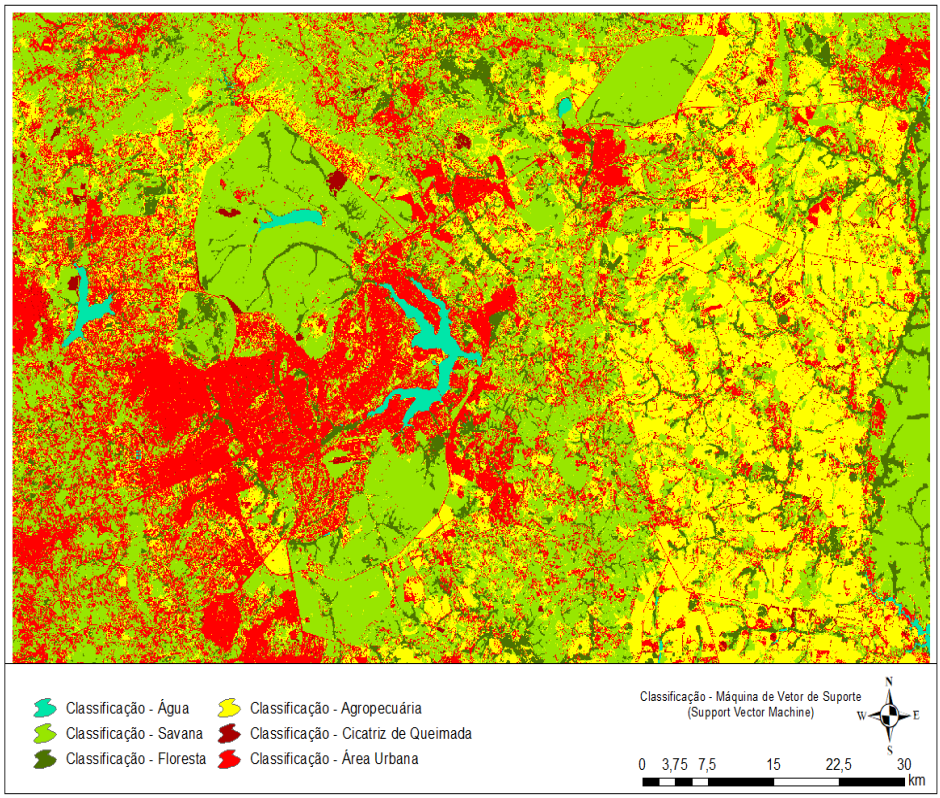
Ressalta-se, na tabela 9 os valores da matriz de confusão para o algoritmo de Máquina de vetor de suporte seguido do mapa (figura 9).

Tabela 19. Tabela com a matriz de confusão para o algoritmo de classificação de Máquina de vetor de suporte.

	Matriz de confusão Máquina de Vetor de Suporte						
	Classes	Área Urbana	Queimada	Agricultura	Floresta	Savana	Água	Soma linhas
Dados classificados	Área Urbana	33	0	5	0	0	0	38
	Queimada	0	2	0	0	0	0	2
	Agricultura	4	0	28	0	5	0	37
	Floresta	1	0	1	29	13	0	44
	Savana	8	0	9	2	55	0	74
	Água	0	0	0	0	0	5	5
	Soma colunas	46	2	43	31	73	5	200

Fonte: Elaborado pelo autor.

Figura 9. Mapa de classificação do algoritmo de classificação máquina de vetor de suporte.



Fonte: Elaborado pelo autor.

Na classificação da Máquina de Vetor de Suporte constatou-se uma menor confusão na classificação das manchas urbanas, principalmente entre as classes de floresta, agricultura e savana. Essas áreas podem ser observadas na região leste da imagem e, ao ser comparo com o algoritmo de máximo verossimilhança, observa-se uma menor quantidade de área urbana (áreas vermelhas) classificadas na região, no qual predomina agricultura, savana e floresta. Além disso, não foram constatadas confusões na classificação de áreas queimadas e corpos hídricos. As áreas de savana preservada e florestas densas mostraram melhores classificações e menores quantidades de equívocos entre as classes de agropecuária e áreas urbanas quando comparadas com o algoritmo de máximo verossimilhança.

Já os valores da matriz de confusão para o algoritmo de Floresta Aleatória são informados na tabela 10, seguido do resultado explicito no mapa (figura 10).

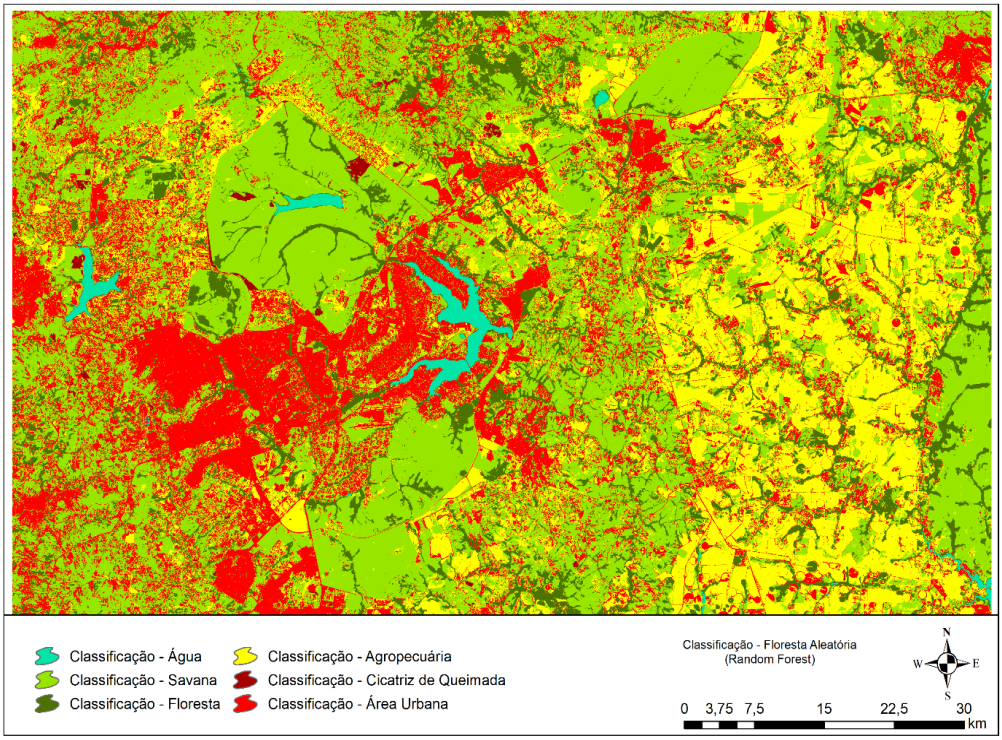
Tabela 20. Tabela com a matriz de confusão para o algoritmo de classificação de Floresta Aleatória.

Matriz de confusão Floresta Aleatória

	Classes	Área Urbana	Queimada	Agropecuária	Floresta	Savana	Água	Soma linhas
Dados classificados	Área Urbana	37	0	1	0	0	0	38
	Queimada	0	2	0	0	0	0	2
	Agropecuária	2	0	30	0	5	0	37
	Floresta	0	1	1	33	9	0	44
	Savana	7	0	8	1	58	0	74
	Água	0	0	0	0	0	5	5
	Soma colunas	46	3	40	34	72	5	200

Fonte: Elaborado pelo autor.

Figura 10. Mapa de classificação do algoritmo de classificação de Floresta Aleatória.



Fonte: Elaborado pelo autor.

De maneira semelhante aos outros algoritmos, as áreas de corpos hídricos não apresentaram erros na classificação, contudo, o algoritmo de Floresta Aleatória foi o único em que as áreas queimadas mostraram equívoco na classificação com áreas de floresta. Quanto às áreas urbanas, houve consideravelmente menores erros com as classes de agropecuária e savana. Já as áreas de florestas também demonstraram uma quantidade de equívocos consideravelmente menores na classificação, apresentando apenas um erro na classe de savana onde deveria ser floresta. As áreas de agropecuária, que ganham destaque principalmente na porção leste da imagem, também obtiveram uma menor quantidade de classificação errada em relação as classes de savana, áreas urbanas e florestas.

Para facilitar a comparação entre os três algoritmos, a tabela 11 destaca os resultados da acurácia total, acurácia do produtor, acurácia do usuário e índice Kappa. O índice Kappa para os algoritmos de máximo verossimilhança e máquina de vetores de suporte situa-se na média de 0,67, deste modo, de acordo com a classificação sugerida por Landis e Koch (tabela 3), os índices possuem a qualidade de classificação no parâmetro “Bom”. Já o classificador de árvore de decisão possui um valor de 0,76, que embora ainda esteja dentro da amplitude da classificação “Bom”, demonstra valores mais próximos ao patamar de qualidade “Muito Bom”.

Observando a acurácia do produtor, nota-se que o algoritmo de árvore aleatória obteve melhores resultados, com uma porcentagem menor apenas para a classe de queimada, na qual houve confusão com a classe de floresta. Comportamento semelhante foi obtido nas acurácias do usuário, porém os resultados das árvores aleatórias foram mais próximos do algoritmo de máximo verossimilhança.

Tabela 21. Resultados obtidos para cada classe por meio dos algoritmos de classificação supervisionada aplicados.

	Máximo Verossimilhança		Máquina de Vetores de Suporte		Floresta Aleatória	
Classes	Acurácia do produtor	Acurácia do usuário	Acurácia do produtor	Acurácia do usuário	Acurácia do produtor	Acurácia do usuário
Floresta	93%	59%	94%	66%	97%	75%
Savana	76%	69%	75%	74%	81%	78%
Área Urbana	69%	97%	72%	87%	80%	97%
Água	100%	100%	100%	100%	100%	100%
Agropecuária	68%	81%	65%	76%	75%	81%
Queimada	100%	100%	100%	100%	67%	100%
Acurácia Total	75,5%		72%		82,5%	
Kappa	0,6744		0,6786		0,766	

Fonte: Elaborado pelo autor.

O resultado do algoritmo de Floresta Aleatória apresentou melhor desempenho em comparações com os demais algoritmos.

6. CONCLUSÃO

O estudo permitiu compreender acerca das diferenças nas acurácias dos três algoritmos, além de identificar as semelhanças na classificação de algumas classes.

Notaram-se confusões nos resultados dos três algoritmos entre as classes de áreas urbanas, agropecuárias e formação savânicas. Essas confusões também se mostraram presentes em outros estudos, ratificando-se, portanto, os resultados encontrados e justificando-se devido ao fato de os padrões espectrais no período de seca poderem ser semelhantes entre as classes, causando confusão na classificação.

O algoritmo de Floresta Aleatória obteve valores de índice Kappa de 0,766 e 82,5% de acurácia global, superiores aos resultados de acurácia e índice Kappa dos outros. O algoritmo SVM apresentou bom valor de acurácia global de 72%. Contudo, apresentou menor valor de índice Kappa, sendo de 0,678, comparado ao de

Floresta Aleatória. O resultado do algoritmo de MAXVER indicou um valor intermediário de acurácia global e menor índice Kappa dentre os três algoritmos, sendo os valores de 75,5% e 0,674. No algoritmo de Floresta Aleatória foi observado uma menor confusão entre as principais classes, apenas na classe de queimada ocorreu uma classificação minimamente errada ao comparar com os outros classificadores (MAXVER e SVM).

O bioma Cerrado destaca-se como um grande desafio para classificação por conta da escala, das complexidades e padrões espectrais. Contudo, com o auxílio da aplicação desses algoritmos é possível otimizar e facilitar o mapeamento e monitoramento do uso do solo.

7. TRABALHOS FUTUROS

Com base na revisão bibliográfica de outros estudos e os resultados obtidos neste trabalho, sugere-se que em estudos posteriores deste tema sejam coletadas amostras no campo (in loco) a fim de melhorar a acurácia e minimizar incertezas na visualização das imagens de satélite. Recomendam-se, ainda, maiores coletas de amostras das áreas de maior índice de confusões na classificação e comparação com outras áreas de referência.

Sugerem-se também as coletas de amostras em bases de referências como MAPBIOMAS e TerraClass, pois são produtos realizados com maiores cargas de amostras e avaliação, servindo de base para algumas classes que apresentam confusões como áreas urbanas, sombras de relevo, agricultura e áreas satânicas.

Por fim, aconselha-se realizar a aquisição de uma imagem de satélite no período chuvoso para coleta de amostras e aplicação dos algoritmos a fim de viabilizar a comparação das acurácias, principalmente nas regiões onde a presença de recursos hídricos são fatores sensíveis na mudança de paisagem, como, por exemplo, nas áreas de campo no qual predominam gramíneas que no período seco podem ser confundidas com áreas de pecuária, agricultura ou solo exposto.

REFERÊNCIAS

ALENCAR, A. et al. Mapping Three Decades of Changes in the Brazilian Savanna Native Vegetation Using Landsat Data Processed in the Google Earth Engine Platform. Remote Sensing. 2020, 12, 924. Disponível em: https://doi.org/10.3390/rs12060924. Acesso em: 03 de abril de 2022.

ARAUJO FILHO, M. C. 2005. Desenvolvimento de um sistema de classificação hierárquico para mapas de uso e cobertura da terra por meio de imagens do satélite Landsat ETM+. Dissertação (Mestrado) - Universidade de Brasília, Instituto de Geociências, Brasília, 2005. 126 p

BREIMAN, L. Random forests. Machine learning, v. 45, n. 1, p. 5-32, 2001.

BROWN, M.; LEWIS, H.G.; GUNN, S.R. Linear spectral mixture models and support vector machines for remote sensing. IEEE Transactions on geoscience and remote sensing, v. 38, n. 5, p. 2346-2360, 2000

CHAMBERS, J.; Software for data analysis: Programing with R. New York, USA: Springer New York, 2008, p. 500

CHANDER, G.; MARKHAM, B. L.; HELDER, D. L. Summary of current radiometric calibration coefficients for Landsat MSS, TM, ETM+, and EO-1 ALI sensors. Remote sensing of environment, v. 113, n. 5, p. 893-903, 2009.

CONGALTON, R.G.; GREEN, K. Assessing the Accuracy of Remotely Sensed Data: Principles and Practices, Third Edition (3rd ed.). CRC Press, 2021. Disponível em: https://doi.org/10.1201/9780429052729. Acesso em 21 de abr 2022.

COHEN, J. A coefficient of agreement for nominal scales. Educational and psychological measurement, v. 20, n. 1, p. 37-46, 1960.

CRAWLEY, M, J. The R Book. Chinchester, Inglaterra: John Wiley & Sons, 2007. P. 950

DEILMAI, B. R.; AHMAD, B. B.; ZABIHI, H. Comparison of two Classification methods (MLC and SVM) to extract land use and land cover in Johor Malaysia. IOP Conference Series: Earth and Environmental Science, v. 20, p. 012052, 23 jun. 2014.

FACELI, K. et al. Inteligência Artificial: uma abordagem de aprendizado de máquina. 2. Ed. Rio de Janeiro: LTC, 2021.

HAN, J., KAMBER, M. & PEI, J. 2011, Data Mining: Concepts and techniques, 3rd ed., Morgan Kaufmann Publishers, Waltham, MA, USA.

IBGE. www.ibge.gov.br, 2021. Planilha disponível para download. Disponível em: https://www.ibge.gov.br/geociencias/organizacao-do-territorio/estrutura

territorial/15761-areas-dos-municipios.html?=&t=saiba-mais-edicao>. Acesso em: 15 maio 2022.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE) - Manuais: tutorial de geoprocessamento SPRIN, 2008.

LANDIS, J. R. The measurement of observer agreement for categorical data. Biometrics, v. 33, n. 3, p. 159-174, 1977.

MAPBIOMAS. Mapbiomas.org/ 2022. Análises de acurácia. Estimativas da acurácia do mapeamento da cobertura da terra pelo projeto MAPBIOMAS. Disponível em https://mapbiomas.org/analise-de-acurcia#:~:text=Acur%C3%A1cias%20do%20produtor%3A%20S%C3%A3o%20as,pixel%20da%20classe%20j%20corretamente. >. Acesso em: 15 maio 2022.

MELGANI, F.; BRUZZONE, L. Classification of Hyperspectral Remote Sensing Images with Support Vector Machines. IEEE Transactions on Geoscience and Remote Sensing, vol. 42, No. 8, August 2004.

MENESES, P.R.; ALMEIDA, T. Introdução ao Processamento de Imagens de Sensoriamento Remoto; UnB, CNPq: Brasília, Brasil, 2012; Disponível em: http://www.cnpq.br/documents/10157/56b578c4-0fd5-4b9f-b82a-e9693e4f69d8. Acesso em: 13 abril 2022

MIKHAIL, E.; ACKERMAN, F. Observations and Least Squares. University Press of America, 1976. 497 p.

MITTERMEIER, R.A.; TURNER, W.R.; LARSEN, F.W.; BROOKS, T.M.; GASCON, C. Global Biodiversity Conservation: The Critical Role of Hotspots. In Biodiversity Hotspots; Springer: Berlin/Heidelberg, Germany, 2011; pp. 3–22

MOHRI, Mehryar; ROSTAMIZADEH, Afshin; TALWALKAR, Ameet. Foundations of machine learning. MIT press, 2018.

PEDRINI, H.; SCHWARTZ, W. R. Análise de imagens digitais: princípios, algoritmos e aplicações. São Paulo: Thomson Learning, 2008. 508p.

PRATES, W. R. Ciência e negócios. Fonte: Ciência e negócios.com. 4/10/2018. https://cienciaenegocios.com/o-que-e-arvore-de-decisao-decision-tree-linguagem-r/

PINTO, A. S. A. S. et al. Avaliação de índices espectrais e do algoritmo Random Forest para a detecção de mudanças da cobertura do solo no Cerrado brasileiro. [s.l: s.n.]. Anais do XIX Simpósio Brasileiro de Sensoriamento Remoto, Santos – SP, 2019. Disponível em: <https://queimadas.dgi.inpe.br/~rqueimadas/material3os/2019_Pinto_etal_SoloCerrado_XIXSBSR_DE3os.pdf>. Acesso em: 8 de maio de 2022.

SANO, E.E.; ROSA, R.; SCARAMUZZA, C.A.M.; ADAMI, M.; BOLFE, E.L.; COUTINHO, A.C.; ESQUERDO, J.C.D.M.; MAURANO, L.E.P.; DA NARVAES, I.S.; OLIVEIRA FILHO, F.J.B.; et al. Land use dynamics in the Brazilian Cerrado in the period from 2002 to 2013. Pesquisa. Agropecuária Bras. 2019, 54.

CARVALHO, I. et al. Classificação da vegetação do parque nacional da chapada das mesas, maranhão, usando obia, machine learning e softwares livres. [s.l: s.n.]. Disponível em: <http://marte2.sid.inpe.br/col/sid.inpe.br/marte2/2019/10.23.11.17/doc/97824.pdf>. Acesso em: 8 de maio de 2022.

SARMIENTO, C. M.; RAMIREZ, G. M.; COLTRI, P. P.; SILVA, L. F. L. E.; NASSUR, O. A. C.; SOARES, J. F. Comparação de classificadores supervisionados na discriminação de áreas cafeeiras em Campos Gerais - Minas Gerais. Coffee Science, v. 9, n. 4, p. 546–557, 2014. Disponível em: < http://www.coffeescience.ufla.br/index.php/Coffeescience/article/view/760 >

SAWYER, D., "População, meio ambiente e desenvolvimento sustentável no cerrado"., In: Hogan, D.J.; Cunha, J.M.P.; Carmo, R.L. (org.). Migração e ambiente no Centro-Oeste, UNICAMP: PRONEX, Campinas, pp. 279-299, 2002

STRASSBURG, B. B. N. et al. Moment of truth for the Cerrado hotspot. Nature Ecology & Evolution, v. 1, n. 4, p. 1-3, 2017.

SUN, J.; YANG, J.; ZHANG, C.; YUN, W.; QU, J. Automatic remotely sensed image classification in a grid environment based on the maximum likelihood method. Mathematical and Computer Modelling, v. 58 n. 3, p. 573-581, 2013. Doi: 10.1016/j.mcm.2011.10.063

VAPNIK, V. The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995.

APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS EM DIAGNÓSTICO DE COVID-19 EM RAIOS-X DE TORAX

Edna Pereira Pinto Fernandes

Ricardo José Menezes Maia

RESUMO

A pandemia COVID-19 trouxe grandes e novos desafios para todos os segmentos da sociedade, principalmente para área médica e Tecnologia da Informação. O diagnóstico rápido e a identificação do estágio da doença são fatores essenciais no enfrentamento do COVID-19, no tratamento e recuperação de pacientes, no controle da propagação do vírus e no melhor gerenciamento dos sistemas de saúde. O raios-X de tórax é uma alternativa de exame a ser utilizada no diagnóstico rápido por permitir identificação do COVID-19 mesmo na fase inicial, estar disponível na maioria dos municípios e hospitais no Brasil e no mundo, ser de laudo rápido, de custo reduzido em comparação com Reverse-Transcriptase Polymerase Chain Reaction (RT-PCR) e Tomografica Computadorizada (TC). O uso da Inteligência Artificial (IA) permite a automatização da análise de raios-X, auxiliando as equipes médicas na triagem de pacientes, no diagnóstico e prognóstico de COVID-19, ampliando a capacidade de atendimento e de resposta da rede de saúde. Sendo assim, neste estudo foram realizadas pesquisas sobre IA e sua aplicação no diagnóstico de COVID-19, testados frameworks de aprendizagem profunda na identificação automática da doença em radiografias de base pública de imagens. Os frameworks escolhidos foram de rede neural convolucional Xception, ResNet50, ResNet152V2, InceptionResNetV2, MobileNetV2, DenseNet201, NASNetLarge, em conjunto com Transfer Learning, na classificação multiclasses. A solução Xception obteve o melhor desempenho com 88% de precisão de classificação da classe COVID-19 e a ResNet50, a accuracy-Score de 87,88%, demonstrando ser promissor o uso de técnicas de IA, mais especificamente Deep Learning, no diagnóstico de COVID-19 e de outras enfermidades pulmonares.

Palavras-chave: Covid. Diagnóstico. Raios-X. Rede Neural Convolucional. Transfer Learning.

1 INTRODUÇÃO

A doença Coronavírus 2019 (COVID-19) surgiu em 2019, causada pelo vírus Severe acute respiratory syndrome Coronavirus 2 (SARS-CoV-2), (OPAS, 2020), e espalhou-se no mundo inteiro, afetando países ricos e pobres, infectando milhões de pessoas (ASLAN et al., 2021).

Em 30 de janeiro de 2020, a Organização Mundial da Saúde (OMS), conforme Organização Pan-Americana da Saúde (OPAS), declarou que o surto de COVID-19 constituía uma Emergência de Saúde Pública de Importância Internacional (ESPII) e, em 11 de março, uma pandemia, ou seja, "distribuição geográfica de uma doença e não a sua gravidade" (WORLD HEALTH ORGANIZATION, 2021).

Mortes ou sequelas em função da COVID-19 têm produzido tragédias de todo tipo. Essa enfermidade, pelo seu nível de contaminação, tem afetado, em muitas situações, mais de uma pessoa por família, empresa e comunidade. O paciente precisa ser isolado do convívio de todos, sendo que, em algumas situações, o tratamento é agressivo e invasivo (ESTRELA et al., 2021).

Ademais, a pandemia do COVID-19 vem produzindo "repercussões e impactos sociais, econômicos, políticos, culturais e históricos sem precedentes na história recente das epidemias" (FIOCRUZ, 2020).

A triagem rápida de pacientes, o diagnóstico preciso (SHAMOUT et al, 2021; OZTURK et al., 2020), o início de tratamento de imediato, com o menor esforço de equipes médicas, o uso otimizado de recursos hospitalares e a redução do desgaste dos pacientes, das chances de contaminação, óbitos ou sequelas, do sofrimento da sociedade sempre foram importantes para uma medicina preventiva e efetiva (PRADO et al.). Entretanto, no caso da COVID-19, o atendimento desses requisitos é um desafio e chave de sucesso (PRADO et al.).

Dessa forma, o entendimento das características dos grupos acometidos de COVID-19, tendo em vista os tipos e as oportunidades de recuperação, complicações e até óbitos, são essenciais na definição de estratégias de atendimento e de políticas públicas (PRADO et al.).

Nesse cenário caótico e de nova realidade mundial, tornou-se urgente a oferta de ferramentas e instrumentos (PRADO et al.) que permitam o diagnóstico, prognóstico, tratamento e a cura dessa nova doença.

A Inteligência Artificial (IA) tem se consolidado como uma forte aliada na busca e apresentação de soluções na luta contra a COVID-19 (ALBAHRI et al., 2020), assim como tem sido adotada em outras áreas da medicina na detecção de arritmia, câncer, pneumonia, na segmentação de imagem de fundo de olho, na classificação de doenças como as cerebrais (OZTURK et al., 2020).

Destaca-se que "uma das principais complicações causadas pelo COVID-19 é a pneumonia" (PEREIRA et al., 2020) e que imagens radiológicas, inclusive raios-X, podem ser utilizadas como uma das primeiras opções de exame no diagnóstico de pneumonia (PEREIRA et al., 2020).

O uso de IA em raios-X de pacientes com suspeitas de COVID-19 é uma opção a ser explorada e implementada pelos ganhos reais alcançados para pacientes e equipes médicas (PEREIRA et al., 2020) e é o tema desta monografia com a aplicação de redes neurais convolucionais em diagnóstico da doença.

No capítulo 0 há esclarecimentos sobre este trabalho contendo Problema, Motivações e Justificativa, Objetivo Geral, Objetivos Específicos, Forma de Realização e Estruturação do Trabalho.

2 ESCLARECIMENTO DO TRABALHO

2.1 Problema

Quais opções de Inteligência Artificial (IA) podem ser utilizadas na identificação automática e diagnósticos de COVID-19 em raios-X de tórax?

Deseja-se encontrar um algoritmo adequado para o diagnóstico de COVID-19 através de radiografias de tórax.

Motivações e Justificativa

A reflexão inicial para a escolha do tema desta monografia foi a pergunta da própria autora sobre o problema que mais afligia as pessoas em 2020 e onde a Inteligência Artificial poderia ser útil.

A pandemia do COVID-19 foi a resposta pelas informações e a realidade abordadas na Introdução deste trabalho.

Assim sendo, a motivação deste trabalho pela pandemia COVID-19 foi a necessidade premente de buscar alternativas Inteligência Artificial passíveis de serem empregadas no diagnóstico e prognóstico de doenças, soluções que aumentem a qualidade e a agilidade da triagem de pacientes, o suporte às equipes médicas, a assertividade no tratamento das pessoas, a otimização de tempo e uso de recursos a redução de impactos e danos de ordem pessoal, familiar e econômica.

O foco de análise de raios-X de tórax foi decorrente das infecções causadas pelo COVID-19 afetarem, em especial, o sistema respiratório (ESTRELA et al., 2021).

Em termos acadêmicos, este trabalho poderá servir como registro e resgate de experiências de IA no combate ao COVID-19 e, também, de subsídio para trabalhos futuros.

2.2 Objetivo Geral

O objetivo geral deste trabalho é levantar diferentes usos de IA na classificação automática de imagens de raios-X de tórax, com a correspondente escolha de algoritmos a serem experimentados e a proposição de solução a ser ofertada aos intervenientes que lidam com COVID-19.

2.3 Objetivos Específicos

Levantar o estado da arte de modelos de Machine Learning e Deep Learning no diagnóstico e prognóstico de COVID-19 em raios-X de tórax.

Escolher arquiteturas de redes neurais Convolutional Neural Network (CNN) para testes.

Escolher base de dados e balancear classes de raios-x de tórax destinadas ao diagnóstico de COVID-19.

Testar arquiteturas escolhidas.

2.4 Forma de Realização do trabalho

Para alcançar esses objetivos, procedeu-se a um levantamento de experiências nacionais e internacionais na aplicação de IA em diagnóstico e prognóstico de COVID-19, consolidação das estratégias levantadas, com a escolha da análise de raios-x de tórax para testes. Nessa análise foram adotadas arquiteturas CNN do conjunto de framework Keras com aplicação em conjunto com Transfer Learning como solução Deep Learning de Visão Computacional em base de dados de imagens de raios-X disponível no Kaggle.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Diagnóstico e Prognóstico

A palavra diagnóstico é resultante da composição de duas partículas de origem grega: “dia”, que significa através de, e gnosis, conhecer, e, dessa forma, então, significa “conhecer através” ou “através do conhecimento” (SANCHO; PFEIFFER; CORRÊA, 2019).

Diagnóstico na área médica está relacionado com “conhecimento ou determinação duma doença pelo(s) sintoma(s), sinal ou sinais e/ou mediante exames diversos (radiológicos, laboratoriais entre outros)” (FERREIRA, 2004).

“A anamnese e o exame físico conduzem ao diagnóstico, que leva ao plano terapêutico e ao prognóstico do paciente” (SANCHO; PFEIFFER; CORRÊA, 2019).

Anamnese ou anamnésia significa “informação acerca do princípio e evolução de uma doença até a primeira observação do médico” (FERREIRA, 2004).

Prognóstico na área médica está relacionado à probabilidade de desenvolvimento que um determinado estado de saúde ocorra, ou seja, um resultado como dor, morte ou complicações, resposta ao tratamento, mudança do quadro geral e/ou qualidade de vida de um dado indivíduo, com remissão ou progressão de

doença, em um período específico, considerando-se o perfil clínico do paciente, dados como idade, sexo, história, sintomas, sinais e exames realizados (MOONS et al., 2009).

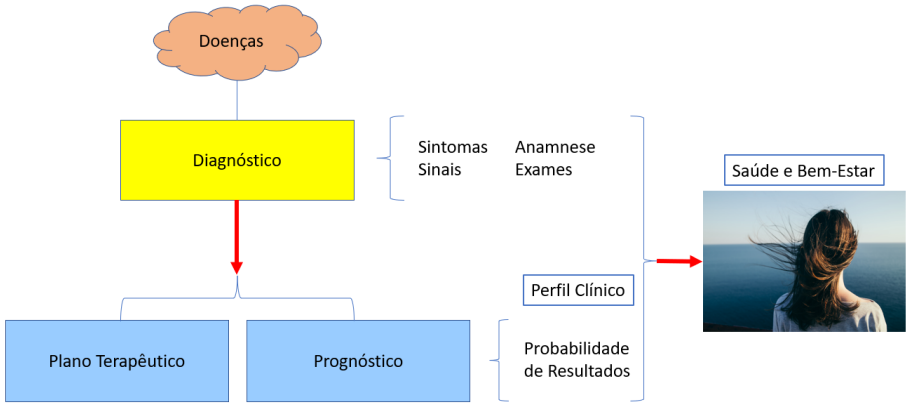
Na realização de prognósticos há necessidade de levantamento de variáveis ou preditores devidamente combinados com o fim de estimar a probabilidade, o risco futuro de um determinado resultado (MOONS et al., 2009).

Preditores, no caso de COVID-19, permitem a identificação de contaminados iniciais, diagnóstico e prognósticos de pacientes, a previsão de riscos de admissão em Unidade de Terapia Intensiva (UTI), ventilação mecânica e morte de pessoas em tempo inclusive de avaliação demográfica e rastreamento de contato e definição de isolamento social e, mesmo, lockdown (LAGUARTA; HUETO; SUBIRANA, 2020).

O prognóstico é instrumento importante na indicação de condutas na vida dos indivíduos, na prevenção e no tratamento de doenças, na seleção de pacientes para pesquisas terapêuticas, na comparação de desempenho de hospitais, na previsão de taxas de mortalidade (MOONS et al., 2009).

Dessa forma, o diagnóstico é a identificação de uma doença, conforme conhecimento do quadro geral de um dado paciente e do universo de doenças existentes, e o prognóstico, portanto, utiliza-se do diagnóstico realizado no entendimento dos possíveis desdobramentos e resultados, conforme representação na Figura 11.

Figura 11 — Diagnóstico, Plano Terapêutico e Prognóstico

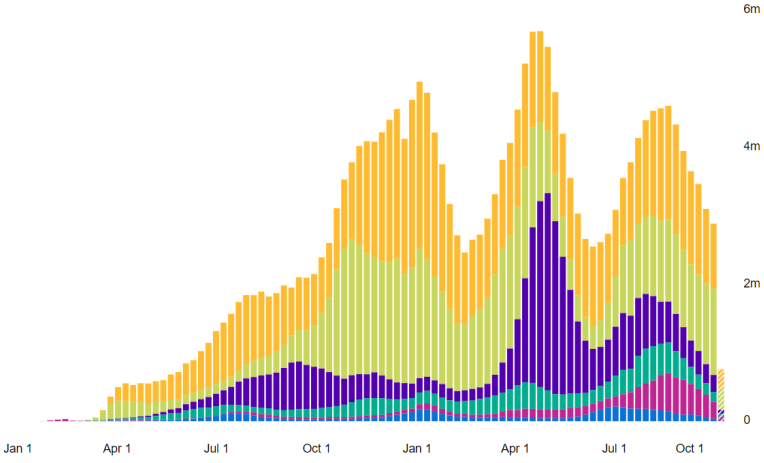


Fonte: Produzida pela autora.

3.2 COVID-19 — Panorama

No Painel da Organização Mundial da Saúde (OMS), representado no Gráfico 7 (WORLD HEALTH ORGANIZATION, 2021) fica demonstrada a progressão acelerada da COVID-19, pois na data inicial de análise, em 30 de dezembro de 2019, havia 1 paciente identificado e já em 11 de outubro de 2021, 238.229.187 casos foram confirmados.

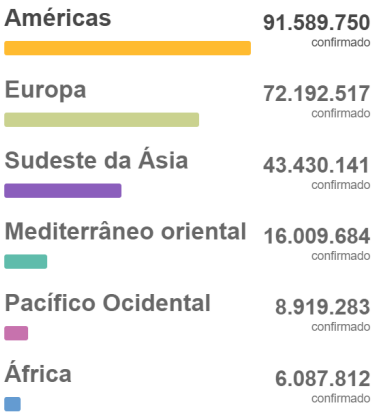
Gráfico 7 — Casos confirmados de COVID-19 por Região



Fonte: Painel do Coronavírus (WORLD HEALTH ORGANIZATION, 2021).

Apenas nas Américas foram confirmados 91.589.750 (WORLD HEALTH ORGANIZATION, 2021), Gráfico 8.

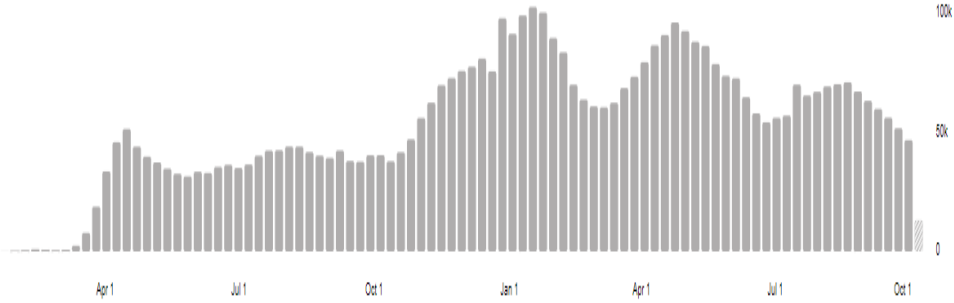
Gráfico 8 — Casos confirmados de COVID-19 nas Américas



Fonte: Painel do Coronavírus (WORLD HEALTH ORGANIZATION, 2021).

No Gráfico 9 há o Painel da evolução de mortes no mundo com 4.010.834 registradas de 30 de dezembro de 2019 a 11 de outubro de 2021 (WORLD HEALTH ORGANIZATION, 2021).

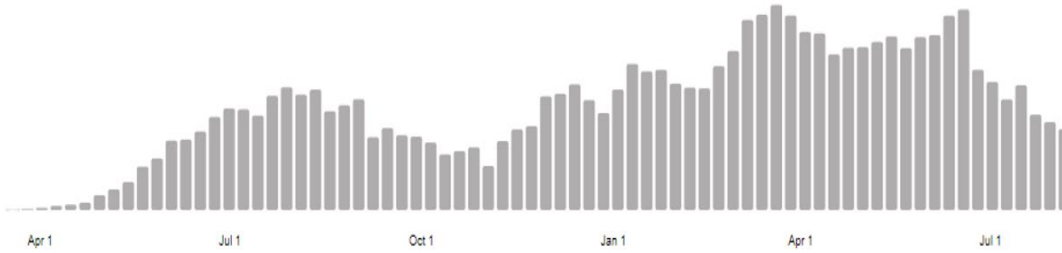
Gráfico 9 — Mortes confirmadas mundialmente



Fonte: Painel do Coronavírus (WORLD HEALTH ORGANIZATION, 2021).

No Brasil, de 3 de janeiro de 2020 a 11 de outubro de 2021 foram confirmados 21.582.738 casos, Gráfico 10 (WORLD HEALTH ORGANIZATION, 2021).

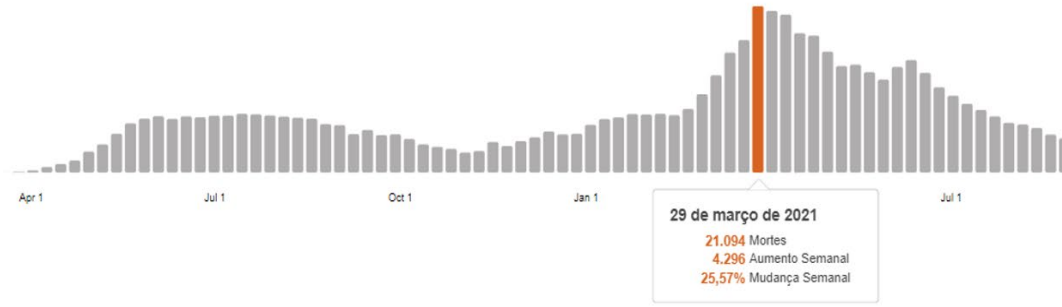
Gráfico 10 — Casos Confirmados do Brasil



Fonte: Painel do Coronavírus (WORLD HEALTH ORGANIZATION, 2021).

Ainda no Brasil foram registrados 601.213 óbitos de COVID-19, com pico em 29 de março de 2021 de 21.094 mortes, Gráfico 11.

Gráfico 11 — Mortes Confirmadas do Brasil



Fonte: Painel do Coronavírus (WORLD HEALTH ORGANIZATION, 2021).

Segundo Moura et al (2021) há novos picos da doença, após a contenção da primeira onda, sendo que no Brasil a segunda onda foi iniciada a partir de novembro de 2020.

O tempo de internação por COVID-19, pela gravidade da doença, traz mais dificuldades de atendimento de pacientes desta e de outras enfermidades. O tempo médio de hospitalização tem sido de 18,5 dias e o de UTI, 11,55 dias, tomando-se como referência os números obtidos na pesquisa de Moura et al (2021) para as duas ondas de COVID-19 no Brasil.

Cabe destacar que o efeito na saúde das pessoas vai além da infecção pela COVID-19. Com tudo que envolve a doença, ampla e repetidamente noticiado, a

população é afetada no mínimo, psicologicamente, pois há sentimentos como medo, raiva, desamparo, abandono, tristeza, desesperança, ansiedade, angústia, incerteza e a doença soa como uma sentença de morte (ORNEL et al., 2020; KOURY, 2020; ESTRELA et al., 2021). A COVID-19 também afeta a vida e o emocional das crianças, tendo em vista que foram submetidas a uma nova realidade, como pode ser observado em livro de uma menina de 9 anos sobre a doença (COUTINHO, 2021).

Outros problemas podem ser observados, pois, acompanhamentos regulares e tratamentos médicos importantes estão sendo perigosamente desconsiderados ou adiados, provocando complicações de saúde e doenças e, também, mortes por outros motivos como implicação secundária e colateral à pandemia. Fundamentando este fato, com a pandemia de COVID-19, "vinte e três milhões de crianças deixaram de receber vacinas infantis básicas por meio de serviços de saúde de rotina em 2020, o maior número desde 2009 e 3,7 milhões a mais do que em 2019", de acordo com os dados OMS e UNICEF, considerando-se os números oficiais de imunização mundial (UNICEF, 2021).

Em julho de 2021, no Brasil "nos dois meses mais mortais de 2021 (março e abril), foram registrados valores 87% e 88% superiores ao patamar pré-pandemia (188.725, em março, e 187,842 em abril)", sendo que a média mensal de mortes em 2018 e 2019 ficou em torno de 100 mil mortes/mês. (NICOLELIS, 2021) Segundo o Neurocientista, esse excedente de óbitos de todos os tipos, e não apenas de COVID-19, é decorrente do colapso hospitalar, mortes por falta de leitos, por atrasos de tratamentos, adiamento de cirurgias, e, provavelmente, pelo fato de mais da metade da população encontrar-se em estado de insegurança alimentar (NICOLELIS, 2021).

A dinâmica de funcionamento de hospitais e sistemas de saúde também foi totalmente alterada, com o aumento de demanda relevante pelo uso de leitos hospitalares, sendo que muitos, inclusive, de Terapia Intensiva (NEDEL, 2020; PRADO et al.; WYNANTS et al., 2020).

As equipes médicas têm trabalhado no limite da capacidade humana e sofrido baixas expressivas. Estão submetidos a pressão e riscos dos mais variados tipos (ESTRELA et al., 2021).

3.3 COVID-19 — Diagnóstico

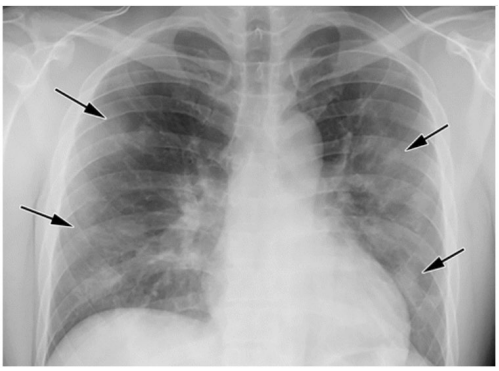
Segundo Organização Pan-Americana de Saúde, COVID-19 é uma doença infecciosa tendo como principais sintomas febre, cansaço e tosse seca, podendo ainda serem apresentadas "dores, congestão nasal, dor de cabeça, conjuntivite, dor de garganta, diarreia, perda do paladar ou olfato, erupção cutânea na pele ou descoloração dos dedos ou dos pés" (OPAS, 2020).

O SARS-CoV-2 pode causar pneumonia grave (PEREIRA, Rodolfo M. et al., 2020), lembrando que "a pneumonia é uma infecção da parte do pulmão responsável pela transferência de gases (alvéolos, ductos alveolares e bronquíolos respiratórios), denominada parênquima pulmonar" (PEREIRA et al., 2020).

Os achados típicos em imagens de casos de COVID-19 são Ground-Glass Opacities (GGO), ou seja, opacidade em vidro fosco, com predileção pela periferia do pulmão, "pavimentação em mosaico" e mais raramente "sinal de halo reverso ou invertido" (REVZIN et al., 2020).

Nas imagens a seguir são verificáveis estágios do COVID-19. Na Figura 12 percebe-se a opacidade em vidro fosco bilateral e periférica como resultado da progressão da doença ao longo de 4 dias (REVZIN et al., 2020).

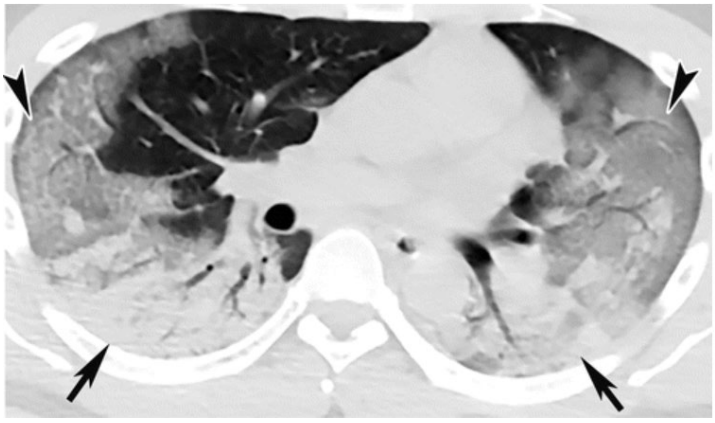
Figura 12 — Raios-X de progressão 4 dias — Opacidade em Vidro Fosco periférica



Fonte: Figura 2a (REVZIN et al., 2020).

Na Tomografia Computadorizada (TC) da Figura 13 após 4 dias percebe-se a opacidade em vidro fosco bilateral, periférica no pulmão direito e difusas no pulmão esquerdo (REVZIN et al., 2020).

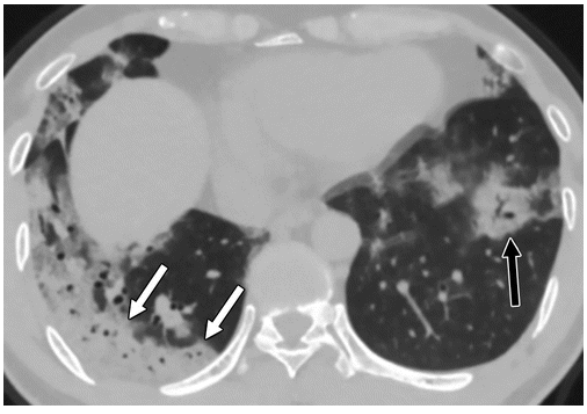
Figura 13 — Tomografia de progressão depois de 4 dias — Opacidade em Vidro Fosco periférica



Fonte: Figura 2b (REVZIN et al., 2020).

A Figura 14 trata-se de Tomografia Computadorizada após 10 dias do início dos sintomas em que são percebidas opacidades consolidativas bilaterais, com distribuição maior na parte inferior do lobo direito (BERNHEIM et al., 2020).

Figura 14 — Tomografia — Opacidade consolidativa bilaterais



Fonte: Figura 3 (BERNHEIM et al., 2020).

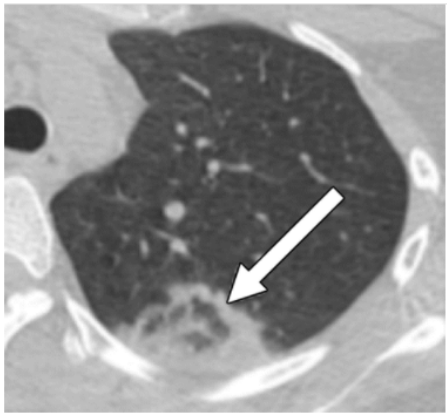
Na Tomografia Computadorizada da Figura 15 é apresentada opacidade em um padrão de “pavimentação de mosaico” (BERNHEIM et al., 2020).

Figura 15 — Tomografia — Pavimentação em Mosaico

Fonte: Figura 5a (BERNHEIM et al., 2020).

Por fim, a Tomografia Computadorizada da Figura 16 exemplifica a opacidade em vidro fosco com um anel de consolidação mais denso denominado “halo reverso” (BERNHEIM et al., 2020).

Figura 16 — Tomografia — Halo reverso



Fonte: Figura 5b (BERNHEIM et al., 2020).

Há várias sequelas fisiopatológicas e psicológicas em função do acometimento do COVID-19. Lesões cardíacas diretas ou indiretas, inflamação sistêmica, sopro, arritmias, infarto e morte súbita podem ocorrer (ESTRELA et al.,

2021). Ademais, há casos de efeitos neurológicos “associados a outras condições, como acidente vascular cerebral agudo, lesão muscular esquelética e encefalopatia”, confusão mental, dor de cabeça, vômito, náusea (ESTRELA et al., 2021).

De acordo com OPAS, “cerca de 80% das pessoas se recuperam da doença sem precisar de tratamento hospitalar”, sendo que uma em cada seis pessoas fica gravemente doente e desenvolve dificuldade de respiração (OPAS, 2020). Idosos e pessoas com comorbidades como pressão alta, cardiopatias, problemas pulmonares, diabetes e câncer possuem mais riscos de desenvolver quadros mais graves de COVID-19 (OPAS, 2020).

Apesar do teste Reverse Transcription Polymerase Chain Reaction (RT-PCR) ser uma das técnicas valiosas na identificação de pessoas infectadas pela COVID-19, infelizmente não está disponível em quantidade suficiente para a população (ASLAN et al., 2021). Testes virais e sorológicos são caros, tornando inviável testar-se um país inteiro, pois, exemplificando-se, para teste de toda a população dos Estados Unidos seriam necessários US$ 8,6 bilhões, assumindo-se o valor por teste de US$23,00 (LAGUARTA; HUETO; SUBIRANA, 2020). Ademais, RT-PCR apresenta sensibilidade de 60% — 70% (OZTURK et al., 2020).

Outros exames tornam-se alternativas na detecção de COVID-19 como Tomografia Computadorizada (TC), Ressonância Magnética (RM), Tomografia por Emissão de Pósitron, Ultrassom e Radiografia (raios-X) de Tórax (ASLAN et al., 2021).

O critério de diagnóstico adotado no Brasil, na sua grande maioria, foi o laboratorial, com aumento do critério clínico-imagem a partir da segunda onda da doença (MOURA et al., 2021).

Cabe mencionar que sintomas da doença podem ser verificados em imagens radiológicas, inclusive em pacientes com resultados de RT-PCR negativos (OZTURK et al., 2020; ZU et al., 2020).

Raios-X é uma opção importante na identificação ou confirmação de quadro de COVID-19, tendo em vista que os equipamentos convencionais de realização

estão disponíveis na maioria dos hospitais e clínicas, diferentemente de Tomografia Computadorizada (HEMDAN; SHOUMAN; KARAR, 2020).

Em um trabalho de revisão sistemática de publicações na área da Saúde, Wynants et al. (2020) verificaram os processos de construção, avaliação e divulgação científica de 66 modelos preditivos para COVID-19. Uma parcela considerável dos modelos foi elaborada para a predição da doença via imagens. Além disso, os autores identificaram como preditores mais comumente utilizados para o diagnóstico de COVID-19 fatores como: idade, gênero, temperatura corpórea, sinais vitais, sintomas, pressão sanguínea e creatinina. Também se destacaram entre os modelos de prognóstico os de desfechos para o óbito, deterioração clínica e tempo de internação.

A realização dos exames é importante, mas a rápida e precisa avaliação dos resultados, a identificação do estágio da doença, a detecção de doentes, mesmo que assintomáticos, o entendimento da evolução possível do quadro dos pacientes são etapas decisivas para o melhor acolhimento de pessoas (OZTURK et al., 2020), a execução de tratamentos e as tomadas de decisão em toda a cadeia de planejamento estratégico, tático e operacional, envolvendo governos, autoridades sanitárias, iniciativa privada e comunidades de especialistas de várias áreas de conhecimento técnico e científico (CASTRO et al., 2020).

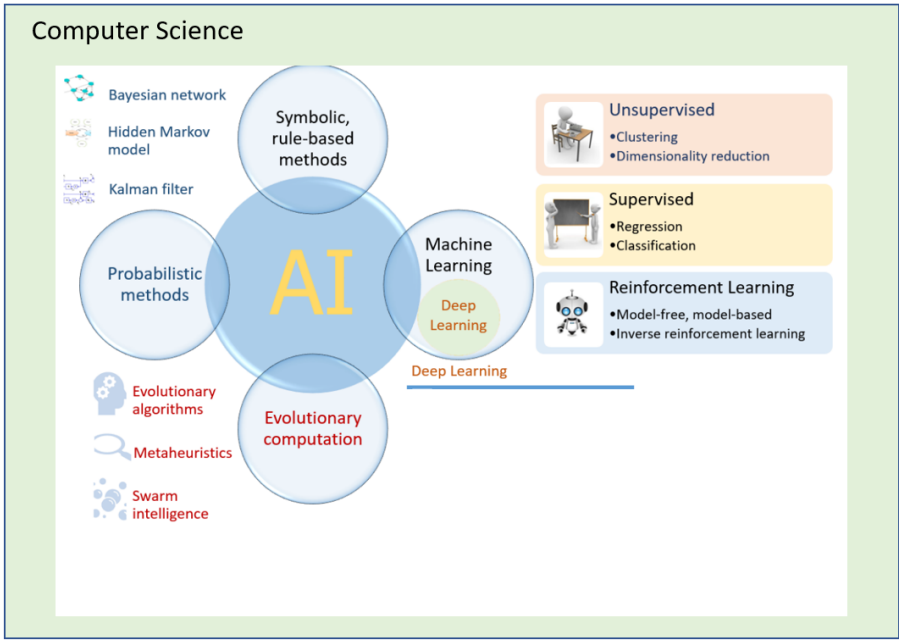
3.4 Inteligência Artificial (IA)

Inteligência artificial (IA) representa a ciência e a engenharia de criação de computadores e programas com inteligência (MCCARTHY, 2007), portanto, à semelhança do que acontece com o cérebro humano, que permite o aprendizado a partir de dados e, consequentemente, a tomada de decisão e ações (SHARMA, 2018).

O Propósito da Inteligência Artificial é automatizar tarefas repetitivas e redundantes a partir do aprendizado de dados ou registros, do entendimento de padrões e estruturas dos dados ou registros anteriores e da previsão do resultado futuro (SHARMA, 2018).

A Inteligência Artificial é parte da Ciência da Computação e possui vários subcampos de pesquisa (SARKAR; BALI; GHOSH, 2018), como demonstra a Figura 17, sendo que neste trabalho serão abordados apenas Machine Learning e Deep Learning.

Figura 17 — Artificial Intelligence — Abrangência



Fonte: Figura 1 — Artificial Intelligence abrangência (NGUYEN et al., 2020) ajustada pela autora.

3.5 Machine learning

Machine Learning é ramo da Inteligência Artificial que visa análise e previsão de padrões em amostras de dados para aplicação em novos dados, via desenvolvimento de métodos de aprendizagem automática (SILVA JÚNIOR; BEZERRA; ANDRADE, 2020)

Os algoritmos de aprendizado de máquina podem ser divididos basicamente em supervisionado, não supervisionado, semi-supervisionado e por reforço, sendo que na Figura 17 são representados apenas 3.

Aprendizado supervisionado é realizado a partir de dados de treinamento rotulados/classificados/categorizados por seres humanos (SARKAR; BALI;

GHOSH, 2018), ou seja, as classes de dados de treinamento são previamente definidas. Classificação e Regressão são dois representantes desta categoria de aprendizado (SARKAR; BALI; GHOSH, 2018).

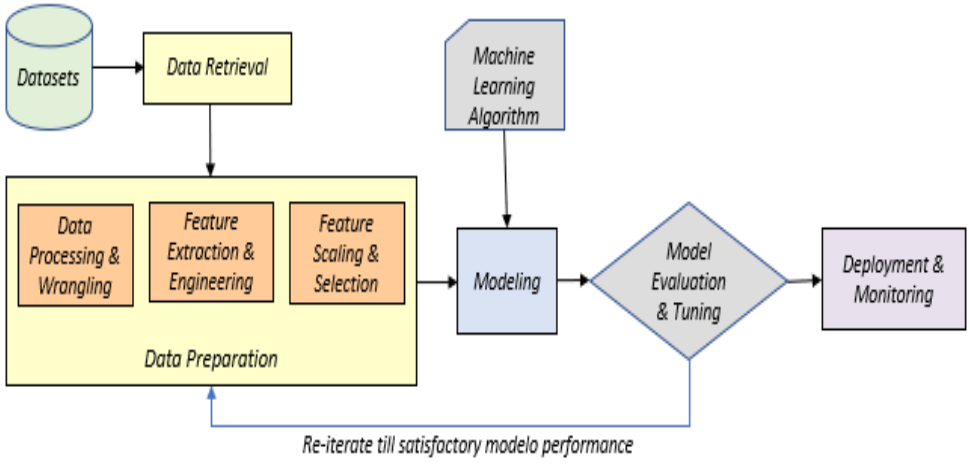
No Aprendizado não Supervisionado as entradas brutas são analisadas por algoritmo, a estrutura, relacionamentos e padrões, identificados e as classes, formadas sem intervenção humana (SARKAR; BALI; GHOSH, 2018). Exemplos desta categoria de aprendizado são Clustering, Dimensionality Reduction e Association Rule Mining (SARKAR; BALI; GHOSH, 2018).

O Aprendizado Semi-supervisionado é uma categoria híbrida dos aprendizados supervisionado e não supervisionado, valendo-se das vantagens desses dois tipos, com uma quantidade pequena de dados rotulados e uma maior não rotulada (SARKAR; BALI; GHOSH, 2018).

No Aprendizado por Reforço há a figura de um agente que interage com o ambiente por um período, testa várias possibilidades e, via maximização de prêmio e recompensa, o algoritmo aprende, sanando ou mitigando o erro (SARKAR; BALI; GHOSH, 2018).

O workflow de Machine Learning e seus diferentes subcomponentes estão representados na Figura 18 (SARKAR; BALI; GHOSH, 2018).

Figura 18 – Machine Leaning — Etapas



Fonte: Pipeline padrão de Machine Learning e etapas (SARKAR; BALI; GHOSH, 2018).

Os Datasets apresentam-se em formatos diversos, estruturados ou não, com dados de qualidade variada, necessitando de formas de Recuperação, Coleta, Limpeza e Ajustes específicos (SARKAR; BALI; GHOSH, 2018).

A Preparação dos Dados a serem utilizados nos algoritmos é a etapa que requer mais tempo para o processo de Machine Learning sendo dividida em outras subetapas como demonstrado na Figura 18 (SARKAR; BALI; GHOSH, 2018).

A análise exploratória dos dados coletados/recuperados é fundamental para o entendimento de atributos, facetas, dimensões e outras questões dos dados que precisam ser identificadas logo no início do processo de Machine Learning (SARKAR; BALI; GHOSH, 2018).

Compreendido o Dataset, a próxima subetapa da preparação de dados é o Processamento dos dados com o fim de tornar os dados passíveis de utilização pelos algoritmos de Machine Learning, realizando-se procedimentos de limpeza, transformação, mapeamento (SARKAR; BALI; GHOSH, 2018).

A Engenharia e Extração de Features (características, recursos) é subetapa em que atributos existentes são utilizados na derivação e extração de atributos ou recursos específicos de contexto/casos de uso, conforme emprego de técnicas próprias a cada tipo de dado (SARKAR; BALI; GHOSH, 2018).

Na Seleção e Dimensionamento de Features há a identificação do subconjunto de características que efetivamente é essencial à etapa de modelagem, evitando-se a "maldição da dimensionalidade" decorrente da dificuldade de interpretação, visualização e outros procedimentos em Datasets com números grandes de atributos (SARKAR; BALI; GHOSH, 2018).

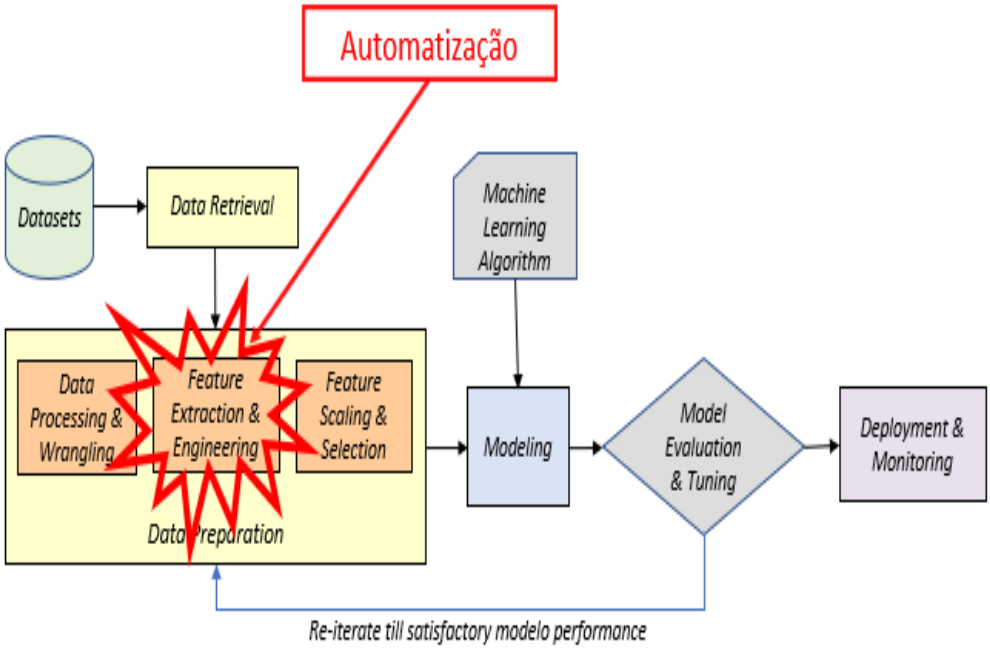
Na etapa de Modelagem há o treinamento iterativo das features de dados por método ou algoritmo de Machine Learning, a partir da realização de avaliações de métricas e refinamentos sucessivos até que seja obtida a melhor performance, com o propósito de aplicação/generalização futura do aprendizado em outros dados (SARKAR; BALI; GHOSH, 2018).

3.6 Deep Learning

A engenharia e extração de características pode ser complexa, demorada e trabalhosa dependendo do tipo de dado analisado como, por exemplo, imagens, vídeos, sons (SARKAR; BALI; GHOSH, 2018). A engenharia de características pode não ser efetiva, pois um número pequeno ou irrelevante de características pode provocar underfitting ou subajuste (modelo não consegue aprender com os dados de treinamento) e um número alto de características, overfitting ou sobreajuste (modelo não consegue generalizar o aprendizado dos dados de treinamento em dados de testes) (SARKAR; BALI; GHOSH, 2018).

Sendo assim, surgiu Deep Learning ou Aprendizagem Profunda, como uma especialização de Machine Learning, com o foco na automatização da engenharia e extração de características conforme Figura 19 (SARKAR; BALI; GHOSH, 2018).

Figura 19 — Deep Learning — Automatização de Engenharia e Extração de Features



Fonte: Pipeline padrão de Machine Learning e etapas (SARKAR; BALI; GHOSH, 2018) ajustado pela autora para representar Deep Learning.

Com dados suficientes, é possível com um modelo de Deep Learning descobrir uma hierarquia adequada de características de crescente complexidade (SARKAR; BALI; GHOSH, 2018).

Em função dessa automatização, há outras diferenças em relação à Machine Learning decorrentes. No Quadro 1 há um comparativo resumido algumas diferenças entre Machine Learning e Deep Learning, considerando-se os dados de Sharma (2018).

Quadro 1 — Comparativo entre Machine Learning e Deep Learning

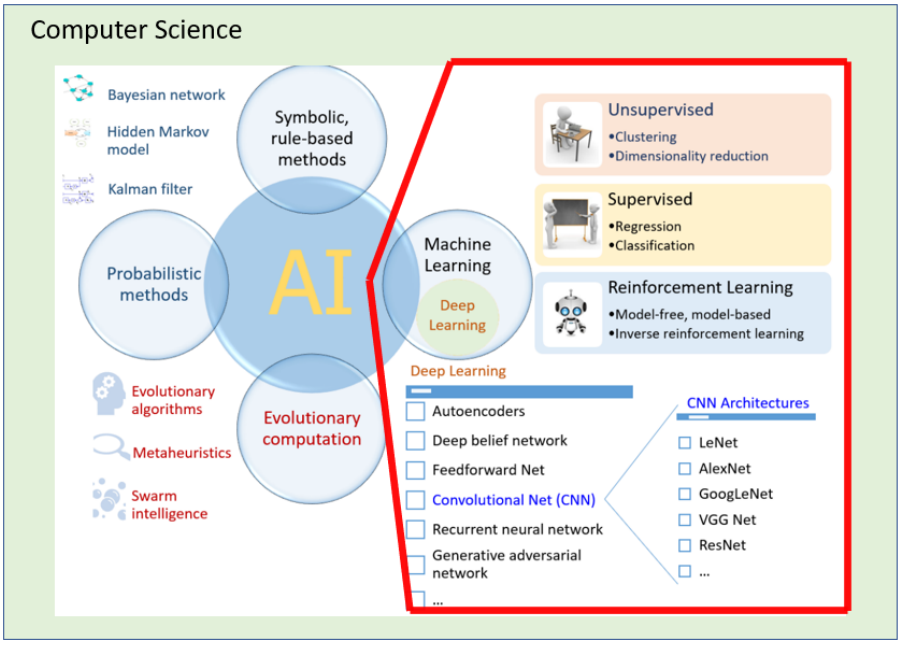
Comparação	
Machine Learning	Deep Learning
Dados como uma entrada, análise dos dados, tomada de decisões via aprendizado adquirido com treinamento.	Dados como entrada, tomada de decisões via rede neural artificial.
Depende de recursos criados manualmente, tais como padrões binários locais, histograma de gradientes.	Realiza a extração de recursos significativos dos dados brutos. Executa extração hierárquica de recursos. Aprende recursos em camadas, aprendendo recursos de mais baixo nível nas camadas iniciais e uma representação mais abstrata nas camadas subsequentes.
Algoritmos de aprendizado geralmente funcionam bem, mesmo que o conjunto de dados seja pequeno.	Necessita de volume significativo de dados para melhorar o desempenho. Com mais dados, a profundidade da rede (número de camadas) também aumenta, portanto, exige mais computação.
Exige uma CPU com especificações razoáveis.	Exige uma Graphics Processing Units (GPU) que tenha milhares de núcleos de comparação.
O tempo de treinamento pode variar de minutos a algumas horas, podendo demorar um pouco durante os testes.	Tempo de treinamento pode variar entre algumas horas a meses, impactado também pelo aumento de número de camadas da rede e o número de parâmetros conhecidos como pesos.

Comparação	
Machine Learning	Deep Learning
	Tempo de inferência pode ser aumentado conforme a profundidade das redes neurais, considerando que os dados do teste passam por todas as camadas da rede, com várias multiplicações.
A saída em geral é um valor numérico com uma pontuação ou uma classificação.	Saída pode ser uma pontuação, um elemento, texto, fala entre outras.

Fonte: Produzido pela autora do trabalho com os dados coletados em pesquisa de campo Sharma (2018).

Na Figura 20 verifica-se a relação de tipos de Deep Learning, com destaque à Convolutional Net (CNN) que será detalhada e utilizada neste trabalho.

Figura 20 — Deep Learning



Fonte: (NGUYEN et al., 2020) ajustada pela autora.

Atualmente Deep Learning é implementada como uma rede neural de múltiplas camadas ocultas com o fim de representar hierarquicamente os dados de entrada (SARKAR; BALI; GHOSH, 2018).

3.7 Redes Neurais Artificiais

A rede neural, técnica de IA, mais especificamente de Inteligência Computacional (IC), é fundamentada em modelos matemáticos tendo estruturas neurais biológicas como referência (PINHEIRO, 2020).

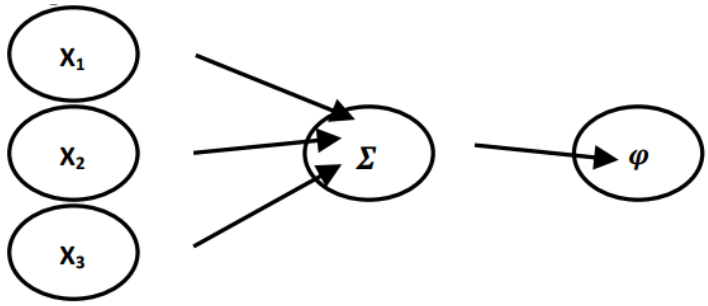
É algoritmo indutivo ou abdutivo, diferente do tradicional que é do tipo dedutivo, dependente de um programador na estruturação de possíveis soluções a partir de regras pré-definidas (PINHEIRO, 2020).

O algoritmo indutivo ou abdutivo aplica o aprendizado constante por generalização, similar ao adotado pelos seres humanos, utilizando dados no aprendizado, de forma supervisionada ou não (PINHEIRO, 2020).

Na aplicação de algoritmo indutivo não cabe a expectativa de acerto de 100%, mesmo com o aumento de dados (PINHEIRO, 2020).

A definição de Redes Neurais Artificiais (RNA) passou por evoluções ao longo do tempo. Inicialmente o neurônio artificial foi definido pelo Warren McCulloch e Walter Pitts, em 1943, e chamado McCulloch-Pitts, conforme estrutura da Figura 21 (PINHEIRO, 2020).

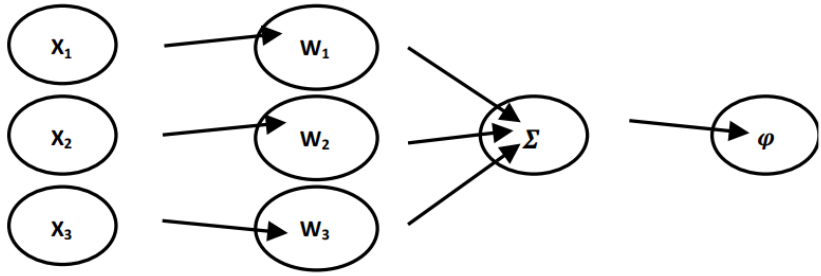
Figura 21 — Rede Neural Regular Vs CNN



Fonte: Figura 4.1 — Neurônio McCulloch-Pitts (PINHEIRO, 2020)

Frank Rosenblatt, em 1950, criou uma rede com vários neurônios, denominada Perceptron, Figura 22, composta de camadas, ou seja, a de entrada, a intermediária e a de saída (PINHEIRO, 2020).

Figura 22 — Perceptron proposto por Frank Rosenblatt

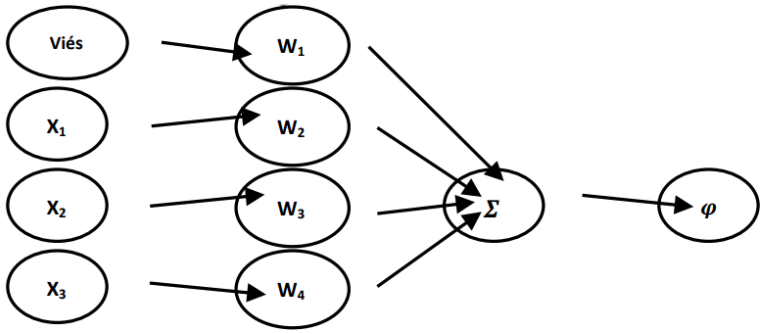


Fonte: Figura 4.2 — Perceptron (PINHEIRO, 2020).

A proposta da aplicação de Perceptron é de resolução de problemas de reconhecimento de padrões, classificando-os e rotulando-os como parte de uma dada classe ou categoria (PINHEIRO, 2020), sendo que x representa a entrada, w, o peso (weight), ou seja, a importância que as entradas exprimem para a saída (NIELSEN, 2015).

Em 1960, Bernard Widrow e Marcian Hoff criaram o modelo ADAptive LInear NEuron (ADALINE), Figura 23, em que foram implementadas a multiplicação de pesos às entradas e a soma de um viés (PINHEIRO, 2020).

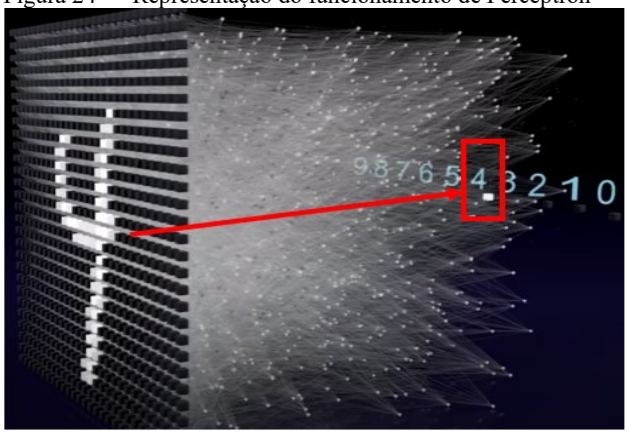
Figura 23 — ADALINE proposto por Bernard Widrow e Marcian Hoff



Fonte: Figura 4.3 — ADALINE (PINHEIRO, 2020).

Na Figura 24 é possível verificar o funcionamento de um perceptron, considerando-se o reconhecimento de dígitos mesmo que grafados de formas diferentes.

Figura 24 — Representação do funcionamento de Perceptron



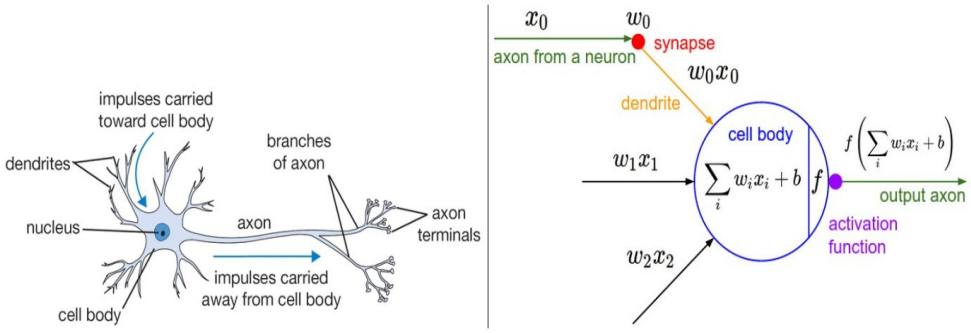
Fonte: Parte do vídeo https://youtu.be/3JQ3hYko51Y adaptada pela autora.

Após essas conceituações iniciais, na Figura 25 há um paralelo do neurônio biológico com o correspondente modelo matemático utilizado em rede neural (KARPATHY, 2019).

O neurônio biológico recebe sinais de entrada dos dendritos e gera sinal de saída ao longo do axônio, que se comunica por sinapses a dendritos de outros neurônios (KARPATHY, 2019).

Por conseguinte, no neurônio artificial (unidade ou unidade de processamento), modelo computacional representativo de um neurônio biológico, as funções calculadas determinarão a saída a ser disparada que pode servir de entrada para um próximo neurônio (KARPATHY, 2019).

Figura 25 — Neurônio biológico Vs Modelo matemático — Rede Neural

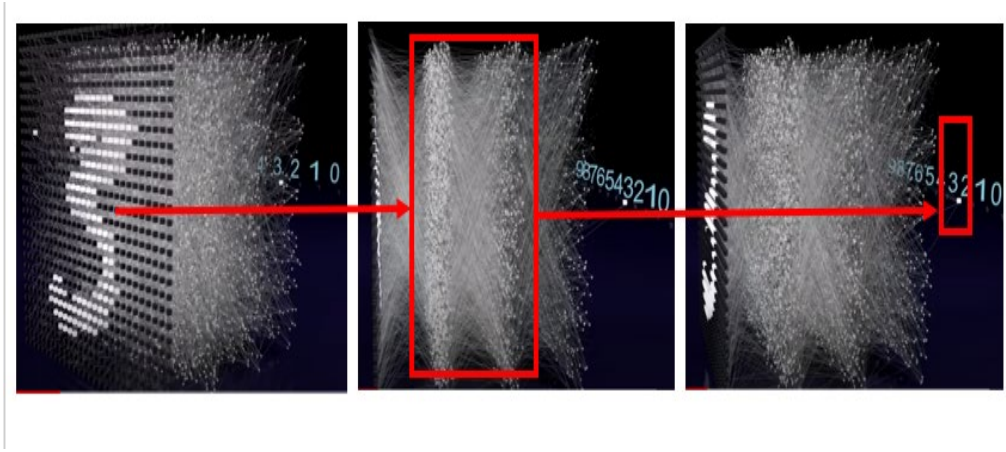


Fonte: Desenho sobre neurônios biológico e artificial — Módulo 1 (KARPATHY, 2019).

Em 1980, foram desenvolvidos o modelo Self-Organizing Maps (SOM), por Kohonen, e o Backpropagation, por Paul Werbos (PINHEIRO, 2020).

Como evolução das redes neurais perceptron e ADALINE, foi implementada a MultiLayer Perceptron (MLP), com o incremento de mais “camadas de neurônios ligadas entre si por sinapses com peso” e a possibilidade de retropropragação do erro, backpropragation, ou seja, em um processo iterativo de ajustes de pesos sinápticos (PINHEIRO, 2020), como representado na Figura 26.

Figura 26 — MultiLayer Perceptron (MLP)



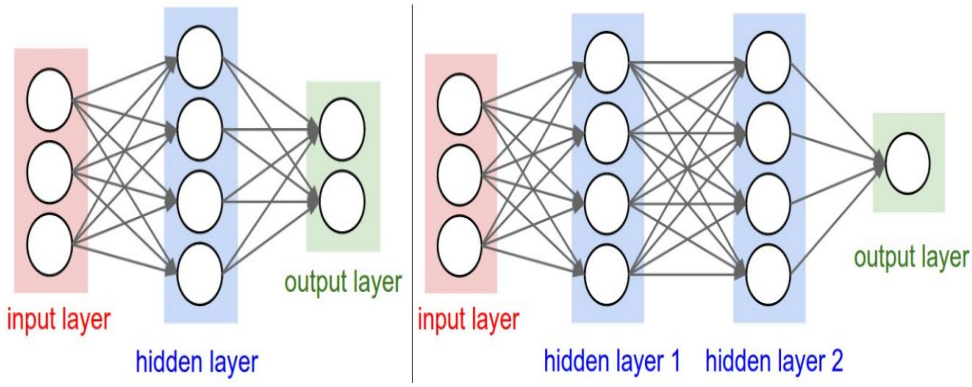
Fonte: Partes do vídeo https://youtu.be/3JQ3hYko51Y adaptadas pela autora.

3.8 Arquiteturas de Rede Neural

As redes neurais são representadas como conjuntos de neurônios, organizados em camadas distintas, sendo que cada neurônio é ligado/conectado aos neurônios da camada consecutiva (totalmente conectada) e, jamais com os da mesma camada, mantendo-se, assim a independência entre neurônios da mesma camada (KARPATHY, 2019).

No primeiro gráfico da Figura 27 há a representação de uma rede neural com 2 camadas, 1 oculta (hidden player), com 4 neurônios, e 1 de saída, com 2 neurônios. No segundo gráfico, há 3 camadas, sendo 2 ocultas, com 4 neurônios cada, e 1 de saída, com um neurônio. Em geral, na última camada há as pontuações de classificação. As camadas ocultas são as que estão entre as de entrada e saída. Cabe mencionar que os círculos representam neurônios e as linhas entre eles as conexões.

Figura 27 — Redes Neurais de forma gráfica



Fonte: Figura representativa de camadas – Módulo 1 (KARPATHY, 2019).

O dimensionamento das redes neurais é obtido a partir dos números de neurônios ou o número de parâmetros (KARPATHY, 2019). Para as redes citadas na Figura 27 os números de parâmetros foram obtidos conforme a Tabela 22.

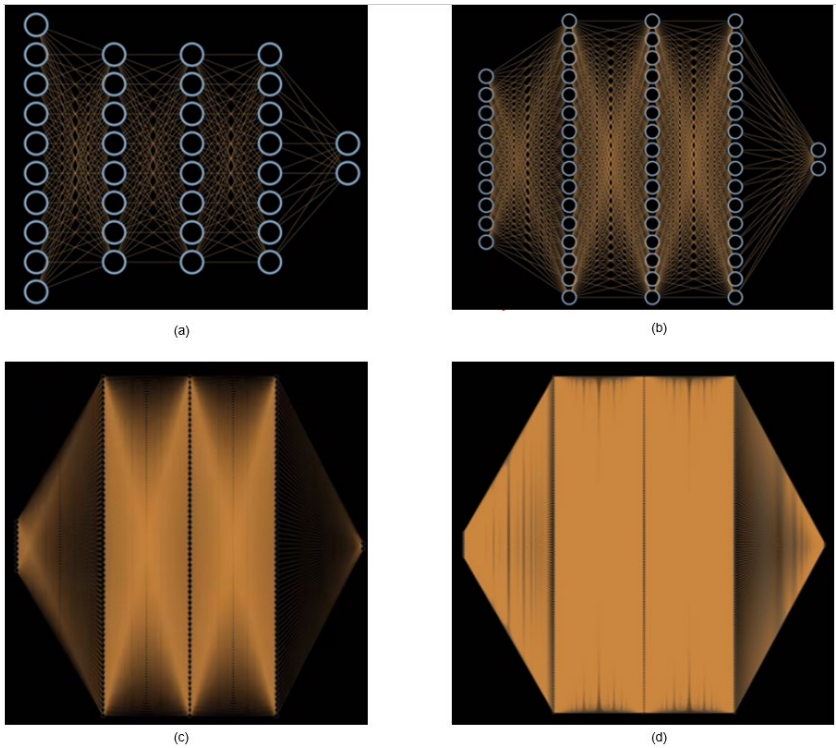
Tabela 22 — Cálculo de Parâmetros Aprendíveis dos gráficos da Figura 27

	Neurônios de Entradas	Neurônios (unidades) nas camadas ocultas (*hidden layer*)	Camadas ocultas	Neurônios na camada de saída	Total de Neurônios (camada oculta + camada saída)	Pesos (multiplicação entre número de neurônios das camadas	Viés (*bias*)	Parâmetros aprendíveis (Pesos + viés)
Rede Neural 1	3	4	1	2	6	20	6	26
Rede Neural 2	3	4	2	1	9	32	9	41

Fonte: Dimensionamento de redes neurais — Módulo 1 (KARPATHY, 2019) com apresentação realizada pela autora em formato de tabela.

O número de neurônios em cada camada tem impacto no número de parâmetros da rede neural, conforme pode ser verificado na Figura 28 em que na rede neural (a) há 10 neurônios de entrada, 8 neurônios em cada uma das 3 camadas ocultas e 2 neurônios de saída. Nas redes (b), (c), e (d) da Figura 28, o número de neurônios de entrada e saída, bem como o de camadas ocultas foram mantidos, mudando-se tão somente o de neurônios nas camadas ocultas para 16, 32 e 64 respectivamente.

Figura 28 — Redes diferenciadas conforme número de neurônios



Fonte: Dimensionamento de redes neurais — parte das imagens do vídeo (KINSLEY; KUKIETA, 2020).

Com essas mudanças, torna-se difícil a visualização dos neurônios das camadas e as conexões entre eles, principalmente na rede neural (d) da Figura 28.

Observa-se, então, que as redes de aprendizado profundo podem ter milhões de parâmetros e serem compostas por várias camadas, 10, 20 ou mais, tendo, em alguns tipos, um número muito maior de conexões efetivas em função do compartilhamento de parâmetros (KARPATHY, 2019).

Na Tabela 23, mantendo-se a coerência com apuração realizada na Tabela 22, foram calculados os números de parâmetros das redes neurais (a), (b), (c) e (d) da Figura 28.

Tabela 23 — Cálculo de Parâmetros Aprendíveis de acordo com aumento de neurônios

	Neurônios de Entradas	Neurônios (unidades) em camadas ocultas (*hidden layer*)	Camadas ocultas	Neurônios na camada de saída	Total de Neurônios (camada oculta + camada saída)	Pesos (multiplicação entre número de neurônios das camadas	Viés (*bias*)	Parâmetros aprendíveis (Pesos + viés)
Rede Neural (a)	10	8	3	2	26	224	26	250
Rede Neural (b)	10	16	3	2	50	704	50	754
Rede Neural (c)	10	32	3	2	98	2432	98	2530
Rede Neural (d)	10	64	3	2	194	8960	194	9154

Fonte: Dimensionamento de redes neurais da Figura 28 com apresentação adaptada pela autora no formato de tabela.

A organização em camadas e as correspondentes operações de matrizes entrelaçadas com a função de ativação permitem o uso e a avaliação mais eficiente e simples das redes neurais (KARPATHY, 2019).

Na composição de Rede Neural, além dos neurônios, há pesos e vieses (bias) aprendíveis (KARPATHY, 2019).

3.9 Funções de Ativação

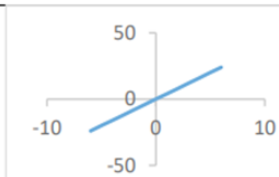
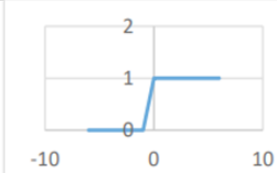
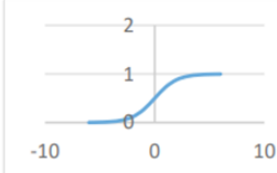
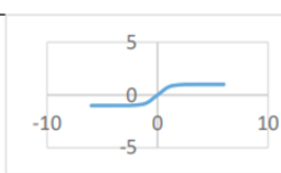
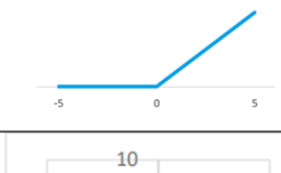
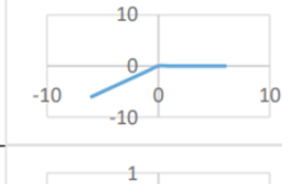
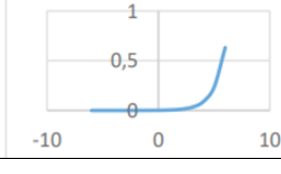
As funções de ativação são importantes para o bom funcionamento da rede neural, pois são expressões matemáticas que resultam em uma determinada saída quando a entrada atingir um determinado nível (valor) (Sampaio, 2018; KARPATHY, 2019).

As funções de ativação permitem introduzir características de não linearidade às redes neurais artificiais, considerando-se que a maioria dos problemas a serem resolvidos são não lineares (PINHEIRO, 2020).

Para entendimento das funções de ativação cabe destacar que gradiente se trata de cálculo de elementos com o fim da maximização do valor de uma função, adotando-se, assim, a derivada parcial da função em relação a "x" (PINHEIRO, 2020). A explosão e o desaparecimento do gradiente acontecem quando os pesos e vieses da rede neural não estiverem adequadamente definidos, muito altos para o primeiro problema ou o contrário para o segundo (PINHEIRO, 2020). A dissipação do gradiente ocorre quando o intervalo de valores for muito estreito (PINHEIRO, 2020).

No Quadro 2 são indicadas as representações gráficas, fórmulas e tratados os intervalos de operação de algumas funções de ativação.

Quadro 2 — Funções de Ativação

Nome	Plote	Equação	Derivada	Range
Linear (Identidade)		$f(x)=x$	$f'(x)=x$	$(-\infty,\infty)$
Degrau (Binária)		$f(x)=1$; se $x>=0$ 0; se $x<0$	$f'(x)=?$; se $x=0$ 0; se $x<>0$	(0,1)
Sigmoide (Logística)		$f(x)=1/(1+e^{-x})$	$f'(x)=f(x)(1-f(x))$	(0,1)
Tangente Hiperbólica		$f(x)=(e^{x}-e^{-x})/(e^{x}+e^{-x})$	$f'(x)=1-f(x)^2$	(-1,1)
Rectifier Linear (ReLu)		$f(x)=x$; se $x>=0$ 0; se $x<0$	$f'(x)=1$; se $x>=0$ 0; se $x<0$	$(0,\infty)$
Leaky ReLu		$f(x)=x$; se $x>=0$ $0{,}01x$; se $x<0$	$f'(x)=1$; se $x>=0$ $0{,}01$; se $x<0$	$(-\infty,\infty)$
Soft Max		$f(x)=(e^{xi})/(\Sigma^{K}_{j=1}e^{xj})$	$f'(x)=f(x)(1-f(x))$	(0,1)

Fonte: Tabela 3.2 de Gráficos das Funções de Ativação (PINHEIRO, 2020).

Destacam-se alguns fatos das principais funções de ativação.

- Sigmóide

A função Sigmóide ou Logística, com um formato como um “S”, tem sido utilizada nas opções binárias de ativação, tendo duas desvantagens principais, saturação na cauda 0 e 1 e o consequente prejuízo para gradientes, pois próximo dos limites 0 ou 1 o gradiente é praticamente zero e quase nenhum sinal fluirá via o neurônio (KARPATHY, 2019). A segunda desvantagem é não ter saída centrada em zero, o que causa impacto na dinâmica na descida do gradiente, tornando o gradiente totalmente negativo ou totalmente positivo (KARPATHY, 2019; PINHEIRO, 2020).

- Tanh

A função Tangente Hiperbólica (Tanh) tem sido utilizada nas opções binárias de ativação e, assim, como a Sigmóide tem a desvantagem da saturação, mas, por outro lado, a saída é centrada em zero, sendo mais recomendada em substituição à Sigmóide (KARPATHY, 2019; PINHEIRO, 2020; BOUZON, 2021).

- ReLU

A ReLU ou Unidade Linear Retificada é uma das funções mais utilizadas de ativação e é limitada a zero, contendo como vantagens em relação à Sigmóide e Tanh o aceleramento da descida do gradiente estocástico, implementação mais barata, por estabelecimento simples de um limiar de uma matriz de ativações em zero (KARPATHY, 2019; BOUZON, 2021). Entretanto, se a taxa de aprendizagem for definida muita alta, pode fazer com que o gradiente tenda a zero sempre (KARPATHY, 2019).

É muito utilizada em função de ser mais eficiente do que as funções anteriores e é responsável pela popularização de Deep Learning (PINHEIRO, 2020).

- Leaky ReLU

A implementação da função Leaky ReLU foi realizada na tentativa de corrigir o problema da ReLU, com uma pequena inclinação positiva (de 0,01 ou mais), com relatos de sucesso, apesar de nem sempre consistentes (KARPATHY, 2019).

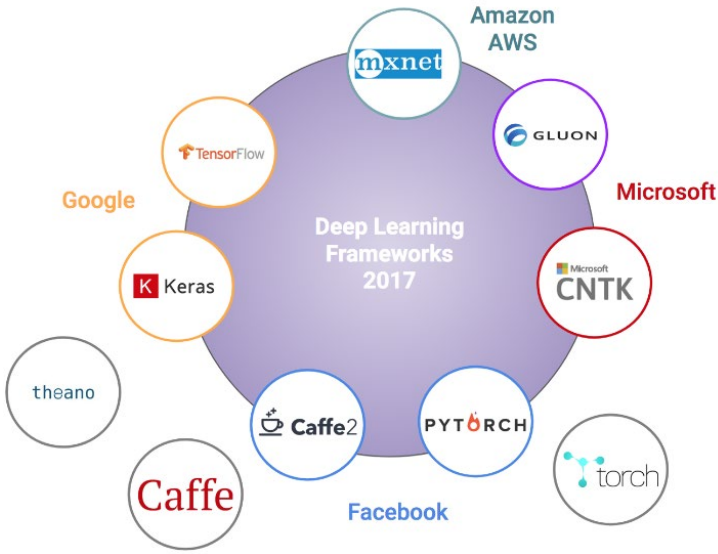
- Soft Max

A implementação da função Soft Max ou Regressão Logística permite, via probabilidade de classes, a definição da ativação a ser utilizada considerando-se o valor acima de 0,5 (KARPATHY, 2019

3.10 Deep Learning Frameworks

Uma das grandes razões do aumento da popularidade e adoção de Deep Learning é a oferta de ambiente Python com open source frameworks, desenvolvidos e apoiados por grandes empresas de tecnologia, conforme demonstrado na Figura 29 (SARKAR; BALI; GHOSH, 2018).

Figura 29 — Deep Learning Frameworks



Fonte: https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750

Keras é de fácil uso, permite que os pesquisadores e profissionais sejam mais produtivos e tem ampla adoção na indústria e na comunidade de pesquisa (KERAS, 2021). No final de 2021, há mais de um milhão de usuários individuais e a adoção de Keras com Tensorflow é favorita em diversos setores, sejam de mercado ou acadêmico, com várias menções em artigos científicos (KERAS, 2021).

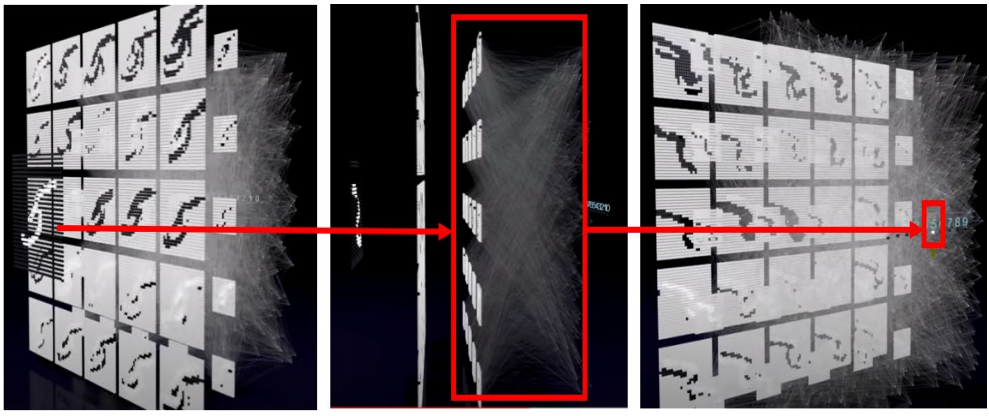
No site do Keras são disponibilizadas várias referências sendo que uma delas é relacionada com Aplicativos, ou seja, modelos de Deep Learning com pesos pré-treinados para uso na previsão, extração de recursos, ajustes finos (KERAS, 2021) e na classificação de imagens.

3.11 CNN

Como parte da evolução das MLP surgiu a Convolutional Neural Network (CNN ou ConvNet) sendo parte da Deep Learning utilizada no processamento e na classificação de imagens, vídeos, som, no processamento de linguagem natural, nos sistemas de recomendação e nas análises temporais (PINHEIRO, 2020).

Na Figura 30 há sequência do funcionamento de uma rede convolucional, em que podem ser observadas as transformações aplicadas na identificação de dígitos apresentados em diversos tipos de grafia.

Figura 30 — Rede Neural — Funcionamento de uma CNN

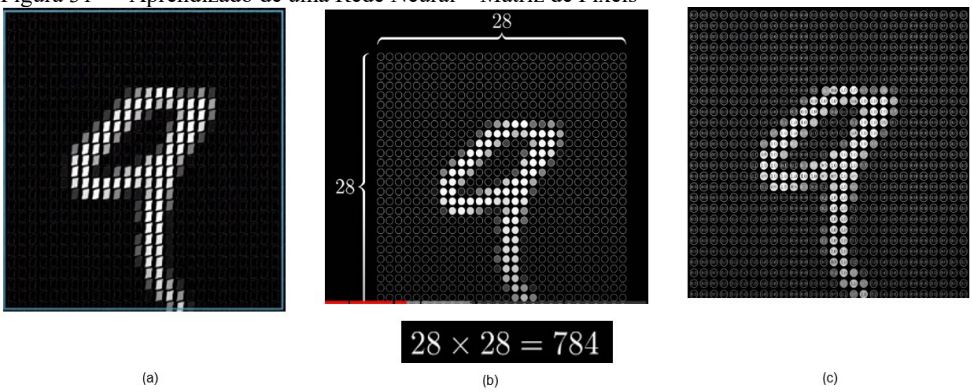


Fonte: Partes do vídeo de https://youtu.be/3JQ3hYko51Y adaptadas pela autora.

A CNN monta padrões mais complexos de rede neural, valendo-se do padrão hierárquico dos dados, utilizando padrões menores e mais simples, traçando, assim, paralelo com a organização do córtex visual dos animais, com respostas a estímulos apenas em uma região restrita do campo visual (PINHEIRO, 2020; SARKAR; BALI; GHOSH, 2018), compondo recursos de complexidade crescente em camadas sucessivamente organizadas (SARKAR; BALI; GHOSH, 2018).

Para o entendimento do funcionamento do aprendizado de uma de uma rede neural a partir da hierarquia de características, é apresentada na Figura 31 uma sequência de imagens a partir de um dígito até a sua identificação na última camada. A imagem (a) mostra o dígito 9, a (b), a matriz de pixels (picture e element, ou seja, elemento de imagem), sendo que a dimensão desta matriz de 784 pixels é obtida com a multiplicação da largura de 28 pela altura de 28. A cada pixel é associado um número que identifica a escala de branco a preto, conforme imagem (c).

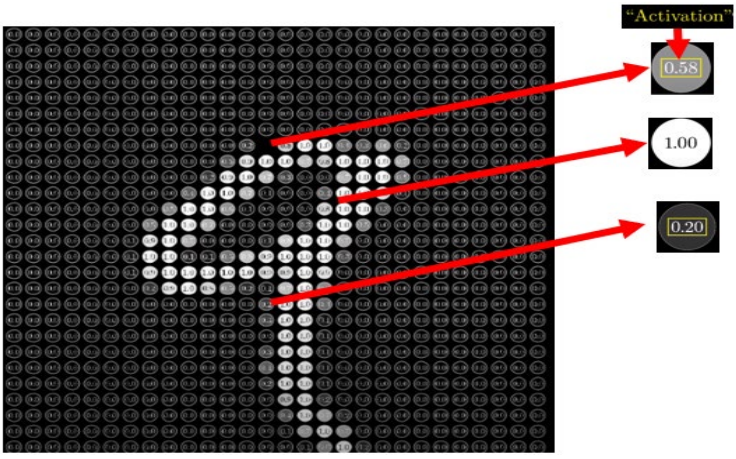
Figura 31 — Aprendizado de uma Rede Neural – Matriz de Pixels



Fonte: Partes do vídeo https://www.youtube.com/watch?v=aircAruvnKk&t=

Na Figura 32 são destacados os valores de pixels, conforme o número da ativação relacionado às cores.

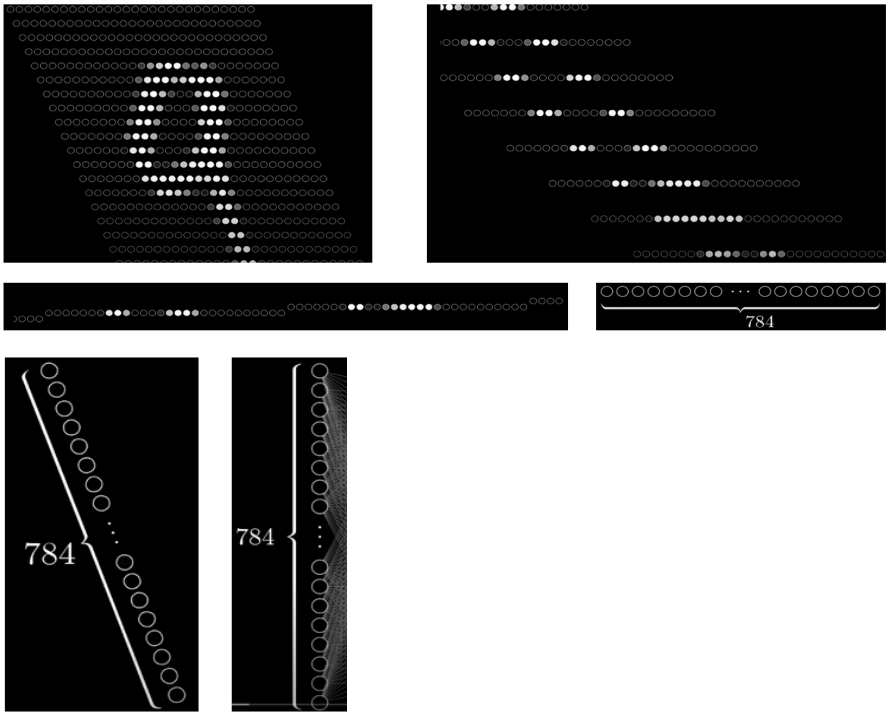
Figura 32 — Aprendizado de uma Rede Neural — Ativações de pixels



Fonte: Parte de vídeo https://www.youtube.com/watch?v=aircAruvnKk&t= adaptada pela autora.

Na sequência há transformações realizadas para que essa matriz de pixels se torne a camada de entrada da rede neural no formato de vetor, Figura 33.

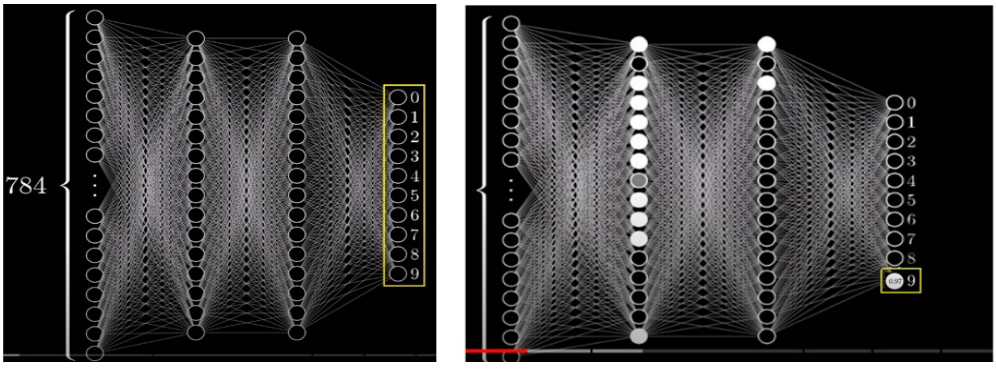
Figura 33 — Aprendizado de Rede Neural — Criação da camada de entrada no formato de vetor



Fonte https://www.youtube.com/watch?v=aircAruvnKk&t=

Na Figura 34 há a rede neural completa com as camadas de entrada, ocultas e de saída, sendo que nesta última são indicadas as categorias/classes esperadas.

Figura 34 — Aprendizado de uma Rede Neural completa

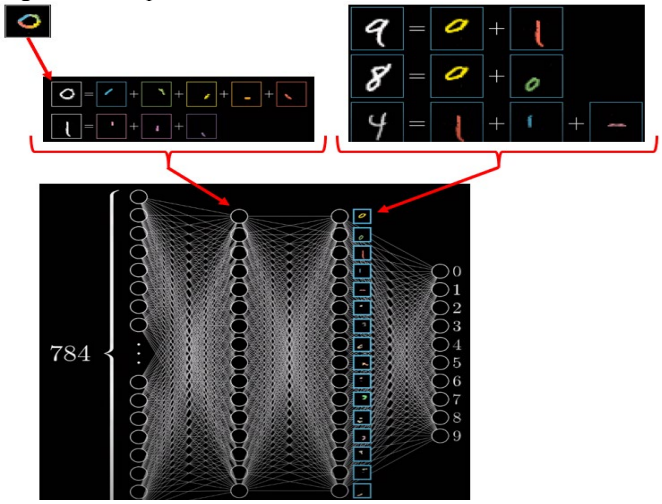


Fonte https://www.youtube.com/watch?v=aircAruvnKk&t=

..... Na

Figura 35 é detalhada a representação das características dos dígitos por linhas e bordas na primeira camada oculta e, na sequência, na segunda camada oculta, como esses elementos formam outros mais complexos e, estes, no que lhe concerne, podem ser utilizados na identificação de uma dada classe/categoria na camada de saída.

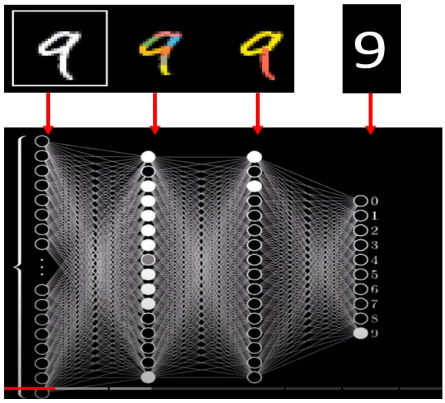
Figura 35 — Aprendizado de uma Rede Neural — Camadas com hierarquia de características



Fonte: Partes do vídeo https://www.youtube.com/watch?v=aircAruvnKk&t= adaptadas pela autora.

Na Figura 36 apresentam-se as camadas ocultas com as características de complexidades diferentes na composição do dígito 9 e a ativação da categoria pertinente na última camada da rede neural.

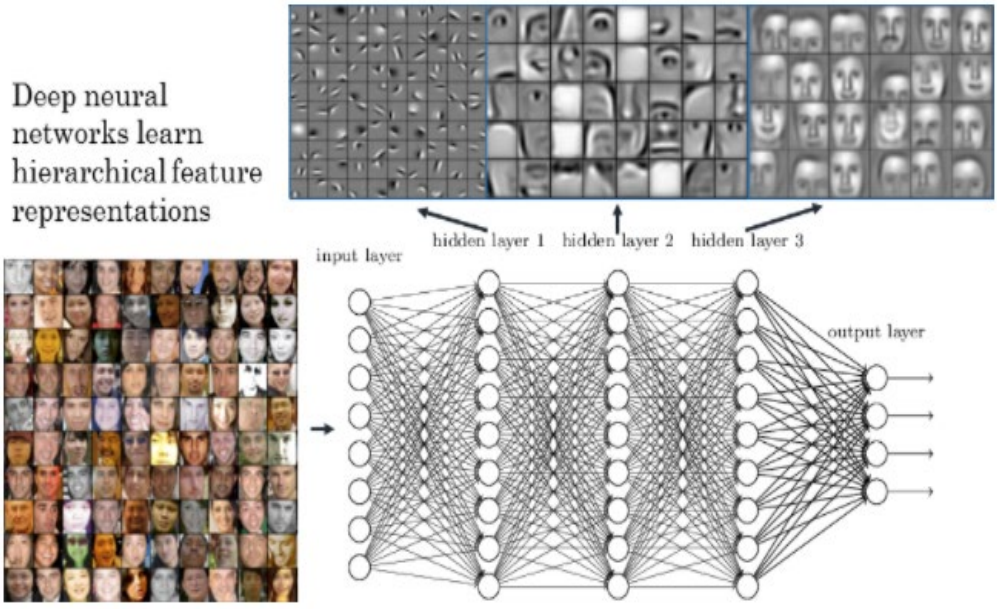
Figura 36 — Aprendizado de uma Rede Neural - Camadas com hierarquia até a classificação



Fonte https://www.youtube.com/watch?v=aircAruvnKk&t= adaptada pela autora.

Similarmente, na Figura 37, os mesmos procedimentos são aplicáveis em fotos, sendo que primeira camada oculta, no nível mais baixo, a rede neural atua na identificação bordas, linhas, arestas e padrões de contrastes locais, na segunda há uso dos itens da camada anterior na identificação de olhos, narizes, bocas, e outras características faciais, e na terceira camada, a superior, são utilizadas as características faciais da segunda camada em modelos de rostos (KARPATHY, 2019; SARKAR; BALI; GHOSH, 2018).

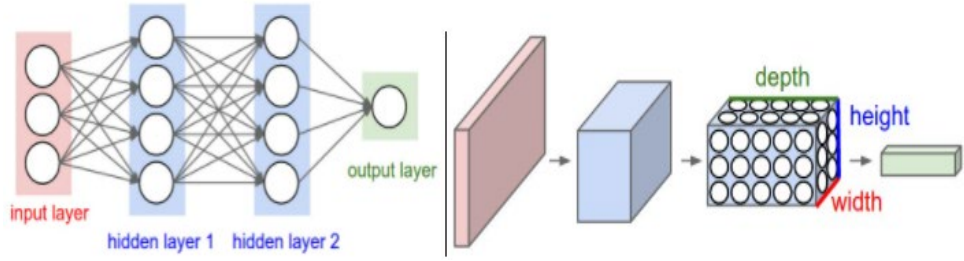
Figura 37 — Aprendizado de uma Rede Neural a partir da representação de hierarquia de características fotos



Fonte: https://rsipvision.com/exploring-deep-learning/

Destaca-se que CNN para imagens, por exemplo, valem-se de neurônios organizados em dimensões, ou seja, largura, altura e profundidade, o que não acontece com redes neurais convencionais (KARPATHY, 2019). Na Figura 38 é possível identificar as diferenças entre a primeira rede neural, a regular, e a segunda, a CNN, pois na última há a representação de largura, altura e profundidade da imagem.

Figura 38 — Rede Neural Regular Vs CNN



Fonte: Figura de Redes Neurais — regular e CNN — Módulo 2 (KARPATHY, 2019).

ConvNets processam dados na forma de múltiplas matrizes (LECUN; BENGIO; HINTON, 2015): “Muitas modalidades de dados estão na forma de vários arranjos: 1D para sinais e sequências, incluindo linguagem; 2D para imagens ou espectrogramas de áudio; e 3D para vídeo ou imagens volumétricas” (LECUN; BENGIO; HINTON, 2015).

Independentemente da dimensionalidade inicial do objeto de CNN, sua representação pode ser flattened (achatada), ou seja, convertida em um vetor de características (DUMOULIN; VISIN, 2018; RODRIGUES, 2019).

Assim, a essência de redes neurais CNN é a transformação que ocorre a partir da entrada inicial, seja imagem, som, vídeo, clipe de som ou uma coleção de características, considerando-se a multiplicação de vetor com uma matriz que produz uma saída, tendo-se a aplicação de viés (bias) antes da geração de uma saída (DUMOULIN; VISIN, 2018).

..... Na camada convolucional, como demonstrado da

Figura 35 à **Erro! Fonte de referência não encontrada.** as unidades são organizadas em mapas de features, sendo cada unidade conectada a patches (pedaços/recortes/regiões) de um mapa (uma matriz) de características anterior, via filtros (kernel, matrizes de pesos que se movimentam pela imagem, mapa de características), através de soma ponderada (LECUN; BENGIO; HINTON, 2015; DUMOULIN; VISIN, 2018).

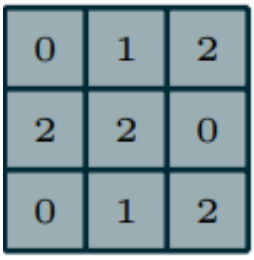
A definição simplificada de Convolução é, então, a de uma operação matemática, com o somatório de produto entre os filtros e as regiões locais da entrada, com fim de recuperar características (KARPATHY, 2019).

Como o foco deste trabalho é o diagnóstico de COVID-19 a partir de raios-X, trataremos de imagem no detalhamento de CNN.

O primeiro detalhe que devemos destacar é que a imagem tem uma estrutura intrínseca que pode ser armazenada em arrays, como pode ser observado na Figura 31 e Figura 38 multidimensionais, com eixos para altura, largura e profundidade, sendo que cada um deles é utilizado para diferente visão do dado (DUMOULIN; VISIN, 2018).

Exemplificando, no funcionamento de uma CNN, há aplicação de um filtro da Figura 39 na matriz inicial da imagem.

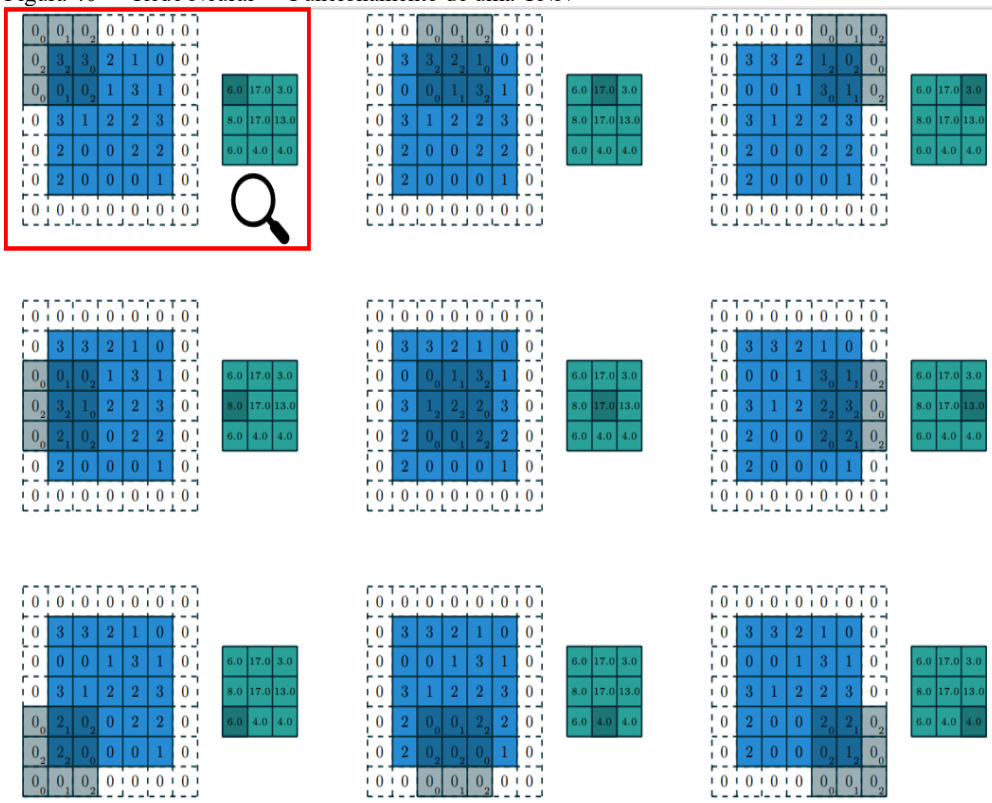
Figura 39 — Filtro



Fonte: Figura 1.1 — Filtro (DUMOULIN; VISIN, 2018)

A imagem inicial já está ajustada com zero padding, preenchimento de zeros nas bordas da imagem original, para que o conteúdo da matriz da imagem tenha aplicação do filtro em todos os seus componentes, conforme valor de stride (ritmo de passo/deslizamento/passada), de acordo com Figura 40 (DUMOULIN; VISIN, 2018).

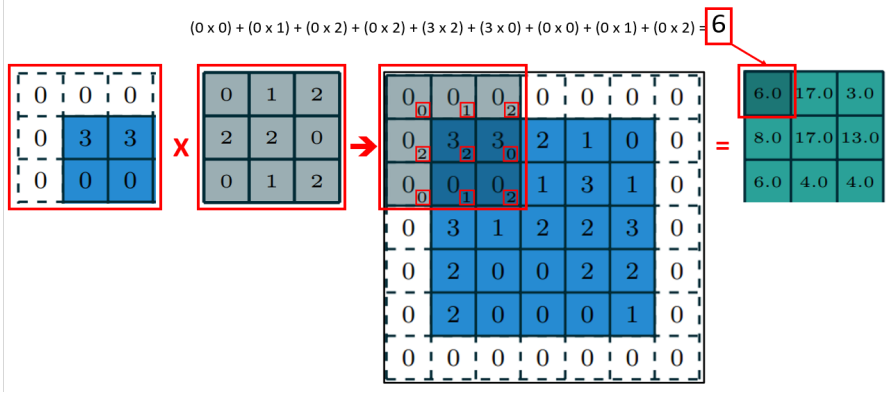
Figura 40 — Rede Neural — Funcionamento de uma CNN



Fonte: Adaptação da Figura 1.2 — Aplicação do Filtro (DUMOULIN; VISIN, 2018)

Na sequência, verifica-se que na Figura 41 há a aplicação do filtro na parte destacada com uma lupa na Figura 40.

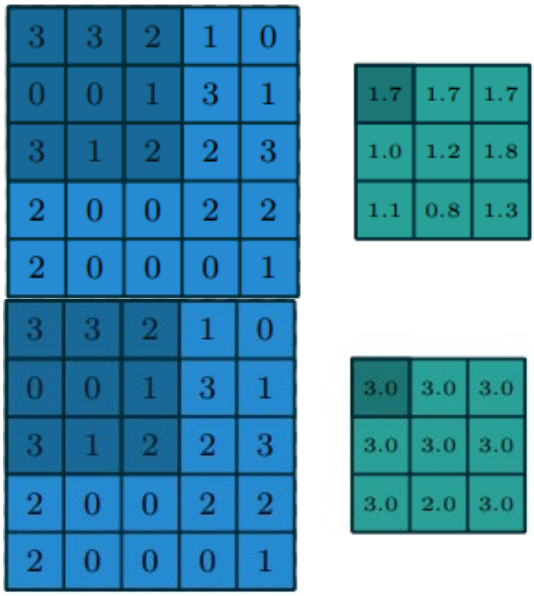
Figura 41 — Rede Neural — Aplicação de Filtro



Fonte: Adaptação da autora a partir da Figura 1.2 de aplicação do Filtro (DUMOULIN; VISIN, 2018)

Na sequência, na imagem (a) da Figura 42 há a aplicação da operação pooling ou subsampling (agrupamento para reduzir matrizes oriundas das convoluções, reduzindo, assim, a quantidade de parâmetros aprendidos) pela média simples e na imagem (b) da mesma figura, por MaxPooling.

Figura 42 — Aplicação de Pooling



a) Pooling pela Média b) Pooling pelo valor máximo

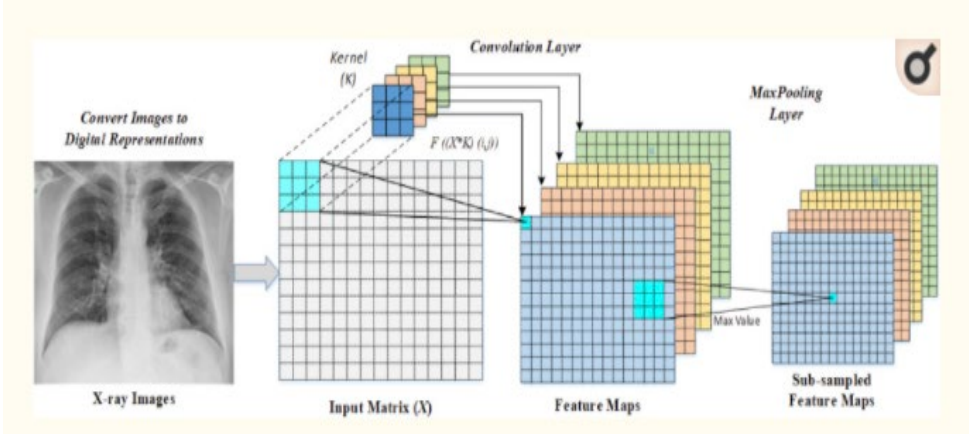
Fonte: Pooling — Figuras 1.5 e 1.6 (DUMOULIN; VISIN, 2018)

A camada de flattening ocorre com a transformação de imagens de saída em um vetor de características depois da sequência das camadas convolucionais e pooling (RODRIGUES, 2019).

Ao final, será realizada a previsão ou classificação binária ou entre diversas classes/categorias, conforme o caso (RODRIGUES, 2019).

Por sua vez, na Figura 43 há a sequência do funcionamento de uma rede convolucional, onde as transformações aplicadas em uma radiografia podem ser observadas até a camada de MaxPooling.

Figura 43 — Aplicação de CNN em imagem de um de raios-X



Fonte: Figura 3 — Apresentação esquemática das operações de convoluções e camada de Maxpooling (OZTURK et al., 2020).

Em uma CNN há o aproveitamento das propriedades dos sinais naturais a saber: conexões locais; pesos compartilhados; pooling ou subsampling; e o uso de muitas camadas (LECUN; BENGIO; HINTON, 2015).

3.11.1 CNN – Evolução

- LeNET

A pioneira das CNN foi a LeNET, apresentada por LeCUN e coautores, em 1998, na classificação de caracteres (SARKAR; BALI; GHOSH, 2018), estruturada com camadas de convolução, de pooling, uma totalmente conectada e uma de ativação softmax (RODRIGUES, 2018).

- ImageNet

Um passo importante de evolução foi o uso CNN na classificação da ImageNet, uma base de dados com centenas de milhares imagens organizadas por categorias seguindo a hierarquia WordNet (SARKAR; BALI; GHOSH, 2018).

- AlexNet

AlexNet, 2012, uma arquitetura similar à LeNet, proposta por Krizhevsky e coautores, com aplicação de mais filtros por camada, com mais profundidade a

introdução do conceito de empilhamento de convoluções (stacked convolution), permitindo a redução de erros a 15,3% (SARKAR; BALI; GHOSH, 2018). “Essa arquitetura utiliza regularização dropout para reduzir overfitting nas camadas totalmente conectadas e ReLU como função de ativação nas camadas convolucionais e totalmente conectadas” (RODRIGUES, 2018).

- ZFNet

ZFNet, 2013, com iniciais dos sobrenomes de seus autores Matthew Zeiler and Rob Fergus, aprimorou a AlexNET, ajustando hiperparâmetros, com a expansão das camadas convolucionais intermediárias e a definição de stride e tamanho de filtro menores para a primeira camada, permitindo, assim, a retenção muito maior de informações originais dos pixels (SARKAR; BALI; GHOSH, 2018).

- GoogleNet (Inception Network)

GoogleNet, 2014, da Google, reduziu o erro a 6,67%, aproximando-se à performance humana de predição, introduzindo inception layer como um novo componente arquitetural, com o uso de convoluções maiores. (SARKAR; BALI; GHOSH, 2018). É uma arquitetura muito profunda (RODRIGUES, 2018).

- VGG

A Visual Geometric Group (VGG), 2014, rede neural criada por grupo de Oxford de mesmo nome, considerando-se a simplicidade, com o uso de camadas convolucionais empilhadas no topo em crescente profundidade, reduzindo o volume a ser trabalhado na camada de maxpooling, confirmando a impacto na profundidade em representações de imagens (SARKAR; BALI; GHOSH, 2018). Inovou com a substituição de filtros maiores por grandes sequências de filtros, reduzindo o custo computacional (RODRIGUES, 2018).

- ResNet

A Residual Neural Network (ResnNet), 2015, introduzida por Kaiming He e coautores da Microsoft Research Asia, é uma arquitetura que salta conexões e normalizações batch, tornando possível o treinamento de uma rede neural mais profunda do que a VGG, com 152 camadas, mas mesmo assim com menor grau de

complexidade e reduzindo o erro para 3,57%, o que supera a performance humana, (SARKAR; BALI; GHOSH, 2018). Compõe-se de blocos residuais, com várias camadas convolucionais empilhadas (RODRIGUES, 2018).

Na Figura 44 há exemplos da representação de modelos CNN.

Figura 44 — Modelos CNN

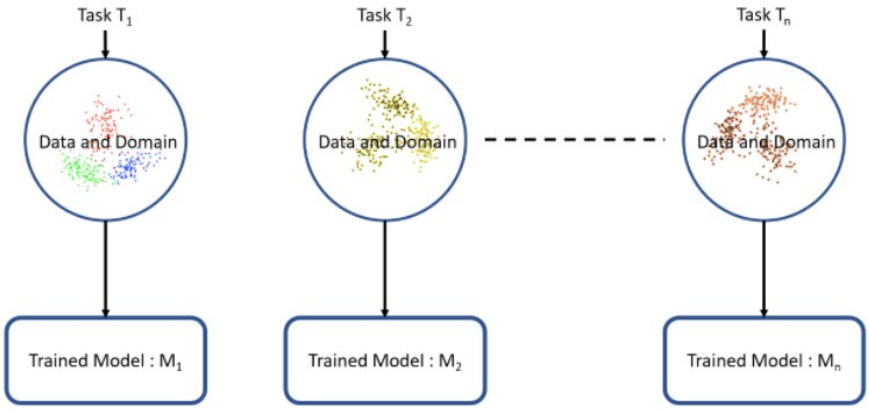
Fonte: Figura 2.12 — Modelos CNN com pequena adaptação pela autora (RODRIGUES, 2018).

3.12 Transfer Learning

Uma importante contribuição para o treinamento em Deep Learning é a utilização de modelos pré-treinados, técnica conhecida como Transfer Learning, viabilizando uma abordagem diferente da adotada em treinamentos isolados, com

tarefas, “T”, domínio e dados específicos para modelos específicos, “M”, como consta na Figura 45 (SARKAR; BALI; GHOSH, 2018).

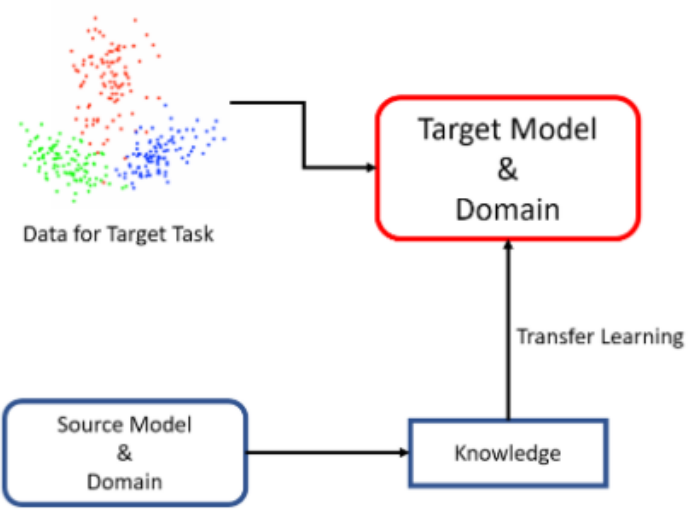
Figura 45 — Deep Learning tradicional



Fonte: Figura (SARKAR, Dipanjan; BALI, Raghav; GHOSH, Tamoghna., 2018).

Com a implementação de Transfer Learning o conhecimento de uma tarefa existente é utilizado como uma entrada adicional em um novo aprendizado, conforme Figura 46.

Figura 46 — Deep Learnig com Transfer Learning



Fonte: Figura (SARKAR, Dipanjan; BALI, Raghav; GHOSH, Tamoghna., 2018).

A técnica de Transfer Learning permite o uso de camadas iniciais pré-treinadas, deixando para as últimas camadas as alterações e refinamento de pesos e parâmetros, agregando vantagens tais como o uso do desempenho da baseline, redução do tempo de treinamento, aumento do desempenho final da implementação realizada (SARKAR; BALI; GHOSH, 2018).

Há uma série de modelos pré-treinados compartilhados na forma de milhões de parâmetros/pesos, disponíveis em bibliotecas diversas, como a Keras, para diferentes usos (SARKAR; BALI; GHOSH, 2018).

Na seção 0 foram elencados alguns modelos pré-treinados.

Há aplicações de Transfer Learning com dados textuais, computação visual e fala/áudio (SARKAR; BALI; GHOSH, 2018).

Com o uso de Transfer Learning, aproveita-se todo o trabalho já realizado nas diversas camadas convolucionais até a camada de flattening, reutilizando-se o aprendizado e os pesos obtidos, restando para uma nova e específica rede neural a implementação das camadas densas. Assim, a inteligência e o aprendizado são transferidos de uma rede neural base para uma nova.

4 MÉTRICAS

As métricas são importantes recursos na avaliação dos modelos treinados e os resultados obtidos.

4.1 Matriz de Confusão

A Matriz de Confusão demonstra a relação entre as classes rotuladas reais e as previstas, com a contagem correspondente, demonstrando o desempenho da classificação (BOUZON, 2021).

A contagem realizada na Matriz de Confusão é realizada tendo em vista Verdadeiro Positivo (VP), Verdadeiro Negativo (VN), Falso Positivo (FP) e Falso Negativo (FN) (BOUZON, 2021), conforme esclarecimento a seguir:

“Verdadeiro positivo (TP): número de predições que o classificador previu corretamente uma classe positiva como positiva” (BOUZON, 2021);

"Verdadeiro negativo (TN): número de predições que o classificador previu corretamente uma classe negativa como negativa" (BOUZON, 2021);

"Falso positivo (FP): número de predições que o classificador previu uma classe negativa incorretamente como positiva" (BOUZON, 2021);

"Falso negativo (FN): número de predições que o classificador previu uma classe positiva incorretamente como negativa" (BOUZON, 2021).

A seguir são apresentadas outras métricas decorrentes da Matriz de Confusão:

4.1.1 Acurácia

Indica a classificação correta realizada, conforme Equação (1) (BOUZON, 2021; SARKAR; BALI; GHOSH, 2018).

$$\textbf{Acurácia} = \frac{(\textbf{TP} + \textbf{TN})}{(\textbf{TP} + \textbf{TN} + \textbf{FP} + \textbf{FN})} \quad (5)$$

Fonte: (BOUZON, 2021)

4.1.2 Precisão

Indica o quanto as predições de classes são realmente positivas, ou seja, poder predição do modelo, conforme Equação (6) (BOUZON, 2021; SARKAR; BALI; GHOSH, 2018).

$$\textbf{Precisão} = \frac{\textbf{TP}}{(\textbf{TP} + \textbf{FP})} \quad (6)$$

Fonte: (BOUZON, 2021)

4.1.3 Sensibilidade — Recall — Hit Rate — True Positive Rate (TPR)

Sensibilidade, Recall, Hit Rate ou True Positive Rate (TPR) indica o quanto as classificações da classe positiva foram classificadas corretamente, ou seja, a

capacidade de recuperar elementos da classe de interesse, conforme Equação (7) (BOUZON, 2021; SARKAR; BALI; GHOSH, 2018).

$$\mathbf{TPR} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FN})} \tag{7}$$

Fonte: (BOUZON, 2021)

4.1.4 F1-Score

Indica a média harmônica entre Precisão e Sensibilidade, conforme Equação (8) (BOUZON, 2021; SARKAR; BALI; GHOSH, 2018).

$$\mathbf{F1 - Score} = \frac{\mathbf{2TP}}{(\mathbf{2TP} + \mathbf{FP} + \mathbf{FN})} \tag{8}$$

Fonte: (BOUZON, 2021)

5 ÉTICA, LEGALIDADE E LGPD

Apesar do consenso da premência de acesso amplo e rápido de informações pela comunidade de pesquisadores em situações críticas como epidemias e pandemias, cabem cuidados referentes ao respeito à "proteção dos dados pessoais, à tutela da privacidade e ao princípio da não discriminação, tocando as dimensões individual e coletiva", atendo-se, inclusive, aos diversos intervenientes que podem ser envolvidos e, consequentemente, os riscos correspondentes (MACHADO; SCHERTEL, 2020).

Complementarmente, alerta-se que os dados de saúde são constantemente alvo de crimes virtuais, citando vários exemplos de ataques a base de dados de pacientes e rede hospitalar, inclusive no Brasil, e que os avanços no uso de TI não devem comprometer a ética e a vida das pessoas (KFOURI NETO; SILVA; NOGAROLI, 2020). Exemplificando, "nos Estados Unidos, apenas em 2015, ocorreram 51 incidentes com hackers em instituições de saúde", "entre os anos de 2018 e 2020, 562 episódios de invasões de dados em organizações da saúde foram reportados ao U.S. Department of Health and Human Services Office for Civil

Rights" (KFOURI NETO; SILVA; NOGAROLI, 2020). O Brasil é alvo de vários ciberataques, fato comprovado por cerca de 15 bilhões de ataques apenas nos três primeiros meses de 2020 (KFOURI NETO; SILVA; NOGAROLI, 2020).

Na linha do mau uso dos dados e estudos resultantes, no Brasil prevalece o uso de tecnologias de perfilamento a partir de "processamento de dados agregados de geolocalização de dispositivos móveis para combater a COVID-19", o que "pode gerar riscos de reidentificação dos usuários de telefones celulares por ataques inferenciais (MIA) e de desvirtuamento de função e finalidade originária do tratamento de dados" (MACHADO; SCHERTEL, 2020) .

Perfilamento é uma técnica de tratamento (parcialmente) automatizado de dados pessoais e/ou não pessoais, que visa a produção de conhecimento por meio da inferência de correlações de dados na forma de perfis que podem ser posteriormente aplicados como base para a tomada de decisão. Um perfil é um conjunto de dados correlacionados que representa um sujeito (individual ou coletivo). A construção de perfis é o processo de descoberta de padrões desconhecidos entre dados em grandes bases de dados que podem ser usados para criar perfis. A aplicação de perfis é o processo de identificação e representação de um indivíduo ou grupo específico como adequado a um perfil, e de tomada de alguma forma de decisão com base nessa identificação ou representação; ((MACHADO; SCHERTEL, 2020); grifo da autora)

Além das questões apresentadas, eventuais imprecisões ou imprevistos na aplicação de inteligência artificial podem ocorrer sendo recomendados que médicos alertem sobre esse risco, pois danos podem ser causados aos pacientes, além das implicações ético-jurídicas na área de saúde (KFOURI NETO; SILVA; NOGAROLI, 2020).

Destaca-se, ainda, que os dados pessoais relativos à saúde, apesar de fonte primária para soluções de interesse do setor médico, "tendem a ser, por natureza, dados pessoais sensíveis" (KFOURI NETO; SILVA; NOGAROLI, 2020).

Nesse contexto, no Brasil, a Lei nº 13.709, de 14 de agosto de 2018 - Lei Geral de Proteção de Dados (LGPD), tendo os primeiros artigos com vigência iniciada em 28 de dezembro de 2018 e os últimos, 1º de agosto de 2021, é "voltada ao indivíduo, pessoa natural identificada ou identificável, e possui elementos

normativos que comportam interesses coletivos e sua tutela" (MACHADO; SCHERTEL, 2020) consolidando-se como um marco na proteção de dados.

Na LGPD, dados relativos à saúde são parte do conceito de "dado pessoal sensível" (KFOURI NETO; SILVA; NOGAROLI, 2020). Após mencionar dados pessoais, faz-se necessário resgatar os conceitos estabelecidos na própria LGPD, com a distinção dos que são definidos como sensíveis:

Art. 5º Para os fins desta Lei, considera-se:

I - dado pessoal: informação relacionada a pessoa natural identificada ou identificável;

II - dado pessoal sensível: dado pessoal sobre origem racial ou étnica, convicção religiosa, opinião política, filiação a sindicato ou a organização de caráter religioso, filosófico ou político, dado referente à saúde ou à vida sexual, dado genético ou biométrico, quando vinculado a uma pessoa natural. (BRASIL, 2018), grifo da autora).

A referida Lei não exige o consentimento, definido no inciso XI do art. 5º como "manifestação livre, informada e inequívoca pela qual o titular concorda com o tratamento de seus dados pessoais para uma finalidade determinada" (BRASIL, 2018), nas situações onde o propósito é o "tratamento de dados pessoais sensíveis excepcionais" tais como alguns relacionados com saúde destinados à "proteção da vida ou da incolumidade física do titular ou de terceiros" e "tutela da saúde, exclusivamente, em procedimentos realizados por profissionais de saúde, serviços de saúde ou autoridade sanitária" (KFOURI NETO; SILVA; NOGAROLI, 2020; BRASIL, 2018). Entretanto, o uso dos dados deve ser alertado ao paciente e a sua conservação após o uso autorizado pelas alíneas "e" e "f" do inciso II do art. 11 da LGPD pode ser realizada de forma excepcional conforme previsto na referida lei, adotando-se procedimentos de garantia de sigilo como anonimização dos dados pessoais, com descaracterização dos dados com o intuito de que seu titular não possa ser identificado, ou com consentimento do titular da informação (KFOURI NETO; SILVA; NOGAROLI, 2020; BRASIL, 2018).

Apesar da existência da LGPD (BRASIL, 2018) e de quaisquer outros atos, de forma complementar, os autores Machado e Schertel (2020) propõem que sejam consideradas como balizas hermenêuticas: a proteção de dados desde a concepção, em infraestruturas de informações e comunicação de suporte aos fluxos de dados, principalmente as referentes à computação preemptiva, integrada pelas tecnologias de perfilamento e analítica de dados; dados não pessoais devem ser tidos como pessoais se adotados para "formação de perfil comportamental"; o exercício do direito à explicação dos critérios e lógicas empregados na dimensão de grupo; e uso de Relatório de Impacto à Proteção de Dados Pessoais (RIPDP) como forma de salvaguardar entes públicos e privados no uso de "tecnologias de perfilamento com o tratamento de dados agregados de geolocalização".

6 METODOLOGIA

6.1 Tipo de Trabalho

O tipo de trabalho desenvolvido é quali-quantitativo aplicado em caso prático.

É qualitativo tendo em vista a pesquisa observacional positivista realizada, mais especificamente descritiva ou exploratória.

Também é quantitativa em função dos critérios utilizados na escolha dos algoritmos adotados neste trabalho e da análise posterior realizada das métricas apuradas em suas execuções.

6.2 Procedimentos

A dinâmica envolvida na realização do trabalho está registrada no diagrama resumo da Figura 47.

Figura 47 — Diagrama Resumo dos Procedimentos Metodológicos

Fonte: Diagrama produzido pela autora.

A tarefa inicial foi de uma pesquisa exploratória, com levantamento Bibliográfico conforme Figura 48 tendo em vista conceitos e, também, experiências no Brasil e no exterior, considerando-se iniciativas de aplicação de IA utilizando-se principalmente das bases bibliográficas "Google Acadêmico", de instituições nacionais e internacionais, com descritores e palavras-chave como "Covid", "diagnóstico", "prognóstico", "inteligência artificial", "Deep Learning", "Machine Learning", "Transfer Learning", "raios-X". As referências foram escolhidas verificando-se o número de citações em outros trabalhos, atualidade, clareza, gratuidade e fundamentações apresentadas em textos de artigos e teses de doutorado, mestrado, especializações, graduação ou documentos científicos e outros meios técnicos especializados.

Figura 48 — Levantamento Bibliográfico

Fonte: Diagrama produzido pela autora.

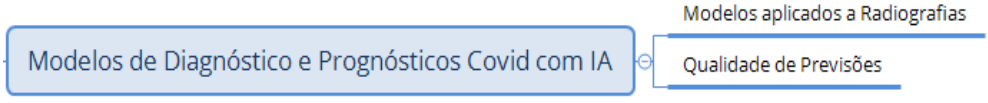
De posse de todo o material levantado, houve a realização de Consolidação da Fundamentação Teórica, distribuindo-se os achados conforme a estruturação da Figura 49.

Figura 49 — Consolidação — Fundamentação Teórica

Fonte: Diagrama produzido pela autora.

Na sequência os Modelos de Diagnósticos e Prognósticos COVID-19 com IA foram apresentados na Figura 50, focando-se os estudos realizados e os casos considerados pela autora como estado da arte.

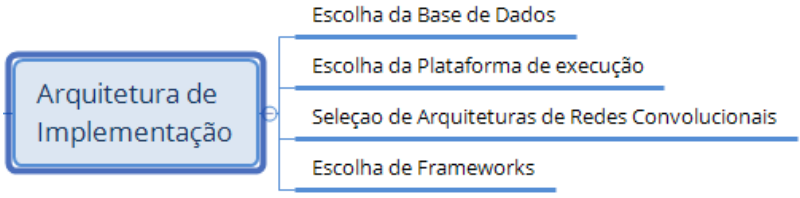
Figura 50 — Modelos de Diagnóstico e Prognóstico de COVID-19 com IA



Fonte: Diagrama produzido pela autora.

Após todos os levantamentos iniciais, foi realizada a escolha e a definição da Arquitetura de Implementação, Figura 51, destacando-se a base de dados a ser adotada, a plataforma de execução, as arquiteturas de redes neurais Convolucionais e os frameworks a serem implementados.

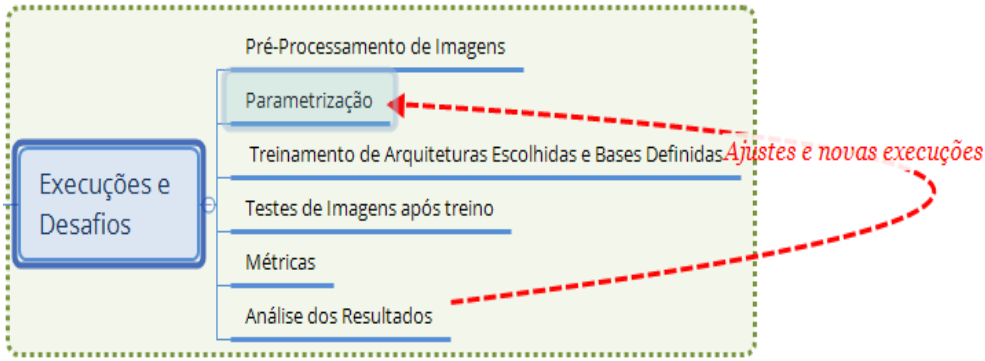
Figura 51 — Arquitetura de Implementação



Fonte: Diagrama produzido pela autora.

Estabelecido todo o ambiente necessário, foram testados os modelos escolhidos, a partir do pré-processamento de Imagens, com a realização de treinamento, testes e apuração de métricas, a análise de resultados, o registro de desafios encontrados, a aplicação de ajustes de parametrizações e de balanceamento das classes de imagens, Figura 52. Este ciclo iterativo de execuções foi realizado até a obtenção dos melhores resultados possíveis de acordo os modelos e bases de dados escolhidos para este trabalho.

Figura 52 — Execuções e Desafios



Fonte: Diagrama produzido pela autora.

Por fim, foi formatada a apresentação de resultados obtidos durante as execuções e registro de desafios.

7 MODELOS DE DIAGNÓSTICOS E PROGNÓSTICOS DE COVID-19

Foi realizada busca na literatura de modelos de diagnóstico e prognósticos de COVID-19 a partir da aplicação de inteligência artificial.

Em um artigo muito referenciado na literatura há a revisão sistemática e avaliação crítica (WYNANTS et al., 2020), em que foram abordados e validados modelos disponíveis de previsão referentes à COVID-19, abordando-se o diagnóstico da doença e o prognóstico de pacientes, a identificação de pessoas na população com maiores riscos de infecção e de internação. Neste estudo, até 5 de maio de 2020, foram recuperados 37.421 títulos, sendo que dos 232 estudos incluídos na revisão de Wynants et al. (2020) 7 foram relacionados a modelos de identificação de pessoas de maior risco de contaminação na população em geral, 118, a diagnósticos (75 a partir de imagens), 107, a prognósticos (incluindo 39 para mortalidade, 28 para progressão para estado grave ou crítico).

No Quadro 3 são apresentadas as principais características dos modelos preditivos e de prognósticos pesquisados por Wynants et al. (2020).

Quadro 3 — Modelos de Predição identificados por Wynants et al. (2020)

Tipos de Modelos	Quantidade Modelos Identificados	Preditores/Fatores	Objetivo	Riscos de viés
Previsão do risco de Covid-19 na população em geral	7	Idade, sexo, internação hospitalar anterior, comorbidades e determinantes sociais de saúde, vídeos térmicos de rostos de pessoas usando máscaras para determinar respiração anormal. Dados demográficos, sintomas e histórico de contatos. Exames adicionais de sangue e Tomografia Computorizada.		Risco incerto ou de viés alto
Diagnóstico de Covid-19 em pacientes com suspeita de infecção	33	**Sinais vitais:** temperatura, frequência cardíaca, frequência respiratória, saturação de oxigênio, pressão arterial **Sintomas similares ao de gripe:** calafrio e fadiga. **Outros preditores:** Idade, eletrólitos, Tomografia computadorizada com imagem compatível com pneumonia. Contato com indivíduos confirmados de Covid-19. Contagem de linfócitos, de neutrófilos, de leucócito, hemácias, enzimas hepáticas . Tosse ou expectoração, sexo. Em casos graves, preditores utilizados foram comorbidades, enzimas hepáticas, proteína C reativa, características de imagem, contagem de linfócitos e neutrófilos. Uso de espectogramas de sons de tosse, ultrosonografia pulmonar, radiografia pulmonar com o fim de monitorar a progressão da doença		Fontes de dados inadequadas. Dados não representativos da população-alvo. Carência de informações claras sobre as etapas de pré-processamento.
Prognósticos para pacientes com diagnóstico de Covid-19	107	Idade, cormobidades, sinais vitais, características de imagem, sexo, contagem de linfócitos e proteína C reativa.	Resultados - Previsão de progressão: estágio grave ou crítico da doença; recuperação; tempo de internação hospitalar, necessidade de Unidade de Terapia Intenciva,intubação, ventilação mecânica, síndrome do desconforto respiratório agudo, lesão cardíaca e complicação trombótica.	Inclusão e exclusão de participantes.

Fonte: Produzido pela autora do trabalho com os dados oriundos de Wynants et al. (2020)

Corroborando com os dados do Quadro 3 — Modelos de Predição identificados por Wynants et al. (2020)Quadro 3, Jimenez-Solem et al. (2021) dedicaram-se a identificar características importantes e impulsionadores da progressão da COVID-19 em pacientes positivos para a doença, incluindo os diagnosticados fora dos hospitais. Os preditores definidos pela pesquisa Jimenez-Solem et al. (2021) foram idade, sexo, índice de massa corporal, comorbidades como diabetes, hipertensão, insuficiência cardíaca e doença neurológica.

Na mesma linha, outros pesquisadores dedicaram-se ao uso de algoritmos de Inteligência Artificial para o aumento do desempenho do diagnóstico de COVID-19 a partir de imagens, (ASLAN et al., 2021), e de tosse e fala (LAGUARTA; HUETO; SUBIRANA, 2020), mesmo em pacientes assintomáticos ou com sintomas nem sempre facilmente identificados por exames tradicionais.

7.1 Modelos aplicados a Radiografias

Do levantamento realizado neste trabalho sobre uso de Machine Learning e Deep Learning no diagnóstico e prognóstico de COVID-19 a partir de análise de imagens de raios-X, destaca-se nesta seção alguns trabalhos realizados, procedimentos e algoritmos adotados, considerando-se o

Quadro 4.

Quadro 4 — Síntese de Modelos de IA aplicados a Radiografias

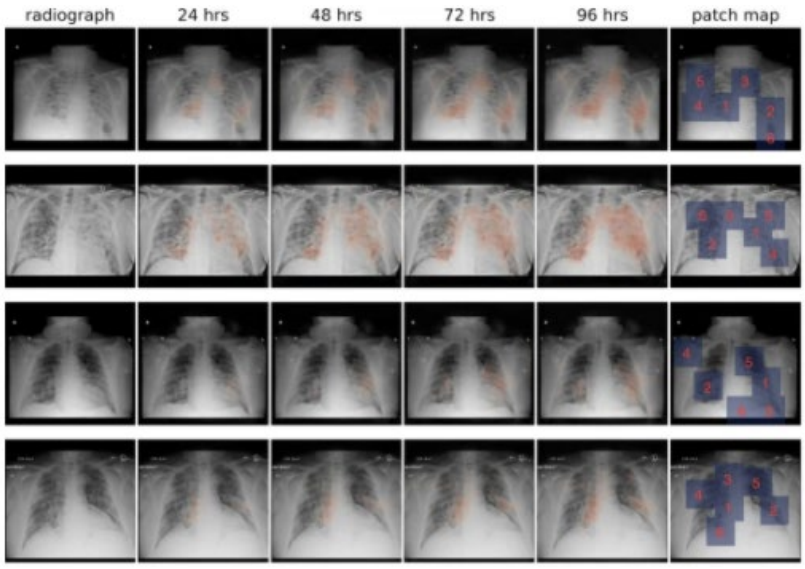
Trabalho Realizado	Foco	Arquitetura (DL/ML)	Análise	Base de Dados			Desempenho
				Origem	Volume	Organização	
01 - Um sistema de inteligência artificial para prever a deterioração de pacientes COVID-19 no departamento de emergência (SHAMOUT, Farah E. et al, 2021).	> **Prognóstico** - Avaliação automática do risco de deterioração, combinado com variáveis clínicas coletadas rotineiramente.	> **COVID-GMIC-DRC** - *Globally Aware Multiple Instance Classifier* (GMIC), *Deterioration Risk Curve* s - previsão de riscos de deterioração de pacientes COVID-19 no departamento de emergência em diferentes horizontes de tempo - 24, 48, 72 e 96 horas.	> Cálcula **mapa de saliências** com destaque das regiões utilizadas para previsões.	NYU Langone Health in New York, USA - 03/03/2020 à 13/05/2020	> 3.661 pacientes. > 5.994 exames - radiografias.	Pacientes: > 2.943 - treino (80%). > 718 - teste (19,61%). Exames - Radiografias: > 5.224 - treino (87,15%). > 770 - teste (12,85%).	**> AUC** - 0,786 (IC 95%: 0,745–0,830) **> PR AUC** - 0,517 (IC 95%: 0,429–0,600) para previsão de deterioração em 96 h. Desempenho comparável ao de dois radiologistas (com 3 e 17 anos de experiência) em um estudo de leitura.
		> **COVID-GBM** - *Gradient Goosting Mode* (GBM), denominado **.**	> Utiliza **conjunto de variáveis clínicas.**		> 13.416 conjuntos de recursos de sinais vitais - Frequência cardíaca, frequência respiratória, Temperatura, Pressão Arterial, Saturação de Oxigênio, Forneciment de Oxigênio Suplementar.	Conjuntos de Recursos Vitais: > 10.640 - treino (79,21%) > 2.776 - teste (20,69%)	
02 - Detecção automatizada de casos COVID-19 usando redes neurais profundas com imagens de raios-X (OZTURK, Tulin et al., 2020).	> **Diagnóstico**	**> DarkCovidNet** - DarkNet foi usado como um classificador para o sistema de detecção de objetos em tempo real You Only Look Once (YOLO).	> Utiliza **Raio X** > Gera **HeatMaps**.	> **Imagens COVID-19** - Dr. Cohen JP 2020 COVID-19 Image Data Collection > **Outras Imagens** - ChestX-ray8	> 1.125 Imagens >> 125 Covid. >> 500 Pneumonia. >> 500 Sem identificação .	> 80% - Treino. > 20% - Teste.	**> Precisão** >> 98,08% para classes binárias. >> 87,02 para três classes.
03 - COVIDX-Net: A Framework of Deep Learning Classifiers to Diagnose COVID-19 in X-Ray Images (HEMDAN, Ezz El-Din; SHOUMAN, Marwa A.; KARAR, Mohamed Esmail, 2020).	> **Diagnóstico**	> **VGG19**	> Utiliza **raio X**	> **Imagens** Dr. Cohen JP 2020 COVID-19 Image Data Collection. Dr. Adrian Rosebrock.	> 50 Imagens >> 25 Covid. >> 25 Normais.	> 80% - Treino. > 20% - Teste.	> **Acurácia** - 90%. > **Precisão** - 0,83 - Covid. - 1,00 - Normal
		> **DenseNet201**					> **Acurácia** - **90%.** > **Precisão** - 0,83 - Covid. - 1,00 -
		> **ResNetV2**					> **Acurácia** - 70%. > **Precisão** - 1,00 - Covid. - 0,62 -
		> **Incepction V3**					> **Acurácia** - 50%. > **Precisão** - 0,00 - Covid. - 0,50 -
		> **IncepctionResNetV2**					> **Acurácia** - 80%. > **Precisão** - 1,00 - Covid. - 0,71 -
		> **Xception**					> **Acurácia** - 80%. > **Precisão** - 1,00 - Covid. - 0,71 -
		> **MobileNetV2**					> **Acurácia** - 60% > **Precisão** - 1,00 - Covid. - 0,56 -

Fonte: Produzido pela autora do trabalho – dados oriundos de referências citadas na coluna “Trabalho Realizado”.

..... Para entendimento do Trabalho Realizado 01 do

Quadro 4 cabe verificar a Figura 53 em que são exibidas imagens de raio X originais com os respectivos mapas de saliência que destacam regiões com padrões visuais, como opacidades e consolidação, para demonstração da deterioração clínica em 24, 48, 72 e 96 horas. Nas imagens são demonstradas patches de seis Regiões de Interesse (ROI) com os escores por ordem de importância (OZTURK et al., 2020).

Figura 53 — Aplicação de COVID-GMIC

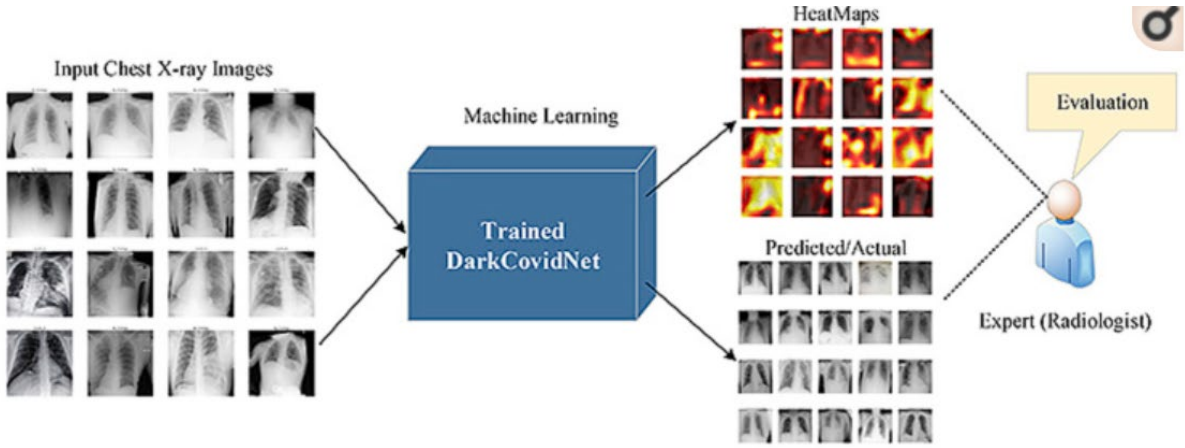


Fonte: Figura 3 reduzida da Explicação de COVID-GMIC (SHAMOUT et al, 2021).

.....A Figura 54 ilustra a implementação do Trabalho 02 do

Quadro 4 iniciando-se com radiografias de tórax, realizando-se o treinamento de Machine Learning via DarkCovidNet, gerando-se HeatMap e predições (OZTURK et al., 2020). O mapa de calor indica áreas importantes enfatizadas no raio pelo modelo (OZTURK et al., 2020).

Figura 54 — Aplicação de DarkCovidNet



Fonte: Resumo Gráfico da aplicação de DarkCovidNet (OZTURK et al., 2020).

7.2 Qualidade de previsões

Pela avaliação de Wynants et al. (2020), a qualidade dos relatórios das previsões realizadas é pobre, considerando-se que a maioria dos estudos tem sido construídas sob dados de uma localidade ou país específico, com alto risco de viés, de acordo com o que foi avaliado via o modelo de predição Risk of Bias Assessment ferramenta (PROBAST) (JIMENEZ-SOLEM et al., 2021). Dados locais ou nacionais, com alto risco de viés, conforme avaliado pelo modelo de predição Probast. Ademais, modelos frequentemente possuem o foco na admissão hospitalar, em dados de pacientes internados confirmados para COVID-19 e ignoram dados de pacientes com trajetória de doença mais leves (JIMENEZ-SOLEM et al., 2021).

Destaca-se que uma das iniciativas para o aumento da precisão de modelos de previsão, permitindo maior robustez e minimizando o potencial de sobreajuste, é a definição adequada da amostra, considerando-se o seu tamanho mínimo e a qualidade dos dados utilizados na representação da população-alvo (RILEY et al., 2019). A determinação de amostras apropriadas pressupõe procedimentos específicos (RILEY et al., 2019) e estratégias de compartilhamento de dados por pesquisadores (SPERRIN; GRANT; PEEK, 2020).

Apesar das questões que envolvem qualidade de previsões, radiologistas levam em geral 15 minutos na avaliação de imagens e indicação se são ou não compatíveis com COVID-19, enquanto software com algoritmos de IA, por exemplo o disponibilizado pela China, de forma gratuita, às instituições médicas ao redor do mundo, realiza diagnóstico em 15 segundos a partir de análise tomográfica de tórax, com uma taxa de precisão de 90% (KFOURI NETO; SILVA; NOGAROLI, 2020).

Dessa forma, cabe o contínuo investimento em IA aplicando-se Machine Learning e Deep Learning no diagnóstico e prognóstico da COVID-19, refinando parâmetros, formando bases de dados ajustadas e representativas com radiografias de pessoas de países diferentes, com quadros diferenciados da doença, de modo a aumentar a qualidade das previsões e aumentar a segurança do uso como ferramenta de trabalho pelas equipes médicas.

8 ARQUITETURA E IMPLEMENTAÇÃO

8.1 Base de dados

Atendendo-se ao que foi tratado nos tópicos 0 e 0, houve o levantamento de base de dados a ser utilizada no código de diagnóstico de COVID-19 com imagens de raios-X públicas disponíveis na Internet, sem identificação de pacientes, preservando-se, portanto, a "proteção dos dados pessoais, à tutela da privacidade e ao princípio da não discriminação, tocando as dimensões individual e coletiva" (MACHADO; SCHERTEL, 2020).

Inicialmente as imagens de raios-X recuperadas da Internet foram agrupadas em Normal, Covid e Pneumonia, considerando-se as referências Ozturk et al (2020) e Xu (2020), e utilizadas para treino e teste de modelos de Deep Learning em código de diagnóstico de COVID-19. Entretanto, após aprofundamento de pesquisa, na busca de uma base de dados com maior número de classes e de dados, foi adotada uma do Kaggle.

Cabe esclarecer que Kaggle é uma subsidiária da Google LLC, fundada em 2010, "uma comunidade online de cientistas de dados e profissionais de aprendizado de máquina"[1], que se propõe a resolver tarefas

[1] https://stringfixer.com/pt/Kaggle.

complexas e cognitivas a partir de diversos tipos de conhecimentos, estabelecendo-se como uma plataforma de predição e analytics (AL-TAIE; SALIM; OBASA, 2017). “Permite que usuários encontrem e publiquem conjuntos de dados, explorem e construam modelos em um ambiente de ciência de dados baseado na web” 2.

A base de dados escolhida no Kaggle foi o “Banco de Dados de raios-X de tórax COVID-19” (KAGGLE, 2021), elaborada a partir dos trabalhos de Rahman et al (2021) e Chowdhury et al (2020). O banco de dados criado por uma equipe de pesquisadores da Qatar University, Doha, Qatar, e Dhaka University, Bangladesh, em conjunto com pesquisadores do Paquistão e da Malásia, incluindo a colaboração de médicos (KAGGLE, 2021).

Essa base de dados é composta de arquivos Portable Network Graphics (PNG) com resolução de 299 por 299 pixels, de metadados e está dividida em 4 classes de radiografias, sendo 10.192 imagens de casos Normais, 3616 de Covid, 6.012 de Opacidade Pulmonar e 1345 de pneumonia viral (KAGGLE, 2021).

A base de dados foi construída a partir de imagens coletadas em países diferentes, fontes online e artigos, sem identificação dos pacientes (KAGGLE, 2021).

Da base de dados KAGGLE, foi utilizada apenas uma parte dos dados para este trabalho na maioria das execuções de algoritmos, tendo em vista a dificuldade da autora em executar códigos com os recursos computacionais a sua disposição.

Os dados foram separados em treino e teste, contendo 4 classes de imagens rotuladas/classificadas previamente como COVID-19 (classe 0), Lung Opacity (classe 1), Normal (classe 2) ou Viral Pneumonia (classe 3).

O detalhamento dessas execuções e do tamanho utilizado da base de dados está descrito na seção 0.

8.2 Linguagem e Ambiente

O código de diagnóstico de COVID-19 deste trabalho foi desenvolvido na linguagem Python 3.7, tendo em vista que é de alto nível, multiparadigma, multiplataforma, licença pública, adequada para Machine e Deep Learning, além do fato de ser de fácil utilização, considerando-se a forma como foi concebida e a diversidade de bibliotecas que possui.

O Python deste trabalho foi editado e executado no Google Colaboratory, mais conhecido como Google Colab, que é um serviço de nuvem criado e disponibilizado gratuitamente, associado a uma conta Google Drive, configurado com as principais bibliotecas de Inteligência Artificial conforme conceituações de Tiago Carneiro et al. (2018). O Google Colab pode ser configurado para uso em Graphics Processing Unit (GPU) e Tensor Processing Unit (TPU), além da Central Processing Units (CPU).

Em função do tempo de execução dos modelos testados, foi necessária a realização de upgrade do Google Colab para a opção paga Google Colab Pro com o fim de garantir ambiente de execução com maior duração, mais memória RAM e GPUs mais rápidas. Para isso foi realizado o investimento de R$ 58,00 mensais durante os meses de elaboração da monografia e respectivas execuções de códigos.

[2] Ibid.

9 TRANSFER LEARNING

Tendo em vista que o trabalho em questão se trata de reconhecimento de imagens, foram escolhidas arquiteturas de rede neural convolucional.

Para tornar mais eficiente e eficaz a aplicação de rede neural convolucional, optou-se, ainda, pela implementação de Transfer Learning, permitindo-se o processo de reaproveitamento de redes já treinadas, especializadas na identificação de imagens (objetos, pessoas, animais) e já otimizadas para uma variedade muito maior de discriminadores em outro conjunto de dados, que, no caso, foi o relacionado com raios-X.

Os Modelos de Deep Learning foram treinados tendo como base o Imagenet na implementação de Transfer Learning.

9.1 Frameworks Keras

Os códigos gerados neste trabalho utilizaram Keras como framewok, motivadamente escolhidos em função da pesquisa realizada no início de 2019 entre as equipes que terminaram nas 5 primeiras colocações de competições Kaggle, onde o Keras foi classificado como o número 1 em aprendizado profundo 3.

No site do Keras, no momento de acesso deste trabalho, foram listados como modelos disponíveis os relacionados na Tabela 24.

Tabela 24 — Aplicativos Keras

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0,79	0,945	22.910.480	126
VGG16	528 MB	0,713	0,901	138.357.544	23
VGG19	549 MB	0,713	0,9	143.667.240	26
ResNet50	98 MB	0,749	0,921	25.636.712	-
ResNet101	171 MB	0,764	0,928	44.707.176	-
ResNet152	232 MB	0,766	0,931	60.419.944	-
ResNet50V2	98 MB	0,76	0,93	25.613.800	-
ResNet101V2	171 MB	0,772	0,938	44.675.560	-
ResNet152V2	232 MB	0,78	0,942	60.380.648	-
InceptionV3	92 MB	0,779	0,937	23.851.784	159
InceptionResNetV2	215 MB	0,803	0,953	55.873.736	572
MobileNet	16 MB	0,704	0,895	4.253.864	88
MobileNetV2	14 MB	0,713	0,901	3.538.984	88
DenseNet121	33 MB	0,75	0,923	8.062.504	121
DenseNet169	57 MB	0,762	0,932	14.307.880	169
DenseNet201	80 MB	0,773	0,936	20.242.984	201
NASNetMobile	23 MB	0,744	0,919	5.326.716	-
NASNetLarge	343 MB	0,825	0,96	88.949.818	-
EfficientNetB0	29 MB	-	-	5.330.571	-
EfficientNetB1	31 MB	-	-	7.856.239	-
EfficientNetB2	36 MB	-	-	9.177.569	-
EfficientNetB3	48 MB	-	-	12.320.535	-
EfficientNetB4	75 MB	-	-	19.466.823	-
EfficientNetB5	118 MB	-	-	30.562.527	-
EfficientNetB6	166 MB	-	-	43.265.143	-
EfficientNetB7	256 MB	-	-	66.658.687	-

Fonte: Dados extraídos https://keras.io/api/applications/ com destaque colorido dado pela autora.

Considerando-se as vantagens de uso do Keras apresentadas na seção 0 e a acurácia apontada nas colunas “Top-1 — Accuracy” e “Top-5 — Accuracy” na Tabela 24 os modelos destacados com tarja verde foram os adotados na execução de códigos de diagnóstico de raios-X neste trabalho.

No Apêndice A há a Lista de parâmetros utilizados nos testes de cada frameworks aplicados neste trabalho 4.

[3] https://keras.io/why_keras/.
[4] Parâmetros adotados conforme recomendação de https://keras.io/api/applications/.

9.2 Código gerado

No código gerado em Python neste trabalho foram utilizados como referência programas produzidos em sala pelo Prof. MSc. Ricardo José Menezes Maia e no curso “Machine Learning para competições Kaggle — Especial COVID-19” 5.

10 EXECUÇÕES E DESAFIOS

10.1 Execuções

A partir das definições e escolhas dos tópicos antecessores neste capítulo, foram realizadas diversas execuções e encontrados desafios que demandaram ajustes e novas execuções de código de forma iterativa.

Os parâmetros adotados para todas as execuções:

Weights = ‘imagenet’;

Batch_size de teste = 1;

Optimizer = ‘Adam’;

Loss = 'sparse_categorical_crossentropy';

Activation = ‘softmax’;

Dropout = 0.2;

Classes = 4.

Para a análise dos resultados foram verificados os indicadores do classification_report e do histórico de épocas, além da matriz de confusão e o número de parâmetros do framework.

No classification-report a prioridade de escolha foi a partir das melhores predições das classes, priorizando o maior percentual de precisão da COVID-19 (classe 0), visto que o propósito deste trabalho é de diagnóstico de casos de COVID-19.

Do histórico de épocas foram utilizados os indicadores: “accuracy_score” (dados de teste) e acerto médio da ‘val_accuracy’ (dados de treino).

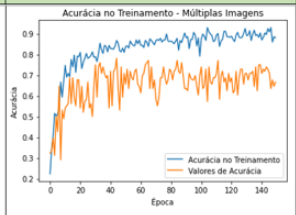
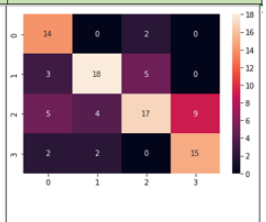
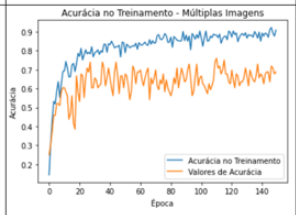
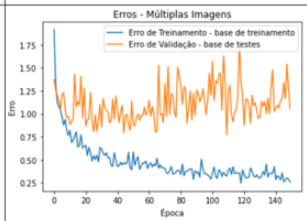
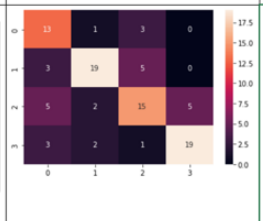
A escolha final foi pelo modelo de menor número de parâmetros após o ranking estabelecido a partir da análise do classification_report e histórico de épocas. Essa estratégia fundamenta-se pela importância de uma inferência eficiente e com o foco na TI Verde, buscando-se o menor uso de recursos possível, principalmente de tempo de execução e, consequentemente, de energia. A título de exemplo, um treinamento do zero de uma rede neural causa a emissão de CO2 similar à vida útil de cinco carros (CAI et al., 2019).

Com a finalidade de identificação dos parâmetros batch_size de treino e epochs e dos melhores frameworks a serem utilizados em bases de dados maiores, foi adotado um universo reduzido de 120 imagens por classe, distribuídos em 80% treino (96 radiografias) e 20% (24 radiografias) em teste.

[5] https://www.udemy.com/courses/search/?src=ukw&q=COVID-19+JONAS.

No Quadro 5 constam os resultados das execuções de códigos com framework MobiliNetV2 utilizados na definição do parâmetro batch_size de treino de 8, visto que a precisão de 0.58 da classe 0 – COVID-19 foi maior, apesar do accuracy-score de 0.6875 do batch_size de treino de 10 ter sido maior.

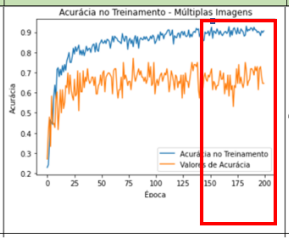
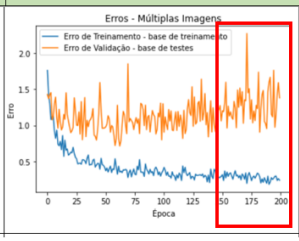
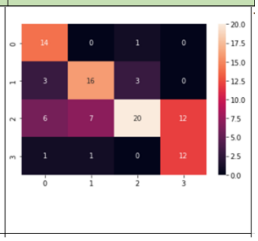
Quadro 5 — Definição do batch_size de treino

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> MobilenetV2 **Par. de Execução** lote_treino = **08** épocas = **150**	**> Acerto Médio** - 0.6555 **> Desvio Padrão** - 0.0911 **> Accuracy-Score** - 0.6666 **precision recall f1-score support** Classes 0 0.58 0.88 0.70 16 Classes 1 0.75 0.69 0.72 26 Classes 2 0.71 0.49 0.58 35 Classes 3 0.62 0.79 0.70 19 **accuracy** 0.67 96 macro avg 0.67 0.71 0.67 96 weighted avg 0.68 0.67 0.66 96			
> MobilenetV2 **Par. Execução** lote_treino = **10** épocas = **150**	**> Acerto Médio** - 0.6313 **> Desvio Padrão** - 0.0805 **> Accuracy-Score** - 0.6875 **precision recall f1-score support** Classes 0 0.54 0.76 0.63 17 Classes 1 0.79 0.70 0.75 27 Classes 2 0.62 0.56 0.59 27 Classes 3 0.79 0.76 0.78 25 **accuracy** 0.69 96 macro avg 0.69 0.70 0.69 96 weighted avg 0.70 0.69 0.69 96			

Fonte: Produzido pela autora.

Na sequência foi definido o número de épocas de 150, observando-se a estabilidade das curvas de acurácia e erros entre as épocas 150 e 200 no Quadro 6.

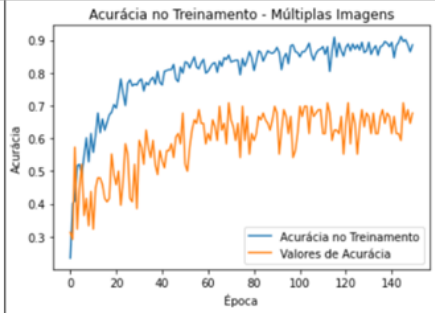
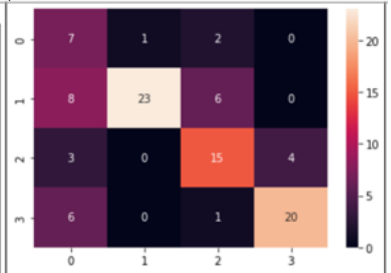
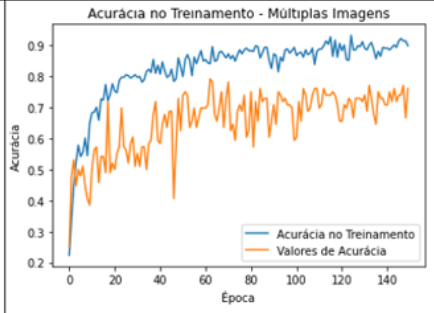
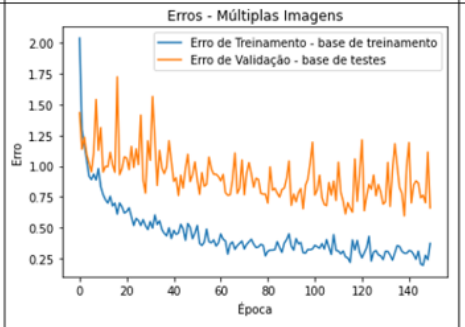
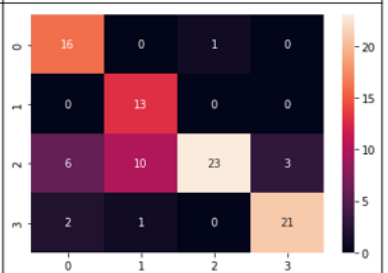
Quadro 6 — Definição de número de épocas

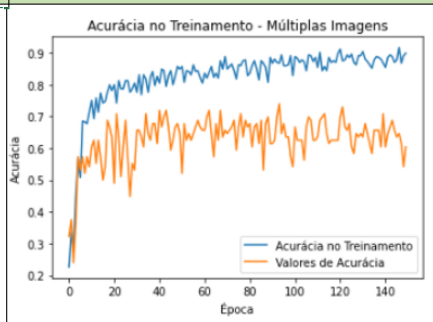
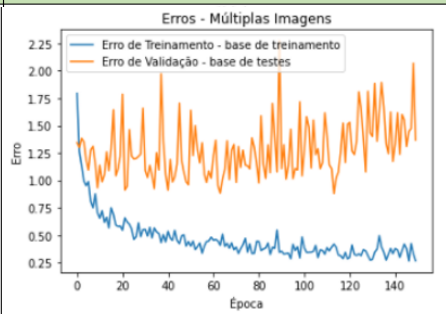
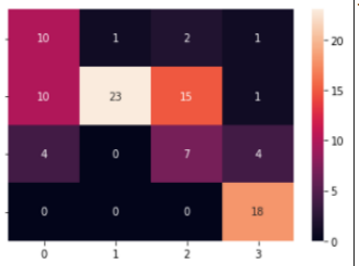
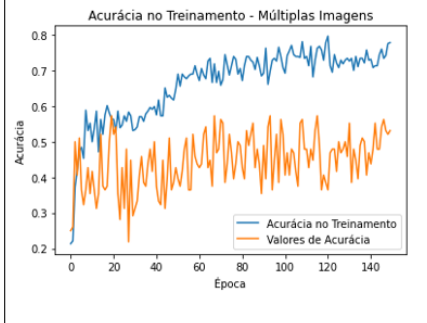
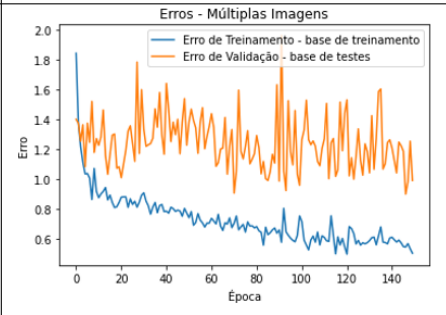
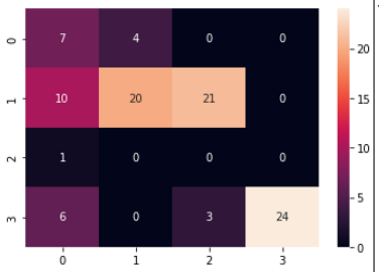
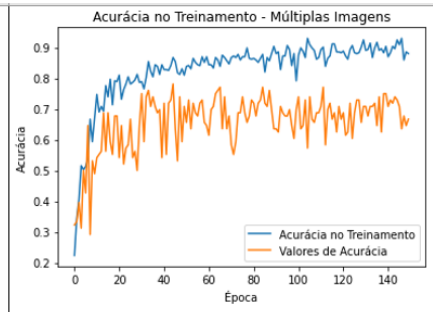
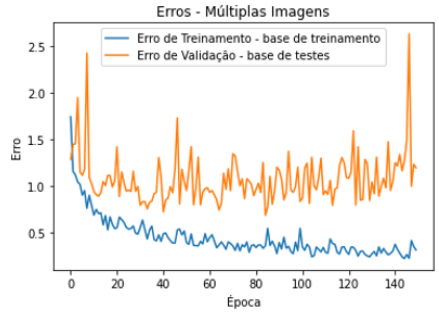
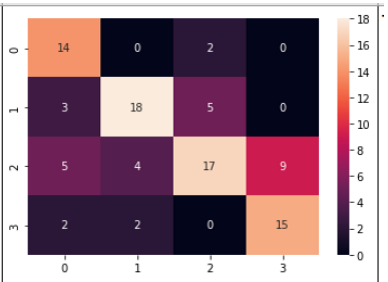
Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> MobilenetV2 **Par. de Execução** lote_treino = **08** épocas = **200**	**> Acerto Médio** - 0.6492 **> Desvio Padrão** - 0.0716 **> Accuracy-Score** - 0.6458 **precision recall f1-score support** Classes 0 0.58 0.93 0.72 15 Classes 1 0.67 0.73 0.70 22 Classes 2 0.83 0.44 0.58 45 Classes 3 0.50 0.86 0.63 14 **accuracy** 0.65 96 macro avg 0.65 0.74 0.66 96 weighted avg 0.71 0.65 0.64 96			

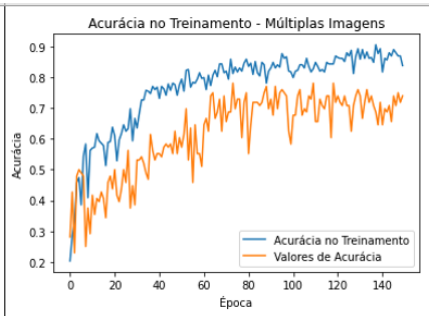
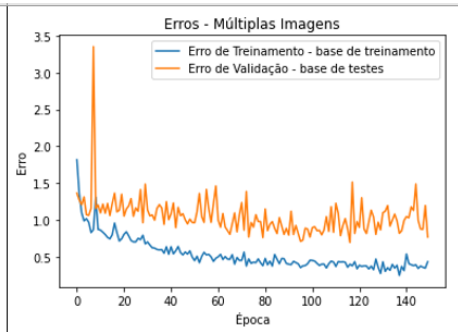
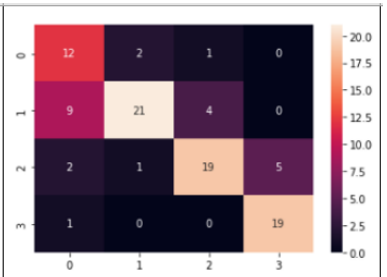
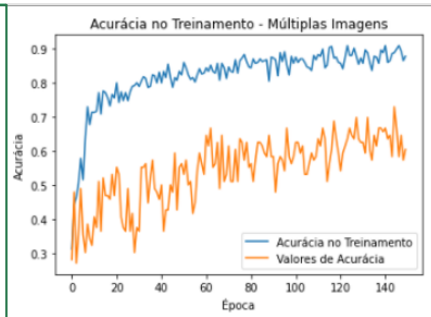
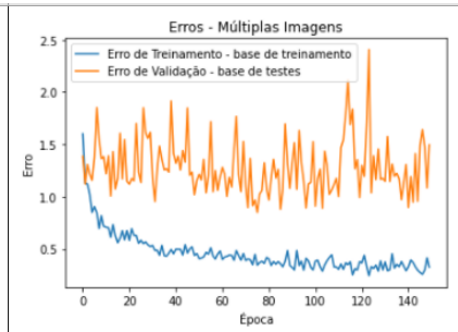
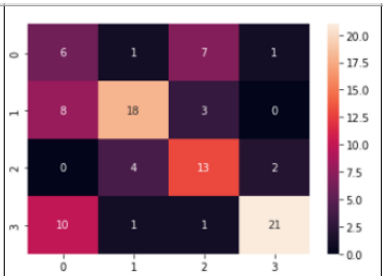
Fonte: Produzido pela autora.

Partindo-se dos parâmetros definidos, foram executados os outros frameworks ainda com a base de dados reduzida de 120 image

Quadro 7 — Execução de framework em base reduzida de 120 imagens por classe

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> Xception **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.5894 **> Desvio Padrão** - 0.0950 **> Accuracy-Score** - 0.6770 **precision recall f1-score support** Classes 0 0.29 0.70 0.41 10 Classes 1 0.96 0.62 0.75 37 Classes 2 0.62 0.68 0.65 22 Classes 3 0.83 0.74 0.78 27 **accuracy** 0.68 96 macro avg 0.68 0.69 0.65 96 weighted avg 0.78 0.68 0.70 96			
> Resnet50 **Parâmetros de Execução** lote_treino = **08** épocas = **150**	**> Acerto Médio** - 0.6548 **> Desvio Padrão** - 0.0979 **> Accuracy-Score** - 0.7604 **precision recall f1-score support** Classes 0 0.67 0.94 0.78 17 Classes 1 0.54 1.00 0.70 13 Classes 2 0.96 0.55 0.70 42 Classes 3 0.88 0.88 0.88 24 **accuracy** 0.76 96 macro avg 0.76 0.84 0.76 96 weighted avg 0.83 0.76 0.76 96			

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> Resnet152V2 Parâmetros de Execução lote_treino = **08** épocas = **150**	**> Acerto Médio** - 0.6278 **> Desvio Padrão** - 0.0732 **> Accuracy-Score** - 0.6041 **precision recall f1-score support** Classes 0 0.42 0.71 0.53 14 Classes 1 0.96 0.47 0.63 49 Classes 2 0.29 0.47 0.36 15 Classes 3 0.75 1.00 0.86 18 **accuracy** 0.60 96 macro avg 0.60 0.66 0.59 96 weighted avg 0.74 0.60 0.62 96			
>InceptionResNetV2 **Parâmetros de Execução** lote_treino = **08** épocas = **150**	**> Acerto Médio** - 0.4434 **> Desvio Padrão** - 0.0752 **> Accuracy-Score** - 0.5312 **precision recall f1-score support** Classes 0 0.29 0.64 0.40 11 Classes 1 0.83 0.39 0.53 51 Classes 2 0.00 0.00 0.00 1 Classes 3 1.00 0.73 0.84 33 **accuracy** 0.53 96 macro avg 0.53 0.44 0.44 96 weighted avg 0.82 0.53 0.62 96			
> MobilenetV2 **Par. de Execução** lote_treino = **08** épocas = **150**	**> Acerto Médio** - 0.6555 **> Desvio Padrão** - 0.0911 **> Accuracy-Score** - 0.6666 **precision recall f1-score support** Classes 0 0.58 0.88 0.70 16 Classes 1 0.75 0.69 0.72 26 Classes 2 0.71 0.49 0.58 35 Classes 3 0.62 0.79 0.70 19 **accuracy** 0.67 96 macro avg 0.67 0.71 0.67 96 weighted avg 0.68 0.67 0.66 96			

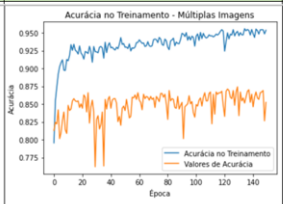
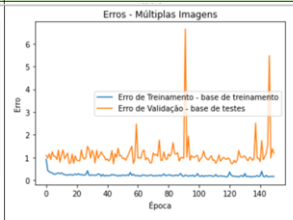
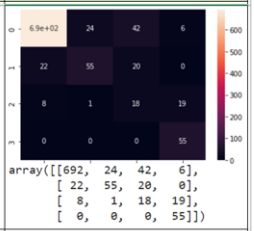
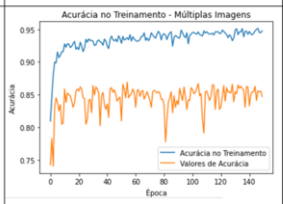
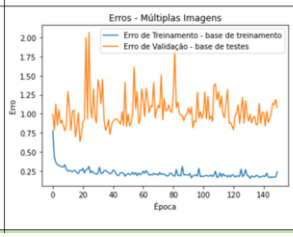
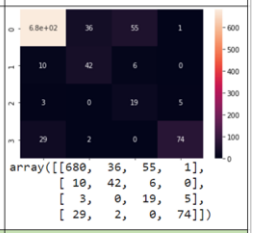
Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> DenseNet201 **Par. de Execução** lote_treino = **8** épocas = **150**	> **Acerto Médio** - 0.6184 > **Desvio Padrão** - 0.1286 > **Accuracy-Score** - 0.7395 **precision recall f1-score support** Classes 0 0.50 0.80 0.62 15 Classes 1 0.88 0.62 0.72 34 Classes 2 0.79 0.70 0.75 27 Classes 3 0.79 0.95 0.86 20 **accuracy** 0.74 96 macro avg 0.74 0.77 0.74 96 weighted avg 0.78 0.74 0.74 96			
> NASNetLarge **Parâmetros de Execução** lote_treino = **8** épocas = **150**	> **Acerto Médio** - 0.5437 > **Desvio Padrão** - 0.0975 > **Accuracy-Score** - 0.6041 **precision recall f1-score support** Classes 0 0.25 0.40 0.31 15 Classes 1 0.75 0.62 0.68 29 Classes 2 0.54 0.68 0.60 19 Classes 3 0.88 0.64 0.74 33 **accuracy** 0.60 96 macro avg 0.60 0.59 0.58 96 weighted avg 0.67 0.60 0.63 96			

Fonte: Produzido pela autora.

Os percentuais de precisão da COVID-19 (classe 0) apurados nos modelos executados não foram bons, inclusive em relação às outras classes, como pode ser constatado no Quadro 7. A classe Viral Pneumonia (3) demonstrou bons resultados nas predições.

Dessa forma, um novo Dataset foi criado com uma quantidade maior de dados para a classe COVID-19 (0), 3.614 imagens, e o número de 400 imagens para as outras categorias, com a mesma proporção de 80% de dados para treino e 20% para testes. Houve uma mudança significativa na precisão da classe COVID-19 (0), Quadro 8.

Quadro 8 — Execução de framework em base com número maior de imagens para COVID-19

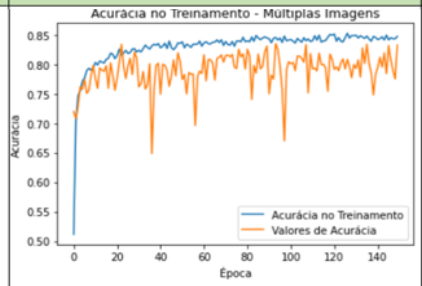
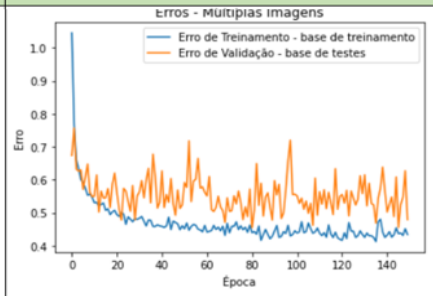
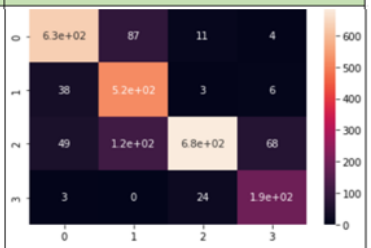
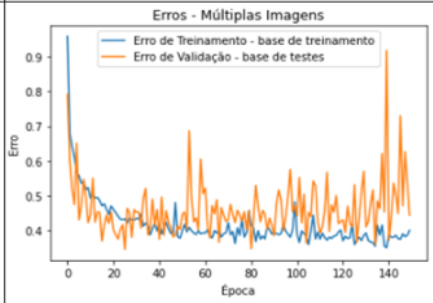
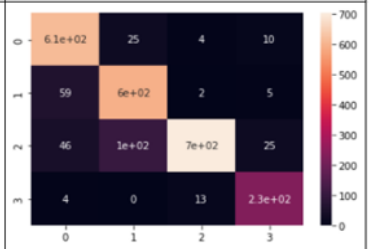
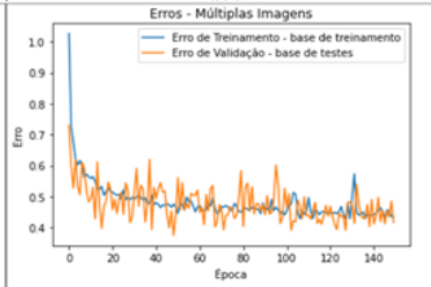
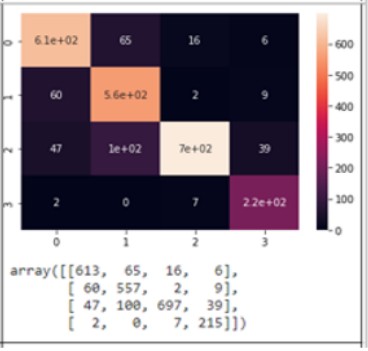
Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> Resnet50 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.8488 **> Desvio Padrão** - 0.0182 **> Accuracy-Score** - 0.8523 **precision recall f1-score support** Classes 0 0.96 0.91 0.93 764 Classes 1 0.69 0.57 0.62 97 Classes 2 0.23 0.39 0.29 46 Classes 3 0.69 1.00 0.81 55 **accuracy** 0.85 962 macro avg 0.64 0.72 0.66 962 weighted avg 0.88 0.85 0.86 962			
> MobileNetV2 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.8439 **> Desvio Padrão** - 0.0204 **> Accuracy-Score** - 0.8471 **precision recall f1-score support** Classes 0 0.94 0.88 0.91 772 Classes 1 0.53 0.72 0.61 58 Classes 2 0.24 0.70 0.36 27 Classes 3 0.93 0.70 0.80 105 **accuracy** 0.85 962 macro avg 0.66 0.75 0.67 962 weighted avg 0.90 0.85 0.86 962			

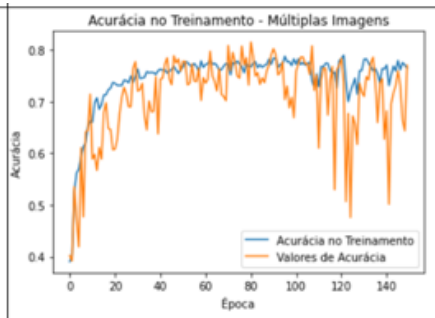
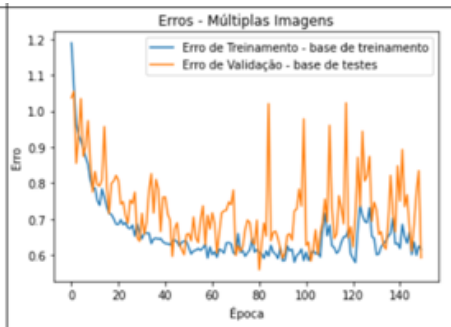
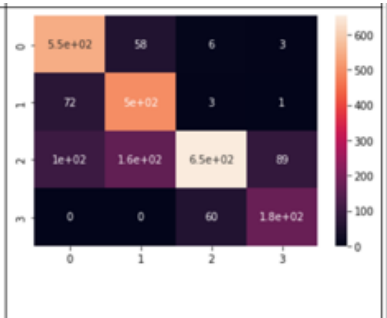
Fonte: Produzido pela autora.

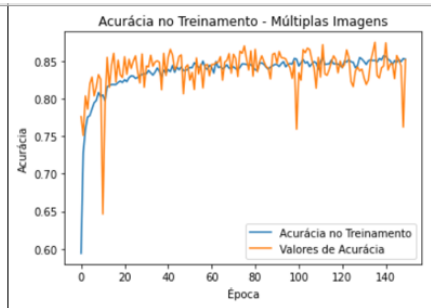
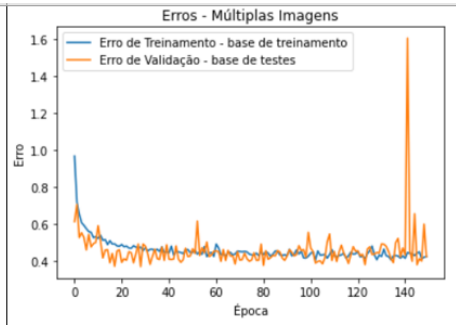
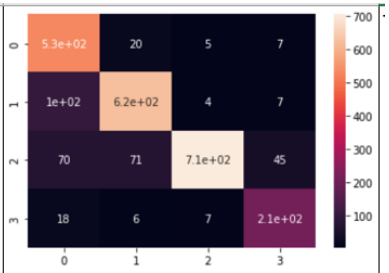
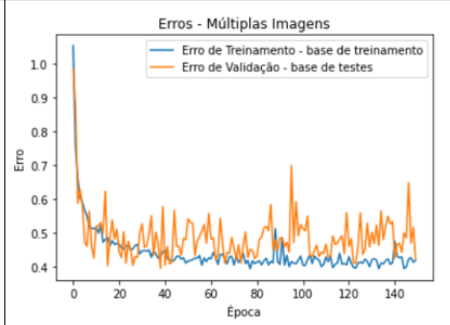
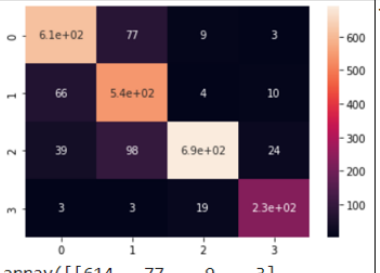
Com a mudança apresentada de comportamento no Quadro 8 ficou evidenciado que o balanceamento das informações é ação essencial na execução bem-sucedida dos modelos. Assim, uma nova tentativa de criação de um Dataset mais adequado para treino foi realizada, tendo como ponto de partida 3.614 imagens, 99,94%, disponíveis na base Kaggle adotada neste trabalho, para a classe COVID-19 (0). Esse mesmo percentual foi utilizado para as classes Lung Opacity (1) e Normal (2), apesar de estarem disponíveis quantitativos maiores de imagens na base Kaggle. Para a classe Viral Pneumonia (3) foram utilizadas 1343 imagens, 99,85%, da base Kaggle.

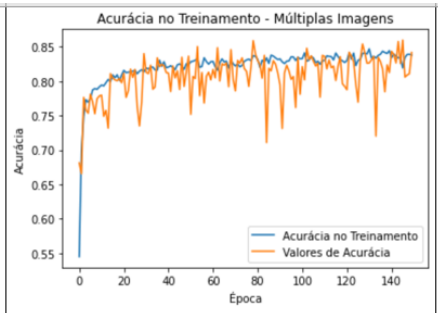
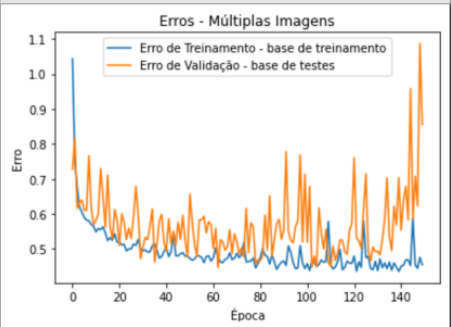
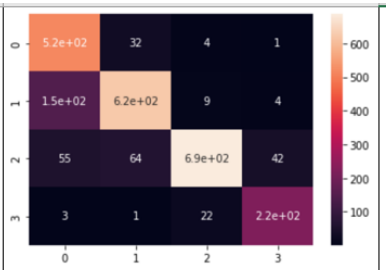
Todos os modelos foram executados a partir dessa nova base de dados e os resultados estão apresentados no Quadro 9.

Quadro 9 — Execução de framework em base com 1.343 imagens para V. Pneumonia e 3.614 para as outras classes

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> Xception **Parâmetros de Execução** lote_treino = **8** épocas = **150**	> **Acerto Médio** - 0.7899 > **Desvio Padrão** - 0.0291 > **Accuracy-Score** - 0.8324 **precision recall f1-score support** Classes 0 0.88 0.86 0.87 734 Classes 1 0.72 0.92 0.81 567 Classes 2 0.95 0.75 0.84 916 Classes 3 0.71 0.88 0.78 218 accuracy 0.83 2435 macro avg 0.81 0.85 0.82 2435 weighted avg 0.85 0.83 0.83 2435			
> ResNet50 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	> **Acerto Médio** - 0.8451 > **Desvio Padrão** - 0.0242 > **Accuracy-Score** - 0.8788 **precision recall f1-score support** Classes 0 0.85 0.94 0.89 652 Classes 1 0.82 0.90 0.86 661 Classes 2 0.97 0.80 0.88 876 Classes 3 0.85 0.93 0.89 246 accuracy 0.88 2435 macro avg 0.87 0.89 0.88 2435 weighted avg 0.89 0.88 0.88 2435			
> ResNet152V2 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	> **Acerto Médio** - 0.8219 > **Desvio Padrão** - 0.0249 > **Accuracy-Score** - 0.8550 **precision recall f1-score support** Classes 0 0.85 0.88 0.86 700 Classes 1 0.77 0.89 0.83 628 Classes 2 0.97 0.79 0.87 883 Classes 3 0.80 0.96 0.87 224 accuracy 0.86 2435 macro avg 0.85 0.88 0.86 2435 weighted avg 0.87 0.86 0.86 2435			

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> InceptionResNetV2 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.7083 **> Desvio Padrão** - 0.0835 **> Accuracy-Score** - 0.7700 **precision recall f1-score support** Classes 0 0.76 0.89 0.82 614 Classes 1 0.69 0.87 0.77 575 Classes 2 0.90 0.65 0.75 1010 Classes 3 0.65 0.75 0.70 236 accuracy 0.77 2435 macro avg 0.75 0.79 0.76 2435 weighted avg 0.79 0.77 0.77 2435			

> MobilenetV2 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.8389 **> Desvio Padrão** - 0.0261 **> Accuracy-Score** - 0.8517 **precision recall f1-score support** Classes 0 0.74 0.94 0.83 565 Classes 1 0.87 0.85 0.86 737 Classes 2 0.98 0.79 0.87 892 Classes 3 0.78 0.87 0.82 241 accuracy 0.85 2435 macro avg 0.84 0.86 0.85 2435 weighted avg 0.87 0.85 0.85 2435			 array([[533, 20, 5, 7], [101, 625, 4, 7], [70, 71, 706, 45], [18, 6, 7, 210]])
> DenseNet201 **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.8249 **> Desvio Padrão** - 0.0390 **> Accuracy-Score** - 0.8542 **precision recall f1-score support** Classes 0 0.85 0.87 0.86 703 Classes 1 0.75 0.87 0.81 624 Classes 2 0.96 0.81 0.88 851 Classes 3 0.86 0.90 0.88 257 accuracy 0.85 2435 macro avg 0.86 0.86 0.86 2435 weighted avg 0.86 0.85 0.86 2435			 array([[614, 77, 9, 3], [66, 544, 4, 10], [39, 98, 690, 24], [3, 3, 19, 232]])

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> NASNetLarge **Parâmetros de Execução** lote_treino = **8** épocas = **150**	**> Acerto Médio** - 0.8069 **> Desvio Padrão** - 0.0324 **> Accuracy-Score** - 0.8418 **precision recall f1-score support** Classes 0 0.71 0.93 0.81 553 Classes 1 0.87 0.80 0.83 786 Classes 2 0.95 0.81 0.88 848 Classes 3 0.83 0.90 0.86 248 accuracy 0.84 2435 macro avg 0.84 0.86 0.84 2435 weighted avg 0.86 0.84 0.84 2435			

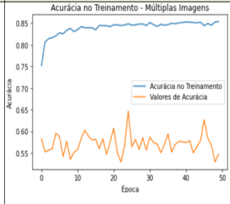
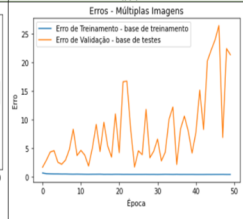
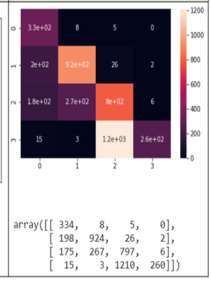
Fonte: Produzido pela autora.

As execuções representadas no Quadro 9 mostraram resultados bem melhores, tanto em termos de Accuracy-Score bem como da precisão da classe COVID-19 (0).

Outra e última execução realizada foi com o universo quase completo Kaggle com 3.614 imagens para a classe COVID-19 (0), 6.010 para Lung Opacity (1), 10.190 para Normal (2) e 1.343 para Viral Pneumonia (3), perfazendo um total de 21.157 imagens.

No Quadro 10 são exibidos os resultados obtidos que não foram bons, pois o Accuracy-Score foi baixo e a precisão da predição da classe COVID-19 (0) ficou longe de 70%.

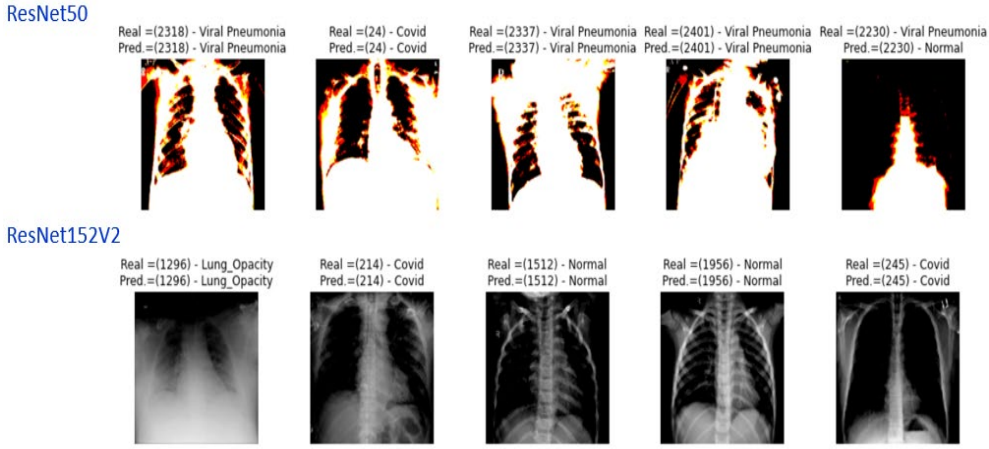
Quadro 10 — Execução de framework em base completa Kaggle

Arquitetura	Resultado	Acurácia no Treinamento	Erros	Matriz de Confusão
> NASNetLarge **Parâmetros de Execução** lote_treino = **10** épocas = **50**	> **Acerto Médio** - 0.5712 > **Desvio Padrão** - 0.0222 > **Accuracy-Score** - 0.5472 **precision recall f1-score support** Classes 0 0.46 0.96 0.62 347 Classes 1 0.77 0.80 0.79 1150 Classes 2 0.39 0.64 0.49 1245 Classes 3 0.97 0.17 0.30 1488 **accuracy** 0.55 4230 macro avg 0.65 0.65 0.55 4230 weighted avg 0.70 0.55 0.51 4230			

Fonte: Produzido pela autora.

Na Figura 55 são exibidas algumas imagens com a classificação real e as predições correspondentes, considerando-se os frameworks ResNet50 e ResNet152V2.

Figura 55 — Classificações reais e predições



Fonte: Produzida pela autora.

Como parte das entregas realizadas, os pesos foram gravados por modelo executado em cada Dataset criado. Essas gravações permitirão a aplicação dos pesos em outros conjuntos de dados, fomentando novas pesquisas e economizando tempo de execução.

Na Figura 56 há exemplos dessa gravação.

Figura 56 — Modelos gravados na execução de algoritmos

Fonte: Produzida pela autora

10.2 Desafios

Os recursos para execução dos frameworks foram provavelmente os maiores desafios na aplicação de algoritmos de CNN em raio-X, motivo, inclusive, pelo qual

as quantidades de imagens nos testes foram reduzidas e o Colab PRO foi contratado. Inicialmente o Colab gratuito foi utilizado, mas as execuções demoravam muito e eram constantemente interrompidas. Com o uso do Colab Pro este problema foi resolvido por um bom tempo. Entretanto, com o grande volume de demoradas e rotineiras execuções, o Colab Pro começou a perguntar se elas eram originadas de um robô e, na ausência de interação e negativa por parte desta autora, eram sumariamente interrompidas, demandando reexecuções.

Mesmo com o Colab Pro, não foi possível aumentar o tamanho da base de dados de treino e teste. As bases reduzidas de 120 imagens por classe, demandaram de 15 a 20 minutos de execução, as de 400 imagens, horas e as ampliadas, de 08 à 12 horas, conforme o framework e a completa, mais de 24 horas.

Uma dificuldade não sanada foi a obtenção de radiografias de brasileiros, pois a base pública encontrada, do Ministério da Saúde, é de acesso reservado a instituições e profissionais da saúde.

O balanceamento da base de dados é outro fator que impactou as predições e este problema é citado por vários autores sendo tema de doutorado reconhecidamente premiado sobre o assunto (PEREIRA; COSTA; SILLA, 2021) sendo constatado neste trabalho.

Este problema afetou o trabalho e demandou várias execuções e buscas da composição ideal da base de dados. As tentativas para sanar o problema, como já mencionado neste tópico, inicialmente foram no sentido de igualar o número de imagens das classes no treinamento dos modelos. Entretanto, esta tática não foi bem-sucedida e os resultados das predições dos modelos não foram bons.

Posteriormente, a opção adotada foi de aumento gradual da quantidade de imagens classe a classe com a correspondente verificação da precisão obtida. Com esta prática, observou-se que mesmo com números baixos de imagens de raios-X de “Pneumonia Viral”, a precisão da predição desta classe foi boa. A quantidade de imagens das classes “COVID-19” precisou ser maior para a correta classificação em tempo de treinamento e teste dos modelos.

11 APRESENTAÇÃO DOS RESULTADOS

O comportamento em todos os modelos aplicados foi de acertos importantes para Pneumonia Viral. Entretanto, por vezes os modelos identificaram COVID-19 como opacidade pulmonar ou vice-versa. Esse fato ocorreu tendo em vista que no quadro pulmonar da COVID-19 são encontradas opacidades multifocais.

O volume de dados e o balanceamento das classes foi fundamental para a melhoria dos resultados.

Tendo em vista registros dos resultados realizados no tópico 0 e dos desafios relatados no tópico 0, constatou-se que o melhor Dataset foi o de base ampliada com a composição de 3.614 para as classes, Covid-19 (0), Lung Opacity (1) e Normal (2) e de 1343 imagens para Viral Pneumonia (3).

Os parâmetros adotados foram:

Classes = 4;

Weights = ‘imagenet’;

Batch_size de treino = 08;

Batch_size de teste = 1;

Optimizer = ‘Adam’;

Loss = 'sparse_categorical_crossentropy';

Activation = ‘softmax’;

Dropout = 0.2.

O resultado geral para esse ambiente de execução está demonstrado no

Tabela 25.

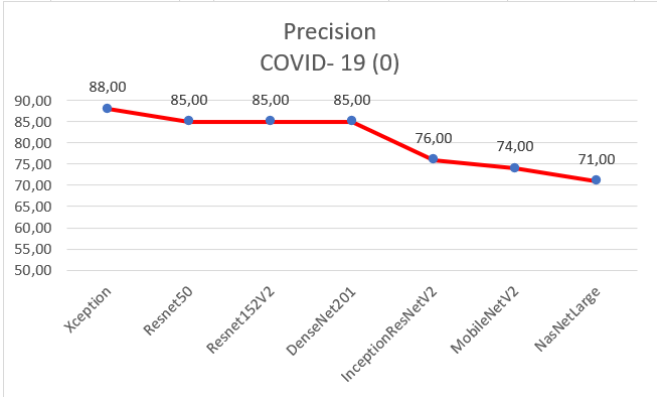
Tabela 25 — Resultado Geral

Frameworks	Acerto Médio	Desvio Padrão	Accuracy-Store	Precision COVID- 19 (0)	Precision L, Opacity (1)	Precision Normal (2)	Precision V, Pneum, (3)	Parâmetros *Framework*
Xception	0,79	0,0291	0,8324	0,88	0,72	0,95	0,71	22.910.480
Resnet50	0,845	0,0242	0,8788	0,85	0,82	0,97	0,85	25.636.712
Resnet152V2	0,822	0,0249	0,855	0,85	0,77	0,89	0,83	60.380.648
InceptionResNetV2	0,708	0,0835	0,77	0,76	0,69	0,9	0,65	55.873.736
MobileNetV2	0,839	0,0261	0,8517	0,74	0,87	0,98	0,78	3.568.894
DenseNet201	0,825	0,039	0,8542	0,85	0,75	0,96	0,86	20.242.984
NasNetLarge	0,807	0,0324	0,8418	0,71	0,87	0,95	0,83	88.949.818

Fonte: Produzido pela autora.

O framework de melhor resultado de predição para a classe Covid-19 (0) foi o Xception (88%) e o pior, NasNetLarge (71%), conforme Gráfico 12.

Gráfico 12 — Precisão Classe COVID-19 (0)



Fonte: Produzido pela autora.

Quanto aos quesitos accuracy-score e acerto médio, os melhores resultados são do algoritmo ResNet50 (87,88%; 84,51%) e os piores do InceptionResNetV2 (77%; 70,83%), conforme Gráfico 13 e Gráfico 14.

Gráfico 13 — Accuracy-Score

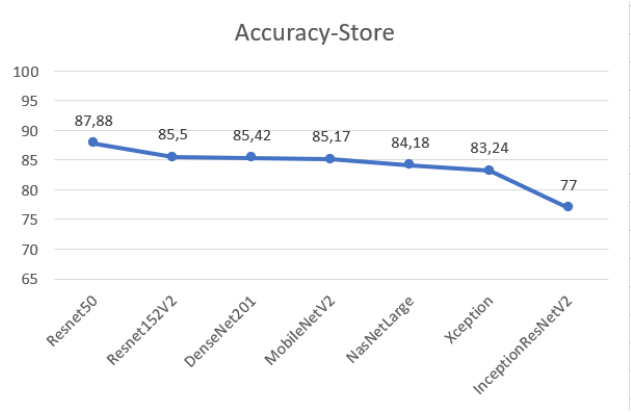
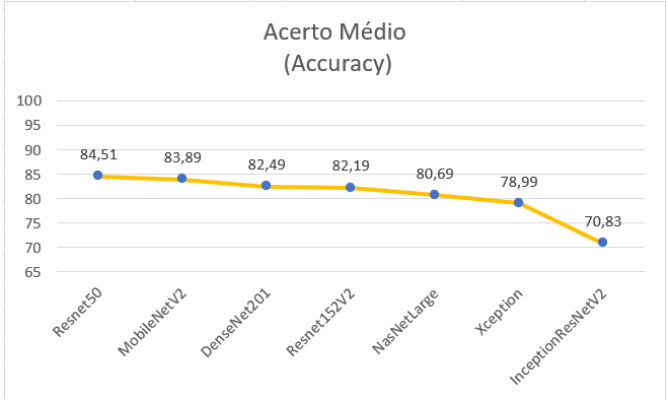


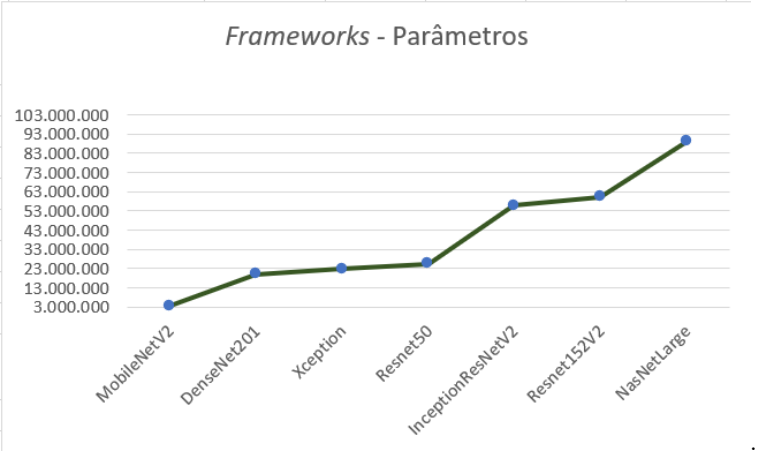
Gráfico 14 — Acerto Médio



Fonte: Produzidos pela autora.

Entre os frameworks escolhidos para testes nesta monografia, o de menor número de parâmetros foi o MobiNetV2, Gráfico 15. Esse algoritmo obteve bons resultados para accuracy-score (85,17%), Gráfico 13, e acerto médio (83,89%), Gráfico 14, mas o penúltimo pior percentual de precisão para a classe Covid-19 (74%),Gráfico 12.

Gráfico 15 — Frameworks por Parâmetros



Fonte: Produzido pela autora.

O número de parâmetros dos frameworks Xception (22.910.480) e ResNet50 (25.613.800) são reduzidos em relação aos outros escolhidos para este trabalho, apesar de não serem os de menor valor.

Dessa forma, em primeiro lugar foi escolhido o algoritmo Xception, em função da precisão obtida para a classe COVID-19, e, em segundo, o ResNet50, por apresentar os melhores resultados de accuracy-score e acerto médio, tendo os dois frameworks um número reduzido de parâmetros.

12 CONCLUSÃO

Neste trabalho foram atingidos os objetivos inicialmente propostos bem como foram respondidas as questões propostas.

Após fundamentação teórica, no capítulo 0 foram tratados Modelos de diagnósticos e prognósticos de COVID-19, sendo que no tópico 0 foram destacadas as aplicações de modelos de IA no diagnóstico em raios-X de tórax, respondendo a questão do problema inicial sobre as opções de IA que podem ser utilizadas na identificação automática e diagnósticos de COVID-19.

Dos frameworks elencados no site do Keras, foram testados os de melhor desempenho, ou seja, Xception, ResNet50, ResNet152V2, InceptionResNetV2, MobileNetV2, DenseNet121 e NasNetLarge na busca de um algoritmo adequado no

diagnóstico de COVID-19 através de radiografias de tórax, atendendo ao proposto neste trabalho.

Várias execuções foram realizadas, com maneiras de balanceamento e recortes diferentes do universo de dados disponibilizado na base Kaggle adotada, e os resultados obtidos foram apresentados no capítulo 0, no tópico 0, após detalhamento da arquitetura utilizada.

Os pesos obtidos conforme parâmetros e frameworks adotados nas execuções dos códigos foram gravados para posterior uso em outros Datasets e aplicação em trabalhos futuros.

A análise dos resultados dos algoritmos considerou com destaque os indicadores de maior precisão da classe COVID-19, accuracy-score, acerto médio, em função da busca do melhor desempenho, e os frameworks de menor número de parâmetros utilizados, com o foco no menor uso de recursos possível.

A apresentação da análise final dos resultados foi objeto do capítulo 0, em que foram indicados o algoritmo Xception como primeiro escolhido, melhor precisão para a classe COVID-19, e, em segundo, o ResNet50, maiores percentuais de accuracy-score e acerto médio, com número reduzido de parâmetros.

Com esses procedimentos foi possível atender o objetivo geral sobre levantamento de diferentes usos de IA na classificação automática de imagens de raios-X de tórax, com a correspondente escolha de algoritmos a serem executados e a proposição de solução a ser ofertada aos intervenientes que lidam com o COVID-19.

Ademais, também foram atendidos os objetivos específicos relacionados, com o levantamento do estado da arte de modelos de Machine Learning e Deep Learning no diagnóstico e prognóstico de COVID-19 em raios-X de tórax, escolha de arquiteturas de CNN para testes, escolha de base de dados e balanceamento de classes destinadas ao diagnóstico de COVID-19, teste de arquiteturas escolhidas, análise dos resultados encontrados e propostas de trabalhos futuros.

Pelos relatos apresentados neste trabalho, conclui-se que todo o investimento possível deve ser realizado na oferta de soluções que sejam aliadas às diversas

equipes envolvidas em diagnóstico, prognóstico do COVID-19, no planejamento estratégico, tático e operacional dos países, conscientização da população e avaliação demográfica.

No entanto, destaca-se que todo o ferramental de Inteligência artificial de suporte à detecção e tratamento dessa doença ou de qualquer outra não pode substituir equipes médicas, responsáveis profissionais pelo acompanhamento clínico de pacientes e pela decisão do tratamento e da condução de procedimentos, protegendo, inclusive o indivíduo de erros gerados por imprecisão, mesmo que residual, na construção e execução de algoritmos de IA (KFOURI NETO; SILVA; NOGAROLI, 2020).

Cabe ainda lembrar que cuidados devem ser adotados com referência à proteção do cidadão, sua vida e privacidade, inclusive com os devidos respaldos legais, com o propósito de evitar-se discriminações de quaisquer tipos, inclusive de seleção de trabalho, de recebimento de tratamento, tendo em vista as chances de evoluções positivas a partir de diagnósticos e prognósticos resultantes da aplicação de modelos de Machine Learning e Deep Learning, exposições desnecessárias e todos os tipos de abusos que possam surgir em nome da proteção da sociedade e de fins econômicos.

A vida e a integridade das pessoas em todas as dimensões possíveis devem ser o principal foco de qualquer estudo a ser realizado, principalmente o da Inteligência artificial que tem o condão de mostrar problemas e situações em escala que não seriam facilmente detectáveis com procedimentos convencionais.

12.1 Proposta de trabalhos futuros

As propostas de trabalhos futuros podem ser divididas em cinco grupos a serem correlacionados: associação de outros preditores à identificação de doenças pulmonares através de raio-X de tórax; uma base de dados robusta; oferta de serviços de diagnóstico e prognóstico para pacientes e equipes médicas; ética, privacidade e segurança no uso de informações médicas; otimização de algoritmos de IA.

O primeiro grupo diz respeito à associação do código de análise de raios-X com outros preditores como tosse e os citados deste trabalho, ampliando as perspectivas de diagnóstico e prognóstico, permitindo a previsão de possível evolução do quadro clínico e de futuras necessidades de recursos hospitalares e tratamentos, de tempos envolvidos, conforme, inclusive, histórico médico do paciente.

O segundo grupo está relacionado à formação de base de dados robusta com incorporação de classes para outros tipos de doenças pulmonares, aumento do volume de dados e inclusão de dados de brasileiros.

No terceiro grupo, o interesse de estudo é a oferta de serviços em dispositivos diversos tais como, celulares e relógios, que permitam o diagnóstico e prognóstico rápido de pacientes e a indicação que uma determinada pessoa deve procurar ajuda médica.

O tema do quarto grupo é a segurança, a ética e a privacidade no uso de dados médicos de pacientes. Para unir privacidade e aprendizado de máquina há pesquisas sobre privacy-preserving machine learning objeto de Doutorado do Orientador deste trabalho que deve ser explorado, sobretudo no Brasil tendo em vista a LGPD.

Por fim, o foco do quinto grupo é a otimização da IA considerando-se a melhoria de algoritmos, o refino de parâmetros, as formas de treinamento e a alocação de recursos, tendo em vista a necessidade de respostas rápidas, de aumento da efetividade das predições e de redução de custos computacionais, além da multiplicidade de dispositivos existentes. Neste grupo o foco também deve ser a inferência eficiente, com redução de recursos, principalmente tempo e energia. Acrescentando-se que o uso de técnicas privacy-preserving machine learning como privacidade diferencial, computação multipartidária segura e criptografia homomórfica exigem mais recursos computacionais. Dessa forma, estudos, desenvolvimento e aplicação de aprendizagem de máquina com a diretriz de “TI Verde” tornam-se urgentes.

REFERÊNCIAS

ALBAHRI, A. S. et al. Role of biological data mining and machine learning techniques in detecting and diagnosing the novel coronavirus (COVID-19): A systematic review. Journal of medical systems, v. 44, p. 1-11, 2020. Disponível em: <https://link.springer.com/content/pdf/10.1007/s10916-020-01582-x.pdf>. Aceso em: 17 ago. 2021.

AL-TAIE, M. Z.; SALIM, N.; OBASA, A. I. Successful Data Science Projects: Lessons Learned from Kaggle Competition. Kurdistan Journal of Applied Research, v. 2, n. 3, p. 40-49, 2017. Disponível em: <Successful Data Science Projects: Lessons Learned from Kaggle Competition | Kurdistan Journal of Applied Research (spu.edu.iq) >. Acesso em: 14 nov. 2021.

ASLAN, M. F. et al. CNN-based transfer learning–BiLSTM network: A novel approach for COVID-19 infection detection. Applied Soft Computing, v. 98, p. 106912, 2021. Disponível em: < https://www.sciencedirect.com/science/article/pii/S1568494620308504>. Acesso em: 17 abr. 2021.

BERNHEIM, A. et al. Chest CT findings in coronavirus disease-19 (COVID-19): Relationship to duration of infection. Radiology, v. 295, n.3, p. 200463, 2020. Disponível em: <https://pubs.rsna.org/doi/full/10.1148/radiol.2020200463:>. Acesso em: 17 out. 2021.

BEZERRA, E. Introdução à aprendizagem profunda. Anais do do 1º Simpósio Brasileiro de Banco de Dados–SBBD2016–Salvador, 2016. Disponível em: <https://www.researchgate.net/profile/Eduardo-Bezerra/publication/309321510_Introducao_a_Aprendizagem_Profunda/links/5809ee1108ae3a04d624f369/Introducao-a-Aprendizagem-Profunda.pdf>. Acesso em: 19 set. 2021.

BOUZON, M. F. Estudo de algoritmos de otimização inspirados na natureza aplicados ao treinamento de redes neurais artificiais. 2021.117f. Dissertação (Mestrado em Engenharia Elétrica) – Centro Universitário FEI, São Bernardo do Campo, 2021. Disponível em: <https://repositorio.fei.edu.br/bitstream/FEI/3214/1/fulltext.pdf>. Acesso em: 03 set. 2021.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Brasília, 15 ago. 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 18 abr. 2020.

CAI, H. al. Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791, 2019. Disponível em: <https://arxiv.org/abs/1908.09791>. Acesso em: 02 nov. 2021.

CARNEIRO, T. et al. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. IEEE Access, v. 6, p. 61677-61685, 2018. Disponível em: < https://ieeexplore.ieee.org/abstract/document/8485684>. Acesso em: 05 jun, 2021.

CASTRO, B. L. G. de et al. COVID-19 e organizações: Estratégias de enfrentamento para redução de impactos. Revista Psicologia Organizações e Trabalho, v. 20, n. 3, p. 1059-1063, 2020.Disponível em: <http://pepsic.bvsalud.org/scielo.php?script=sci_abstract&pid=S1984-66572020000300002&lng=es&nrm=iso&tlng=pt.>. Acesso em:04 set, 2021.

CHOWDHURY, M. E. et al. Can AI help in screening viral and COVID-19 pneumonia? IEEE Access, v. 8, p. 132665-132676, 2020. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9144185>. Acesso em: 09 jul. 2021.

COUTINHO, C. B. P. Um visitante indesejado. Editora SBCSaúde. Goiânia, 2021. Disponível em: <https: editorasaude.com.br>. Acesso em: 20 jul. 2021.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285, 2018. Disponível em: <https://arxiv.org/abs/1603.07285>.Acesso em: 08 set. 2021.

ESTRELA, M. C. A. et al. Covid-19: Sequelas fisiopatológicas e psicológicas nos pacientes e na equipe profissional multidisciplinar. Brazilian Journal of Development, v. 7, n. 6, p. 59138-59152, 2021.Disponível em: <https://www.brazilianjournals.com/index.php/BRJD/article/view/31398>. Acesso em: 18 jul. 2021.

FERREIRA, A. B. H. Novo Dicionário Aurélio Século XXI.3.ed.(2ª impressão). Curitiba: Editora Positivo, 2004.

FIOCRUZ. Impactos sociais, econômicos, culturais e políticos da pandemia, [2020]. Disponível em: <https://portal.fiocruz.br/impactos-sociais-economicos-culturais-e-politicos-da-pandemia>. Acesso em: 18 jul. 2021.

HEMDAN, E. E.; SHOUMAN, M. A.; KARAR, M. E. Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. arXiv preprint arXiv:2003.11055, 2020. Disponível em: <https://arxiv.org/abs/2003.11055>. Acesso em: 22 ago. 2021.

JIMENEZ-SOLEM, E. et al. Developing and validating COVID-19 adverse outcome risk prediction models from a bi-national European cohort of 5594 patients. Scientific reports, v. 11, n. 1, p. 1-12, 2021. Disponível em: <https://www.nature.com/articles/s41598-021-81844-x>. Acesso em: 25 fev. 2021.

KAGGLE. 2021. Banco de dados de radiografia COVID-19. Disponível em: <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>. Acesso em: 14 nov. 2021.

KARPATHY, A. Stanford University cs231n: Convolutional neural networks for visual recognition.CS231n Convolutional Neural Networks for Visual Recognition. 2019. Disponível em: <https://cs231n.github.io/>. Acesso em: 05 set. 2021.

KERAS. Por que escolher Keras?, 2021. Disponível em: <https://keras.io/why_keras/ >. Acesso em: 14 nov. 2021

KINSLEY, H.; KUKIETA, D. Neural Networks form Scratch – P1 Intro and Neuron Code. [S. l.: s. n.], 2020. 1 vídeo (16,38 mim). Disponiível em: <https://www.youtube.com/watch?v=Wo5dMEP_BbI>. Acesso em: 19 set. 2021.

KOURY, M. G. P. O Covid-19 e as emoções: Pensando na e sobre a pandemia. RBSE Revista Brasileira de Sociologia da Emoção, v. 19, n. 55, p. 13-26, 2020. Disponível em: <https://grem-grei.org/wp-content/uploads/2020/05/2020_4_Koury_O-COVID-19-e-as-emo%C3%A7%C3%B5es-pensando-na-e-sobre-a-pandemia-pandemia.pdf>. Acesso em: 18 de jul. 2021.

LAGUARTA S., J.; HUETO, F.; SUBIRANA, B. COVID-19 Artificial Intelligence Diagnosis Using Only Cough Recordings. 2020. IEEE Open Journal of Engineering in Medicine and Biology, v. 1, p. 275-281, 2020. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9208795>. Acesso em: 18 abr. 2020.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. Nature, v. 521, n. 7553, pág. 436-444, 2015. Disponível em: <https://www.nature.com/articles/nature14539>. Acesso em: 22 ago. 2021.

MACHADO, D.; SCHERTEL, L. Tecnologias de perfilamento e dados agregados de geolocalização no combate à COVID-19 no Brasil: Uma análise dos riscos individuais e coletivos à luz da LGPD. Revista Brasileira de Direitos Fundamentais & Justiça [no prelo/Forthcoming], 2020. Disponível em: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3751990>. Acesso em: 18 abr. 2020.

McCarthy, J.. What is Artificial Intelligence?, 2007. Disponível em: <http://www-formal.stanford.edu/jmc/whatisai.pdf>. Acesso em: 04 nov. 2019.

MOONS, K. G. et al. Prognosis and prognostic research: What, why, and how?. Bmj, v. 338, 2009. Disponível em: <https://www.bmj.com/content/338/bmj.b375.long>. Acesso em: 11 ago. 2021.

MOURA, E. C. et al. Disponibilidade de dados públicos em tempo oportuno para a gestão: Análise das ondas da COVID-19, 2021. Disponível em: <https://repositorio.unb.br/handle/10482/40924>. Acesso em: 11 jul. 2021.

NEDEL, F. B. Enfrentando a COVID-19: APS forte agora mais que nunca!. APS em Revista, v. 2, n. 1, p. 11-16, 2020. Disponível em: <https://apsemrevista.org/aps/article/view/68>. Acesso em:18 de jul. 2021.

KFOURI NETO, M.; SILVA, R. G.; NOGAROLI, R. INTELIGÊNCIA ARTIFICIAL E BIG DATA NO DIAGNÓSTICO E TRATAMENTO DA COVID-19 NA AMÉRICA LATINA. Revista Brasileira de Direitos Fundamentais & Justiça, v. 14, n. 1, p. 149-178, 2020. Disponível em:<

http://dfj.emnuvens.com.br/dfj/article/view/974 >. Acesso em: 18 abr. 2021.

NGUYEN, T. T. et al. Artificial intelligence in the battle against coronavirus (COVID-19): A survey and future research directions. arXiv preprint arXiv:2008.07343, 2020. Disponível em: <https://arxiv.org/abs/2008.07343>. Acesso em: 20 out. 2021.

NICOLELIS, M. Um milhão de mortos em 2021 e a pandemia está longe de acabar. Correio Braziliense, 2021. Disponível em: <https://www.correiobraziliense.com.br/opiniao/2021/07/4938221-um-milhao-de-mortos-em-2021-e-a-pandemia-esta-longe-de-acabar.html. Acesso em: 20 jul. 2021.

NIELSEN, M. A. Neural networks and deep learning. San Francisco, CA: Determination press, 2015. Disponível em: <neuralnetworksanddeeplearning20200415-115041-1t7vxpc-with-cover-page-v2.pdf (d1wqtxts1xzle7.cloudfront.net) >. Acesso em: 05 de set. 2021

OPAS. Folha informativa sobre COVID-19, 2020. Disponível em: <https://www.paho.org/pt/covid19>. Acesso em: 11 jul. 2021.

ORNELL, F. et al. Pandemia de medo e COVID-19: impacto na saúde mental e possíveis estratégicas. Revista debates in psychiatry, v. 2020, 2020. Disponível em: < http://www.ufrgs.br/ufrgs/noticias/arquivos/pandemia-de-medo-e-COVID-19-impacto-na-saude-mental-e-possiveis-estrategias>. Acesso em: 18 jul. 2021.

OZTURK, T. et al. Automated detection of COVID-19 cases using deep neural networks with X-ray images. Computers in biology and medicine, v. 121, p. 103792, 2020. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/32568675/>. Acesso em: 20 ago. 2021.

PEREIRA, R. M.; COSTA, Y. M.; SILLA, C. N. Handling imbalance in hierarchical classification problems using local classifiers approaches. Data Mining and Knowledge Discovery, p. 1-58, 2021. Disponível em: < https://link.springer.com/article/10.1007/s10618-021-00762-8>. Acesso em: 18 out. 2021.

PEREIRA, R. M. et al. COVID-19 identification in chest X-ray images on flat and hierarchical classification scenarios. Computer Methods and Programs in Biomedicine, v. 194, p. 105532, 2020. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169260720309664>. Acesso em: 16 out. 2021.

PINHEIRO, A. F. Inteligência Artificial Fundamentos e Aplicabilidades. Recife, 2020. Disponível em: <https://alvarofpinheiro.webnode.com/_files/200001248-8389583897/Intelig%C3%AAncia%20Artificial%20Fundamentos%20e%20Aplicabilidades.pdf>. Acesso em: 31 ago. 2021.

PRADO, N. M. B. L. et al. The international response of primary health care to COVID-19: Document analysis in selected countries. Cadernos de Saúde Pública, v. 36, 2020. Disponível em: <https://www.scielo.br/j/csp/a/7ws7tVbWLS7LYk559MBJfLL/?format=html&lang=en>. Acesso em: 18 jul. 2021.

RAHMAN, T. et al. Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. Computers in biology and medicine, v. 132, p. 104319, 2021. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S001048252100113>. Acesso em: 09 jul. 2021.

REVZIN, M. V. et al. Multisystem imaging manifestations of COVID-19, part 1: Viral pathogenesis and pulmonary and vascular system complications. Radiographics, v. 40, n. 6, p. 1574-1599, 2020. Disponível em: <https://pubs.rsna.org/doi/full/10.1148/rg.2020200149>. Acesso em:17 out. 2021.

RILEY, R. D. et al. Calculating the sample size required for developing a clinical prediction model. Bmj, v. 368, 2020. Disponível em: <https://www.bmj.com/content/368/bmj.m441?ijkey=be69cc727c62254e5c5e16c3f10f93661f6b64c2&keytype2=tf_ipsecsha>. Acesso em: 06 abr. 2021.

RODRIGUES, L. F. Comparação entre redes neurais convolucionais e técnicas de pré-processamento para classificar células HEp-2 em imagens de imunofluorescência. 2018. 65f. Dissertação (Pós-Graduação em Ciência da Computação) – Universidade Federal de Viçosa, Viçosa, 2018.

RODRIGUES, V. Conhecendo a visão do computador: Redes Neurais Convolucionais, 2019. Disponível em: <https://vitorborbarodrigues.medium.com/conhecendo-a-vis%C3%A3o-do-computador-redes-neurais-convolucionais-e1c2b14bf426>. Acesso em: 26 set. 2021.

SAMPAIO, C. Data Science para programadores: Um guia completo utilizando a linguagem Python. Rio de Janeiro: Ciência Moderna, 2018.

SANCHO, K. A.; PFEIFFER, C. R. C.; CORRÊA, C. R. S. Medicalização, diagnóstico clínico e queixa-conduta–redes de significação em jogo. Interface-Comunicação, Saúde, Educação, v. 23, p. e170633, 2019.

SARKAR, D.; BALI, R.; GHOSH, T. Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras. Packt Publishing Ltd, 2018.

SHAMOUT, F. E. et al. An artificial intelligence system for predicting the deterioration of COVID-19 patients in the emergency department. NPJ digital medicine, v. 4, n. 1, p. 1-11, 2021. Disponível em: <https://www.nature.com/articles/s41746-021-00453-0>. Acesso em: 17 ago. 2021.

SHARMA, A. Differences Between Machine Learning & Deep Learning, 2018. Disponível em: <https://www.datacamp.com/community/tutorials/machine-deep-learning>. Acesso em: 17 fev. 2021.

SILVA JÚNIOR, A. P. da; BEZERRA, D. G. S.; ANDRADE, Y. S. Comparação de arquiteturas de Deep Learning para segmentação de imagens dermatoscópicas de melanoma. 2020. 84f. Dissertação (Graduação em Engenharia Eletrônica) – Universidade de Brasília - UnB, Brasília, 2020. Disponível em: <https://bdm.unb.br/bitstream/10483/27785/1/2020_AntonioPrado_DiogoBezerra_YasmineAndrade_tcc.pdf>. Acesso em: 20 out. 2021.

SPERRIN, M.; GRANT, S. W.; PEEK, N. Prediction models for diagnosis and prognosis in COVID-19. BMJ (Clinical Research Edition), v. 369, p. m1464, 2020. Disponível em: <https://www.bmj.com/content/369/bmj.m1464.full>. Acesso em: 14 nov. 2021

UNICEF. A pandemia de COVID-19 leva a um grande retrocesso na vacinação infantil, mostram novos dados da OMS e do UNICEF, 2021. Disponível em: <https://www.unicef.org/brazil/comunicados-de-imprensa/pandemia-de-COVID-19-leva-a-um-grande-retrocesso-na-vacinacao-infantil>.Acesso em: 18 jul. 2021.

WORLD HEALTH ORGANIZATION. WHO Coronavirus (COVID-19) Dashboard. WHO Coronavirus disease (COVID-19) dashboard. Geneva (Switzerland). 2021. Disponível em: < https://covid19.who.int/>. Acesso em: 11 jul. 2021.

WYNANTS, L. et al. Prediction models for diagnosis and prognosis of COVID-19: Systematic review and critical appraisal. BMJ, v. 369, 2020. Disponível em: <https://www.bmj.com/content/369/bmj.m1328>. Acesso em: 25 fev. 2021.

XU, A. Y. Detecting COVID-19 induced Pneumonia from Chest X-rays with Transfer Learning: An implementation in Tensorflow and Keras. Towards Data Science, 2020. Disponível em: <https://towardsdatascience.com/detecting-COVID-19-induced-pneumonia-from-chest-x-rays-with-transfer-learning-an-implementation-311484e6afc1>. Acesso em: 24 mai. 2021

ZU, Z. Y. et al. Coronavirus disease 2019 (COVID-19): A perspective from China. Radiology, v. 296, n. 2, p. E15-E25, 2020. Disponível em: <https://pubs.rsna.org/doi/full/10.1148/radiol.2020200490>. Acesso em: 22 ago. 2021.