



**FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO ENGENHARIA DA COMPUTAÇÃO**

WALNER DE OLIVEIRA
PESSOA
RA 21801644

**Identificar autoridades por meio de reconhecimento facial.
Uso de tecnologia de visão computacional como alternativa
para antigo processo de fotogramas (Carômetro).**

BRASÍLIA
2020



Walner de Oliveira Pessoa

Identificar autoridades por meio de reconhecimento facial. Uso de tecnologia de visão computacional como alternativa para antigo processo de fotogramas (Carômetro)

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do de Computação do UniCEUB– Centro Universitário de Brasília

Orientador: Me. William Roberto Malvezzi

BRASÍLIA
2020



Walner de Oliveira Pessoa

Identificar autoridades por meio de reconhecimento facial. Uso de tecnologia de visão computacional como alternativa para antigo processo de fotogramas (Carômetro).

Trabalho de Conclusão de Curso (TCC) apresentado
como um dos requisitos para a conclusão do curso de
Engenharia de Computação do UniCEUB – Centro
Universitário de Brasília

Orientador (a): Me. William Roberto Malvezzi

Brasília, 2020.

BANCA EXAMINADORA

Me. William Roberto Malvezzi.
Orientador (a)

XXXXXXXXXXXXXXXXXXXX
Examinador (a)

XXXXXXXXXXXXXXXXXXXX
Examinador (a)

Identificar autoridades por meio de reconhecimento facial. Uso de tecnologia de visão computacional como alternativa para antigo processo de fotogramas (Carômetro).

Identify authorities through facial recognition. Use of computer vision technology as an alternative to the old frame process (Carômetro).

Walner de Oliveira Pessoa¹, William Roberto Malvezzi ², XXXXX ³, XXXXXXX

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 trabalhos 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.

Abstract:

Within the context of professionals working in Government Relations, the recognition of parliamentarians in physical environments is an essential task. These professionals carry out an activity that brings to parliamentarians the interests of the categories that lobbyists represent, that is, one of the processes that ensure democracy for our country. The traditional form of recognition used is made by printed frames that the professional has to walk in his hand to identify (recognize) the parliamentary authority. One solution to this task is to seriously use computer vision techniques to help identify, quickly and accurately, an authority in an event that circulates dozens or hundreds of people.

Studies on computer vision are recent, in the 70s two works were precursors - **The Psychology of Computer Vision and A framework to represent knowledge**. From 2012 with the victory of the ImageNet computer image recognition contest, Alex Krizhevsky from the University of Toronto with a neural network called AlexNet opened the frontiers for advances in neural network models and hardware with NVIDIA GPU architectures (today the GPU is available to any user of the Google Colab platform). Innovations like these were instrumental in making it accessible for anyone to develop facial recognition solutions on their home computers. This technical study presents the use of libraries such as Dlib and OpenCV, which simplifies the development of applications to train a model with images of federal deputies, for example, and offer a tool for lobbyists to stop using frames with the image and name of parliamentarians (“**Carômetro**”) printed on paper and start using Deep Learning technological solutions within the reach of lay people.

keywords: Facial detection, Artificial Intelligence, Deep Learning, OpenCV, Dlib, Encoding, Face Embeddings, CNN Method, Python, HOG Method, Lobby, Turnover, Defense of interest.

¹ UniCEUB, aluno.

² UniCEUB, orientador.

³ UniCEUB, primeiro examinador.

⁴ UniCEUB, segundo examinador.

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 reconstróem 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 entre os N dados comparando ao dentro de cada cluster. (JAIN, 1999)

2.3.3 Machine Learning 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 formado 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).

Figura 1. Linha do tempo das tecnologias



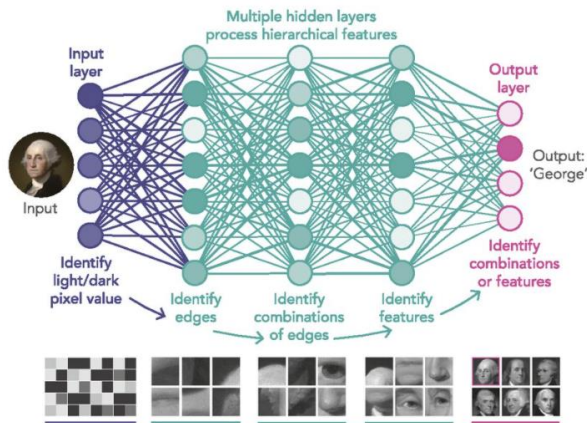
Fonte: AKABANE, 2018

Essa tecnologia busca resolver problemas específicos da indústria e cada desenvolvimento é único. Não dá para treinar um modelo usado para verificar maçãs poderosas, 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 2: 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 ($8 \text{ bits} - 2^8$).

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 $R = 255(\text{FF})$, $G = 00(00)$ e $B = 00(00)$ em decimal ou FF0000 em notação hexadecimal (GONZALEZ, 2000).

Figura 3: Valores válidos para RGB

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).

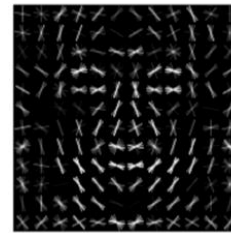
A magnitude e a direção do gradiente usando provem da fórmula abaixo, mesma para cálculo no Detector de Sobel:

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

$$\theta = \arctan \frac{g_y}{g_x}$$

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 versão 19.7 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

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

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++.

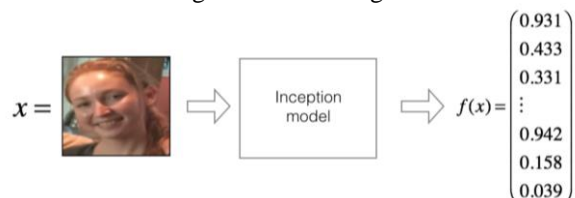
O OpenCV está organizado em módulos, ou seja, os grupos de pacotes incluem muitas bibliotecas como, Processamento de imagem (imgproc), estrutura de recursos 2D (features2d), detecção de objeto (objdetect) dentre outros. (OPENCV, 2020)

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 benchmark 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 conterà 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

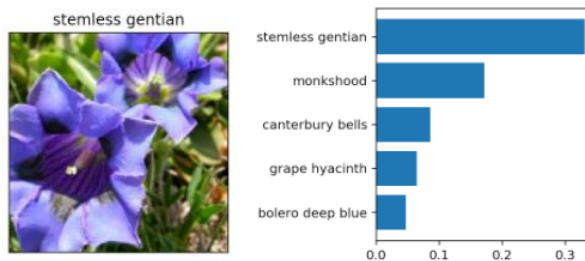
4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

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
Raspberry Pi	Computer Vision
OpenCV Tutorials	Face Applications
Object Detection	Object Tracking
Interviews	Medical Computer
dlib	Vision

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 nos estudos 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 rodar 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.**

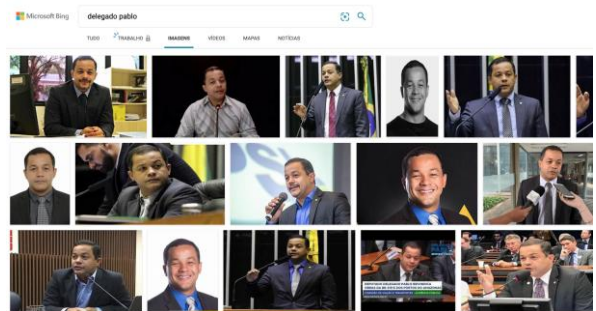
Para qualquer treinamento de faces o primeiro desafio é construir um conjunto de dados. A resolução usual para imagens de entrada no processo de reconhecimento facial deve estar no limite entre 64×64 até 224×224 pixel. O tamanho da imagem que ela ocupa do disco não importa tanto para o resultado final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

```
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

```
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())
~
~
```

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

Esse algoritmo identifica, recorta e alinha todos os rostos encontrados em cada foto, de forma que haverá muitos rostos que precisam ser apagados manualmente pois eles não são alvo da pessoa que está sendo conteúdo de treinamento.

Figura 12: Imagens descartadas



Fonte: montada pelo autor

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 machine 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

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.

```
13 ap = argparse.ArgumentParser()
14 ap.add_argument("-i", "--dataset", required=True,
15                 help="path to input directory of faces + images")
16 ap.add_argument("-e", "--encodings", required=True,
17                 help="path to serialized db of facial encodings")
18 ap.add_argument("-d", "--detection-method", type=str, default="cnn",
19                 help="face detection model to use: either 'hog' or 'cnn'")
20 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

```
macbook-wop:face-recognition-TCC walnerpessoa$ python encode_faces-TCC.py --da
taset dataset --encodings encodings.pickle
[INFO] quantifying faces...
[INFO] processing image 1/3618
[INFO] processing image 2/3618
[INFO] processing image 3/3618
[INFO] processing image 4/3618
[INFO] processing image 5/3618
[INFO] processing image 6/3618
[INFO] processing image 7/3618
[INFO] processing image 8/3618
```

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:

1. Encontrar a face
2. Analisar as características
3. Comparar com as faces conhecidas
4. 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. certificar 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;

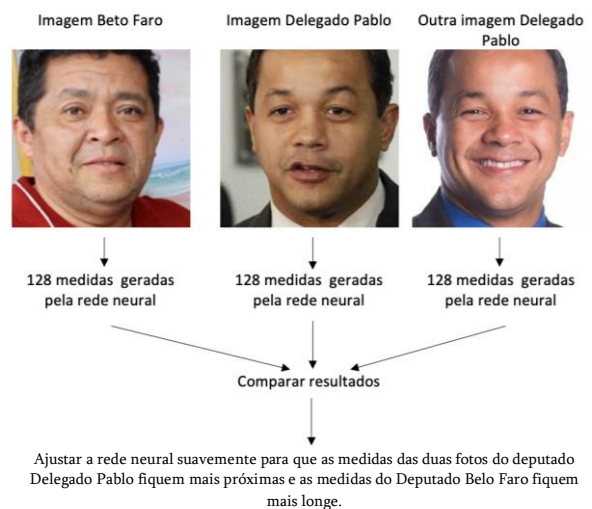
3. medir as características da face que está em análise e encontrar a codificação de 128-d (*encoding faces*) para poder comparar os vetores de características únicas das medidas do rosto de forma para encontrar as semelhanças no rosto

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



Fonte: derivada a partir de Geitgey,2016

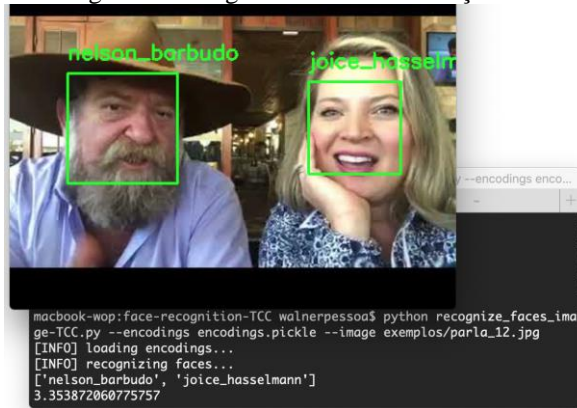
A **biblioteca dlib**, mantida por Davis King, é utilizada para construir os empacotamentos (*embeddings*) de rosto. A **biblioteca face_recognition**, criada por Adam Geitgey, está relacionada com as funcionalidades de reconhecimento facial do dlib.

```
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

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 booleanos 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 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.

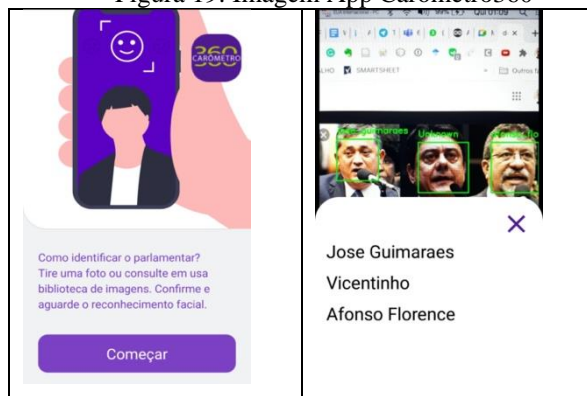
A solução foi testada de forma local no computador do pesquisador e a próxima etapa seria colocar o sistema em produção.

A App foi desenvolvida 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 dos 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 solicitada 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 Aplicações,** 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/ini27/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.wordpress.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://](https://www.kaggle.com/ini27/fastai-v3-lesson-2) >

- www.learnopencv.com/histogram-oriented-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/](https://www.pyimagesearch.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