Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Redes neurais convolucionais aplicadas ao reconhecimento facial em indivíduos com máscara

Ítalo Trindade Noé

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Prof. Dr. Talles Henrique de Medeiros

Agosto, 2021
João Monlevade–MG

Ítalo Trindade Noé

Redes neurais convolucionais aplicadas ao reconhecimento facial em indivíduos com máscara

Orientador: Prof. Dr. Talles Henrique de Medeiros

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Agosto de 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

N763r Noé, Ítalo Trindade.
Redes neurais convolucionais aplicadas ao reconhecimento facial em indivíduos com máscara. [manuscrito] / Ítalo Trindade Noé. - 2021.
47 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Talles Henrique de Medeiros.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia de Computação .

1. Inteligência artificial. 2. Redes neurais (Computação). 3. Rede neural convolucional. 4. Transferência de aprendizagem. I. Medeiros, Talles Henrique de. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.85

Bibliotecário(a) Responsável: Sione Galvão Rodrigues - CRB6 / 2526

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Ítalo Trindade Noé

Redes Neurais Convolucionais
Aplicadas ao Reconhecimento Facial em
Indivíduos com Máscara

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Aprovada em 30 de agosto de 2021

Membros da banca

Dr. - Prof. Talles Henrique de Medeiros - Orientador - Universidade Federal de Ouro Preto
Dr. - Prof. Luiz Carlos Bambirra Torres - Universidade Federal de Ouro Preto
Dra. - Prof. Gilda Aparecida de Assis - Universidade Federal de Ouro Preto

Prof. Talles Henrique de Medeiros, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 20/09/2021

Documento assinado eletronicamente por **Talles Henrique de Medeiros**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 20/09/2021, às 12:20, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0222751** e o código CRC **432DB39A**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é dedicado aos meus pais, que sempre me apoiaram em todas as etapas da minha vida.

Agradecimentos

Agradeço a minha família por todo o apoio. Especialmente aos meus pais, que sempre batalharam juntos para que nada me faltasse e minha irmã por estar ao meu lado em todas as decisões.

Agradeço aos meus companheiros de República pela amizade, aprendizados e por tornarem os dias em João Monlevade mais fáceis.

Agradeço a todos meus amigos que me acompanharam e ajudaram a enfrentar diferentes desafios ao longo dessa jornada.

Agradeço aos professores pela dedicação e ensinamentos ao longo do curso. Especialmente ao meu orientador Talles Henrique pelo seu tempo, paciência e conselhos, que foram fundamentais no desenvolvimento deste trabalho.

"Many of life's failures are people who did not realize how close they were to success when they gave up."

— Thomas Edison (1847 – 1931)

Resumo

Em meio a pandemia do coronavírus, o uso de máscaras se tornou uma das principais formas de prevenção e controle da disseminação desse vírus. Entretanto, as máscaras impactaram o desempenho de diversos modelos de reconhecimento facial, pela redução de características visíveis em imagens. O objetivo deste trabalho é apresentar um modelo de redes neurais convolucionais com aprendizado por transferência capaz de classificar trinta indivíduos independente da utilização de máscaras. O modelo foi treinado em uma base de dados com imagens reais de máscaras e outro com a inserção de máscaras simuladas computacionalmente. Os resultados obtidos de acurácia para as bases são respectivamente 91,94% e 93,54%. Pela quantidade de classes e limitações da base de dados utilizada, o resultado é coerente com o baixo número de trabalhos relacionados e evidencia a complexidade do problema. Acredita-se que a utilização de uma base de dados com qualidade e quantidade de imagens superior tornaria a utilização do modelo mais viável para o mundo real, porém os testes apresentados são promissores.

Palavras-chaves: Reconhecimento facial. Redes Neurais Convolucionais. Aprendizado por transferência.

Abstract

Amidst the coronavirus pandemic, the use of masks has become one of the main ways to prevent and control the spread of this virus. However, masks impacted the performance of several face recognition models, by reducing visible features in images. The objective of this work is to present a model of convolutional neural networks with transfer learning capable of classifying thirty individuals regardless of the use of masks. The model was trained in a database with real images of masks and another with the insertion of computationally simulated masks. The accuracy results obtained for the bases are respectively 91.94% and 93.54%. Due to the number of classes and limitations of the database used, the result is consistent with the low number of related works and highlights the complexity of the problem. It is believed that the use of a database with superior quality and quantity of images would make the use of the model more viable for the real world, but the tests presented are promising.

Key-words: Facial recognition. Convolutional Neural Networks. Transfer learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANN Redes Neurais Artificiais

CL Camada Convolucional

CNN Redes Neurais Convolucionais

MLP Perceptron Multicamadas

NIST Instituto Nacional de Padrões e Tecnologia

ReLU Unidade Linear Retificada

RMFD *Real-World Masked Face Dataset*

SMOTE *Synthetic Minority Over-sampling Technique*

Sumário

1 Introdução

O reconhecimento facial é uma área que apesar de existir a décadas, teve maior destaque nos últimos anos por estar mais presente no cotidiano das pessoas. Essa tecnologia é aplicada, por exemplo, como forma de desbloqueio de aparelhos celulares, controle de acesso em áreas privadas, sistemas de segurança e busca de criminosos em áreas públicas. Então, para que o reconhecimento facial tenha aceitação pública, suas aplicações dependem de modelos com a menor taxa de erro possível. Entretanto, o uso de máscaras se tornou um fator limitante.

No cenário atual, a mortal pandemia da COVID-19 modificou o cotidiano das pessoas pela facilidade de disseminação do vírus pelo ar e superfícies. Por isso, o uso de máscaras faciais se tornou um meio essencial e comprovado de proteção no cotidiano das pessoas (CHENG et al., 2020). Porém modelos de reconhecimento facial foram diretamente afetados com o uso de máscaras pela redução de áreas faciais detectáveis, como a boca e o nariz.

Além da redução na capacidade de classificação de faces com máscara, outro fator que indica a necessidade de estudos nessa área é o uso desta tecnologia como forma de prevenção ao ser aplicada na área biométrica. Meios comumente utilizados para biometria, como o uso de cartões eletrônicos e impressão digital, não são recomendados nesta pandemia por exigirem o contato com superfícies potencialmente contaminadas (OKEREAFOR et al., 2020).

Dessa forma, a biometria facial ganha destaque como alternativa para identificação de pessoas e combate a fraudes de identidade de maneira segura a saúde. Portanto, esse trabalho busca utilizar metodologias atuais para contornar o problema do uso de máscaras aplicado ao reconhecimento facial. Para isso, serão apresentadas sugestões para adaptação de modelos existentes e formas para criação de novos classificadores.

1.1 O problema de pesquisa

De acordo com o Instituto Nacional de Padrões e Tecnologia (NIST), a precisão dos algoritmos de reconhecimento facial anteriores a pandemia diminuiu substancialmente quando aplicados em rostos mascarados (NGAN; GROTHER; HANAOKA, 2020). Enquanto os principais algoritmos possuíam uma taxa de erro de 0,3% para faces sem máscara, agora houve uma redução para cerca de 5% de erro com a inserção das máscaras. Outros algorítimos também eficientes falharam de 20% até 50%.

A maioria dos modelos de reconhecimento facial utilizam redes neurais convoluci-

onais para extração de características faciais. Essa arquitetura classifica uma face pela identificação das características de forma individual, ou seja, basta uma imagem ter o olho, boca e nariz de determinada pessoa que ela poderá ser identificada, independente da posição espacial desses elementos. Então o uso de máscaras restringe a extração de características ao olho e regiões próximas a ele, o que explica a queda de precisão de modelos.

Outro problema é a dependência de grandes quantidades de dados para treinamento de redes neurais convolucionais de forma eficiente. Isso ocorre porque variações de iluminação, angulação, cores podem ser fatores que reduzem essa precisão. São raras as bases de dados que foram criadas com amostras de pessoas com máscara e as que possuem essa característica geralmente não contêm quantidades suficientes de imagens.

1.2 Objetivos

O presente trabalho consiste na elaboração de modelos para reconhecimento facial de indivíduos com máscara e apresentar sugestões de diferentes metodologias que podem ser aplicadas para adaptação de modelos já existentes. Também deseja-se avaliar a precisão dessas arquitetura em cenários reais, principalmente ao se tratar de aplicações que auxiliem a prevenção de COVID-19. Portanto este trabalho possui os seguintes objetivos específicos:

- Apresentar alternativas para base de dados já existentes;
- Criar modelos para reconhecimento facial de indivíduos com máscara;
- Classificar uma grande quantidade de classes de indivíduos;
- Validar a capacidade do modelo em cenários reais.

1.3 Metodologia

O objeto de pesquisa deste trabalho são redes neurais convolucionais aplicadas ao reconhecimento facial. Além disso, será avaliada a capacidade do aprendizado por transferência em auxiliar na criação de novos modelos. Então, os passos para execução deste trabalho são assim definidos:

- Revisão da literatura em aprendizado por transferência e redes neurais convolucionais aplicados ao reconhecimento facial;
- Escolha da base de dados de faces que contenham máscaras;
- Pré-processamento da base de dados escolhida;

- Desenvolvimento dos modelos;
- Treinamento dos modelos com variação de hiperparâmetros;
- Validação dos modelos;
- Análise e discussão dos resultados.

1.4 Organização do trabalho

O restante deste trabalho é organizado como se segue:

- O Capítulo 2 apresenta os trabalhos relacionados ao reconhecimento facial de indivíduos com máscara;
- O Capítulo 3 apresenta os conceitos fundamentais de aprendizado de máquina e visão computacional;
- O Capítulo 4 apresenta como foi realizada a construção do modelo de classificação;
- O Capítulo 5 apresenta os resultados obtidos no trabalho;
- O Capítulo 6 apresenta as considerações finais e trabalhos futuros.

2 Revisão bibliográfica

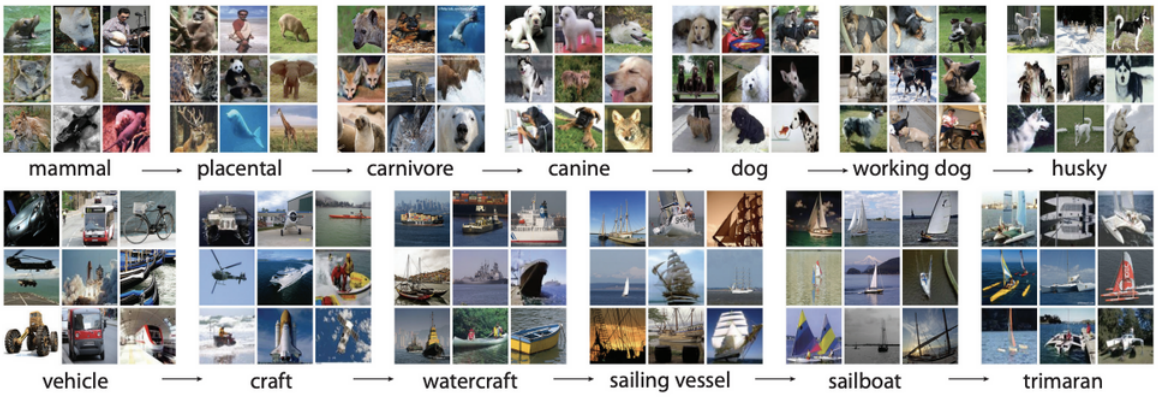
O primeiro relato de tentativa de reconhecimento facial por um computador é de Bledsoe (1966), em que ele tentou encontrar a relação de uma fotografia com registros de fotos em um livro. Neste estudo, Bledsoe foi capaz de evidenciar diferentes problemas que são recorrentes até hoje no reconhecimento facial como: variabilidade de rotação e inclinação da cabeça, intensidade e ângulo de iluminação, expressão facial e envelhecimento. Anos depois, Sirovich e Kirby (1987) foram capazes de mostrar que imagens de rostos poderiam ser representadas de forma eficiente com Análise de Componentes Principais (PCA). Este estudo serviu de inspiração para Turk e Pentland (1991) mostrarem que o PCA é capaz de avaliar a distância entre rostos em um espaço de faces e assim determinar rostos similares.

Com a otimização na representação de uma imagem facial, surgiu a necessidade de mais dados para aplicação em cenários reais. Então houve uma aposta de que o reconhecimento facial seria uma forma mais poderosa de identificar indivíduos quando comparada a impressão digital ((PHILLIPS et al., 2000)). Essa aposta incentivou a criação da primeira base de dados de faces em larga escala, chamada de Face Recognition Technology (FERET). Então, com os avanços no reconhecimento facial, percebeu-se que era necessário uma base de dados com imagens mais variadas e naturais do cotidiano para modelos de reconhecimento facial serem aplicados no mundo real. Para solucionar isso, começaram a ser produzidas bases de dados com imagens coletadas da internet e esse aumento exponencial de dados viabilizou a popularização do aprendizado profundo.

Assim, a arquitetura *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) foi um marco na área de visão computacional e uso de redes neurais convolucionais profundas. Esse título foi conquistado por ser a primeira arquitetura em redes neurais convolucionais a atingir o primeiro lugar no desafio da base *ImageNet* (DENG et al., 2009) quando comparadas as metodologias tradicionais de *machine learning*. Esta base continha 1000 classes com mais de um milhão de imagens (Figura 1) e a rede *AlexNet* reduziu em 10% o erro de classificação. Então, o uso de redes neurais convolucionais para classificação de imagens se tornou a opção mais utilizada até os dias atuais.

Outro marco no reconhecimento facial foi pelo trabalho Taigman et al. (2014). Criado por pesquisadores do Facebook, foi capaz de ultrapassar pela primeira vez a capacidade humana de reconhecimento de faces. Isso foi possível com uma rede convolucional de nove camadas treinada no maior conjunto de dados com faces para a época, com quatro milhões de imagens faciais pertencentes a mais de 4.000 identidades, todas retiradas da própria rede social. Essa arquitetura obteve precisão de 97,35%, uma redução em mais de

Figura 1 – Exemplos de classes da base de dados *ImageNet*



Fonte: Krizhevsky, Sutskever e Hinton (2012)

27% do erro em comparação com o estado da arte anterior.

Hariri (2021) explora a mesma base de dados proposta neste trabalho porém para um número maior de classes variando entre 50, 60, 70 e 100 nomes de pessoas. Outra relação com o trabalho atual é a utilização de redes neurais convolucionais com aprendizado por transferência. Como metodologia de pré-processamento, as imagens da base foram rotacionadas até que os olhos estejam alinhados paralelos as bordas horizontais da imagem. Além disso, as regiões da máscara foram removidas, restando apenas as regiões superiores a ela para treinamento. Com isso, a rede obteve 91,3% de precisão para 60 classes de imagens reais e 88,9% de precisão para 60 classes de imagens geradas artificialmente.

3 Fundamentação teórica

Neste capítulo serão apresentados todos os conceitos necessários para geração de um modelo de reconhecimento facial através de redes neurais convolucionais e aprendizado por transferência. Na Seção 3.1 será apresentada a base teórica de redes neurais artificiais. Seguido da Seção 3.2 que apresentará os conceitos e técnicas necessárias para redes neurais convolucionais. Por fim, a Seção 3.3 irá apresentar o aprendizado por transferências e suas arquiteturas.

3.1 Redes Neurais Artificiais

Originalmente apresentado em 1943 por McCulloch e Pitts (MCCULLOCH; PITTS, 1943), os neurônios artificiais foram desenvolvidos a fim de criar uma abstração matemática do funcionamento dos neurônio humanos. Redes Neurais Artificiais (ANN) são modelos computacionais formados por conjuntos de neurônios artificiais para identificar dados de entrada através de um treinamento prévio. Estas redes podem possuir estruturas simples, formadas por um único neurônio, ou serem mais complexas, com multicamadas totalmente conectadas.

O uso de redes neurais artificiais para problemas do mundo real é necessário para simular a capacidade do cérebro humano em aprender padrões e tomar decisões intuitivas. Muitos problemas são inviáveis de serem resolvidos com a criação de algoritmos por seres humanos pela complexidade na extração de características em grande quantidade de dados e generalizá-las de forma suficiente para permitir que dados distintos obtenham o mesmo resultado. Então, ANN são fundamentais para resolução de diversos problemas, entre eles destaca-se a classificação.

A classificação consiste em calcular as probabilidades de um determinado dado de entrada pertencer aos conjuntos de saídas possíveis para o problema. Conhecido como aprendizado supervisionado, uma forma de geração do modelo para classificação consiste na coleta de características de maneira experimental, em que são fornecidas entradas (camada de entrada) com seus respectivos valores de saída (camada de saída) e então a rede busca representar as características comuns encontradas em parâmetros treinados (camada oculta).

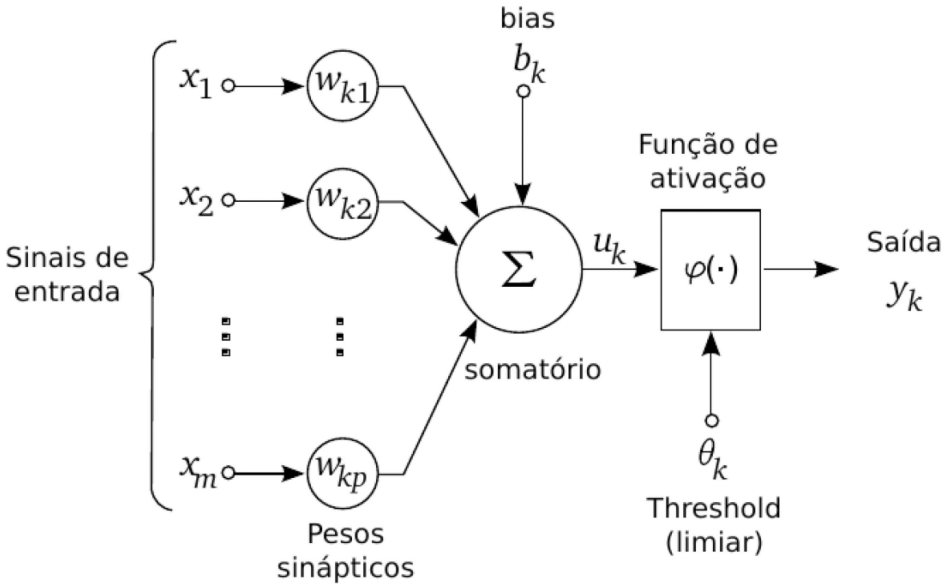
3.1.1 Perceptron

Inspirado nos trabalhos de McCulloch e Pitts, o cientista Frank Rosenblatt criou o modelo mais simples de uma rede neural para aprendizado supervisionado, o *perceptron*

(ROSENBLATT, 1958). Formado por apenas um neurônio, o *perceptron*, demonstrado na Figura 2, é um modelo que recebe uma ou mais entradas x_i e cada entrada é associada a um peso w_i, então é realizada a soma de um bias b ao somatório do produto escalar de $x_i * w_i$. O resultado dessa operação é inserido em uma função degrau que irá retornar 1 para resultados maiores que um valor limiar θ, caso contrário será 0.

$$f(x) = \begin{cases} 0, & \text{se } b + \sum_i x_i * w_i \leq \theta \\ 1, & \text{se } b + \sum_i x_i * w_i > \theta \end{cases} \tag{3.1}$$

Figura 2 – Demonstração de um modelo Perceptron



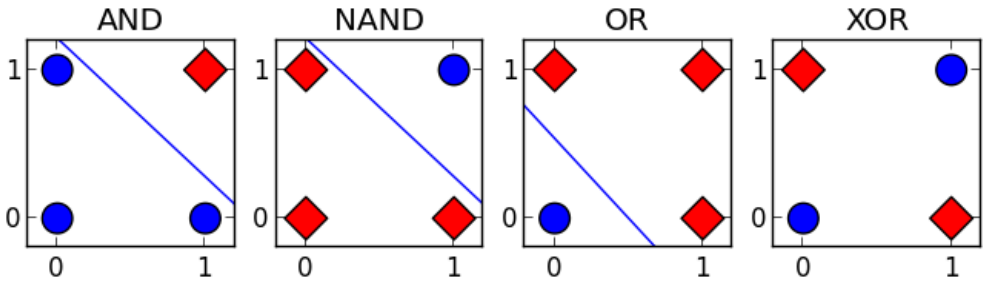
Fonte: Silva e Schimidt (2016)

Por ser construída utilizando apenas uma única camada neural, a rede *perceptron* é capaz de classificar padrões linearmente separáveis, ou seja, problemas do tipo *AND*, *OR* e *NAND*. Porém esta rede é limitada por não conseguir tratar problemas do tipo *XOR*, isto ocorre porque este problema só pode ser resolvido por uma separação não linear como mostrado na Figura 3.

3.1.2 Perceptron Multicamadas

O Perceptron Multicamadas (MLP) é semelhante o *perceptron* simples, porém apresenta pelo menos uma interconexão entre a entrada e a camada de saída, chamada de camada oculta. Com exceção da camada de entrada, todas as outras camadas são constituídas por neurônio. Dessa forma, essa rede é capaz de solucionar problemas não linearmente separáveis, o que as torna extremamente poderosas para problemas complexos.

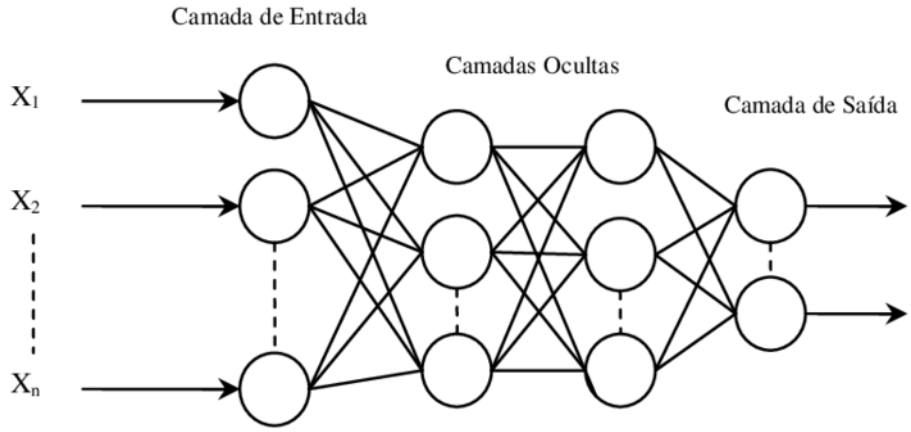
Figura 3 – Problema não linear



Fonte: Hackeling (2017, p. 205)

Por serem redes do tipo *feedfoward*, todas as camadas dessa rede são conectadas as entradas dos neurônios das camadas seguintes, ou seja, os dados apenas transitam na direção da entrada até a saída. Além disso, quando os nós de uma camada são conectados a todas as entradas da camada seguinte, essa rede é considerada completamente conectada (Figura 4).

Figura 4 – Exemplo de um Perceptron Multicamadas



Fonte: Sobreiro et al. (2008)

3.1.3 Aprendizado da rede

As redes MLP são capazes de aprender através do ajuste dos pesos das camadas ocultas com objetivo de reduzir o erro. Para isso é necessário uma função de erro que irá calcular o quão distante a classificação está do valor esperado. Uma função comumente utilizada para isso é a entropia cruzada (*cross-entropy*) que irá penalizar o erro e maximizar

o acerto. Considerando que $\hat{y}_i$ o i-ésimo valor escalar da saída, y_i é o valor desejado, E é definido como o erro e C o número de classes, a entropia cruzada pode ser definida como:

$$\mathrm{E} = -\sum_{i=1}^{C} y_i \cdot \log \hat{y}_i \tag{3.2}$$

Após definida a função de erro, o algoritmo responsável pelo ajuste dos pesos é o *backpropagation* com gradiente descendente. O *backpropagation* irá, após o cálculo do erro, realizar uma retro-propagação do mesmo a fim de atualizar os pesos e consequentemente aproximar o valor esperado do valor classificado.

Como deseja-se minimizar o erro, deve ser aplicado o gradiente descendente. Este algoritmo tem o papel de auxiliar na atualização dos pesos ao aproximar a função de erro ao seu ponto mínimo a passos definidos por um parâmetro fixo chamado de taxa de aprendizado. A taxa de aprendizado é um hiperparâmetro que impacta diretamente no tempo de aprendizado e na capacidade da rede de aprender, um valor muito alto poderá exceder o custo global mínimo e um valor muito pequeno exigirá muitas épocas para convergir.

Outro otimizador muito utilizado para problemas de classificação em múltiplas classes é o *Adam* (KINGMA; BA, 2014). O algoritmo *Adam* é uma extensão do gradiente descendente estocástico clássico. Sua maior vantagem é a parametrização da taxa de aprendizado, permitindo seu ajuste de forma automática durante o treinamento. Dessa forma ele é capaz de convergir mais rapidamente e permite uma maior variação na configuração da taxa de aprendizado.

3.2 Redes Neurais Convolucionais

Ao se tratar de reconhecimento de padrões em imagens, cada amostra pode conter diversos pixels irrelevantes para a classificação. Além disso, pela possibilidade de variações na iluminação, angulação e até mesmo distorções, uma amostragem simples exigiria uma quantidade de amostras extremamente elevada e inviável para o aprendizado computacional. Por estes motivos, as Redes Neurais Convolucionais (CNN) apresentam metodologias e arquiteturas específicas para reduzir o número de parâmetros a serem ajustados pela rede.

A CNN foi inicialmente apresentada pela arquitetura LeNet (LECUN et al., 1998), porém apenas recebeu mais destaque posteriormente pela arquitetura AlexNet. Esta rede pode ser definida como uma rede neural que utiliza a operação matemática convolução no lugar da multiplicação geral da matriz em pelo menos uma de suas camadas. Além da camada de convolução, outras camadas são necessárias nesta arquitetura, como: padding, funções de ativação, pooling, dropout e dense.

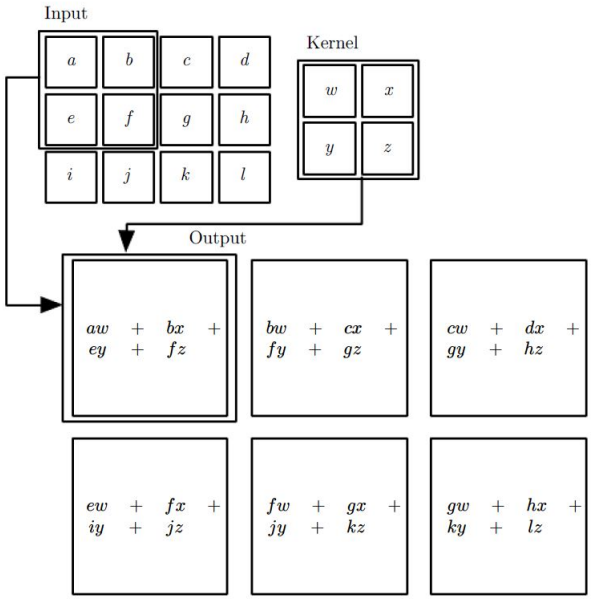
3.2.1 Camada convolucional

A Camada Convolucional (CL) é caracterizada pela soma do produto ponto a ponto de um filtro com uma região subentendida pela sobreposição em função do deslocamento da imagem principal (Figura 5). Então, a saída desta operação pode ser interpretada como um mapeamento de características mais semelhantes entre o filtro e as sub-regiões da imagem, chamado de mapa de recursos. Este mapa representa o peso e o local de um recurso detectado que podem ser bordas, texturas, curvas, retas, entre outras características. Em termos formais, seja $g(x, y)$ a imagem que será aplicada a um filtro $f(x, y)$, a convolução pode ser representada por:

$$f(x,y) * g(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(x,y) * g(x-i, y-j) \tag{3.3}$$

O mapa de recursos possuirá dimensões menores que a sub-região filtrada, porém este tamanho dependerá das dimensões do filtro. Quanto maior as dimensões do filtro, maior será a sub-região e consequentemente o mapa de recursos terá menores dimensões. Com a redução dos parâmetros em cada sub-região, a imagem como um todo será representada com uma redução considerável de parâmetros, mas essa redução dependerá também do tamanho de deslocamento (*stride*) do filtro entre cada sub-região. Assim como o filtro, quanto maior o tamanho do deslocamento maior será a redução nas dimensões do mapa de recursos.

Figura 5 – Exemplo de convolução 2D

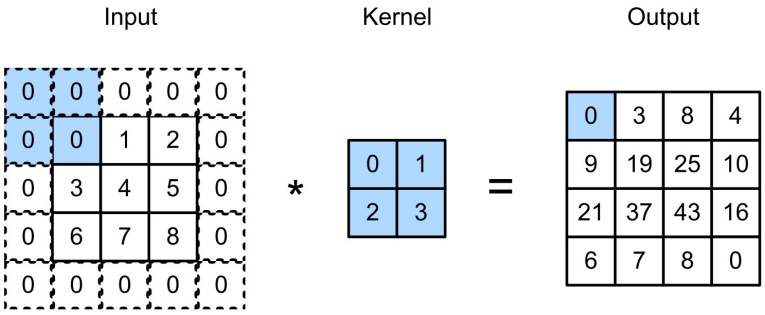


Fonte: Goodfellow et al. (2016, p. 330)

3.2.2 Camada de padding

Após sucessivas aplicações da camada de convolução a CNN pode se tornar menos efetiva pela redução drástica na quantidade de parâmetros que representem a imagem original. Para contornar este problema e aumentar as dimensões obtidas após cada convolução é possível utilizar camadas de padding. Esta camada consiste na adição de pixels nulos nas bordas da imagem para aumentar sua dimensão. Além de aumentar a dimensão do mapa de recursos, esta técnica permite uma conservação maior das bordas da imagem. Como pode ser observado na Figura 6, a inserção de bordas nulas antes da convolução permite a conservação das dimensões dos dados originais.

Figura 6 – Processo de convolução com padding nas bordas



Fonte: Zhang et al. (2021, p. 237)

3.2.3 Função de ativação

Como o problema de classificação de imagens se comporta de forma não linear, é necessário a utilização de um componente matemático após a convolução para introduzir a não-linearidade para rede, chamado de função de ativação. Dentre as funções de ativação existentes, a mais utilizada hoje é a Unidade Linear Retificada (ReLU). A função ReLU irá zerar entradas negativas e preservar os valores positivos, sendo representada por:

$$ReLU(x) = max\{0, x\} \tag{3.4}$$

Além da ReLU, a função *Softmax* é essencial para problemas de classificação e tem como objetivo representar a probabilidade de cada classe da rede para determinado parâmetro de entrada. Por ser uma função de probabilidade, a *Softmax* retorna valores entre 0 e 1, e a soma de todos os valores deve ser igual a 1. Então, a classe que tem maior probabilidade de representar a entrada é a que possuir o maior valor.

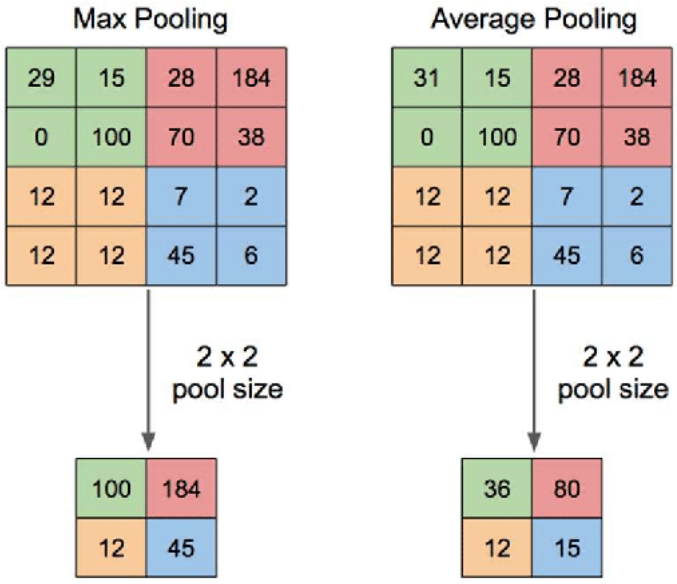
$$Softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))} \tag{3.5}$$

3.2.4 Camada de pooling

Após a camada de convolução, é comum a utilização de uma camada de pooling. A camada de pooling é utilizada para reduzir as dimensões dos parâmetros a serem treinados sem perder as principais características e tornar a rede invariante a transformações geométricas. Desta forma, ocorre uma redução considerável no custo da rede.

Para isso, deve-se definir o tamanho da região sub-amostrada do mapa de ativação, o tamanho do salto em cada iteração e qual o tipo de pooling será aplicado. Para cada região sub-amostrada é retornado um valor de acordo com o resultado da operação realizado no pooling. As operações mais comuns são *MaxPooling* (maior valor da região sub-amostrada), *SumPooling* (soma dos valores da região sub-amostrada), *AveragePooling* (média dos valores da região sub-amostrada). A mais utilizada é a de *MaxPooling* (Figura 7) por obter a característica de maior relevância.

Figura 7 – *MaxPooling* e *AveragePooling* com filtro 2x2 e *stride* 2

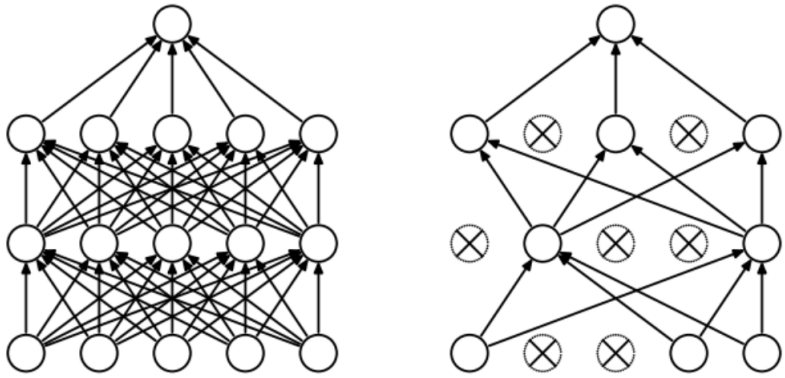


Fonte: Melo (2020)

3.2.5 Dropout

O *Dropout* (SRIVASTAVA et al., 2014) é uma técnica de regularização que propõe a desativação de alguns neurônios em cada interação com base em uma probabilidade. Como mostrado na Figura 8, ao descartar um neurônio, removemos a unidade temporariamente da rede, juntamente com todas as suas conexões de entrada e de saída. Isso força que cada neurônio generalize melhor seu papel e tornar-se independente da presença de outros neurônios. Dessa forma, essa técnica é muito útil para prevenção de *overfiting*.

Figura 8 – Comparação entre uma rede com e sem dropout



Fonte: Srivastava et al. (2014)

3.2.6 Camada Flatten

Após a geração dos mapas de ativação das camadas de convolução e *pooling*, a camada de *flatten* é responsável pela transformação da entrada em um vetor de dimensão única. Assim, a matriz de recursos reduziu os dados contidos na imagem inicial em um vetor mais simples para ser treinado em uma rede totalmente conectada.

3.2.7 Camada Dense

A *Dense* consiste em uma rede neural totalmente conectada que utiliza como entrada o vetor gerado na camada *flatten*. Ao dizer que a rede está totalmente conectada significa que todos os neurônios estão interligados a todos os elementos de entrada. Normalmente é representada por um perceptron de múltiplas camadas. Deve-se informar a dimensão da saída e a função de ativação a ser utilizada e este tipo de camada é a última em uma CNN em conjunto com a função de ativação *Softmax*.

3.3 Aprendizado por transferência

O conceito de aprendizado por transferência permite transpor o conhecimento de uma rede muito profunda em um conjunto de dados diferente com características próximas para resolução de um novo problema (PAN; YANG, 2009). A difusão da área de reconhecimento facial é extremamente beneficiada com este conceito por existirem diversas arquiteturas robustas como *VGG*, *Xception* (CHOLLET, 2017), *ResNet* (HE et al., 2016), *Inception* (SZEGEDY et al., 2015).

As primeiras camadas de uma rede neural convolucional são normalmente utilizadas para detecções de padrões universais e de mais baixo nível, como curvas e bordas. Em

contrapartida, as últimas camadas possuem características mais específicas e por isso podem ser retreinadas com novos dados. Desta forma, os pesos das primeiras camadas podem ser mantidos e as camadas finais substituídas por um novo classificador. Com isso, é possível aplicar parte de modelos robusto para resolução de problemas diferentes daquele proposto por ele. Porém, esta metodologia não pode ser aplicada a qualquer situação e deve ser avaliada a capacidade da arquitetura escolhida em generalizar o problema a ser resolvido.

3.3.1 VGG

As arquiteturas *VGG* (SIMONYAN; ZISSERMAN, 2014) foram treinadas na base de dados *ImageNet* com mais de um milhão de imagens pertencentes a 1000 classes. Estas arquiteturas foram um dos destaques na ILSVRC-2014, principalmente com as arquiteturas *VGG16* e *VGG19*. Apesar de possuírem muitas camadas de profundidade, seu diferencial em relação as arquiteturas anteriores são os resultados superiores para classificação e sua simplicidade na implementação das camadas. Estas arquiteturas possuem como característica apenas filtros de tamanho 3 x 3, diferente da *AlexNet* que utilizava filtros muito maiores (11 x 11 e 5 x 5 na primeira e segunda camada convolucional, respectivamente).

Neste trabalho foi utilizado aprendizado por transferência com a *VGG16*. Esta é uma arquitetura de treze camadas convolucionais e três totalmente conectadas, totalizando dezesseis camadas que originam seu nome. Além disso, as camadas são agrupadas em cinco blocos convolucionais, em que os dois primeiros possuem duas camadas convolucionais e os três últimos possuem três camadas. Outra característica é a utilização de filtros 3x3 com salto 1, este tipo de filtro é o menor filtro possível para obter orientações horizontais e verticais. A diferença de complexidade entre as arquiteturas citadas podem ser comparadas na Figura 9

Figura 9 – Comparação entre as arquiteturas *AlexNet* e *VGG*

Fonte: Li Justin Johnson (2017)

4 Desenvolvimento

Neste capítulo serão apresentados os passos necessários para a elaboração de um modelo de reconhecimento facial através de redes neurais convolucionais e aprendizado por transferência. Na Seção 4.1 será apresentada a base de dados escolhida e suas respectivas alterações. Seguido da Seção 4.2 que apresentará os *frameworks* e bibliotecas utilizadas. Por fim, a Seção 4.3 irá mostrar como foi realizada a construção do modelo e sua estrutura.

4.1 Base de dados

Para geração dos modelos que serão apresentados na Seção 4.3, foi utilizada a base de dados *Real-World Masked Face Dataset* (RMFD) (WANG et al., 2020). Esta é a maior base de dados atualmente de pessoas públicas com e sem máscaras. Criada durante a pandemia de COVID-19, esta base foi desenvolvida para auxiliar no controle da segurança pública por meio de modelos de reconhecimento facial. A RMFD conta com 92614 imagens dispostas em duas pastas (com e sem máscara) e cada uma possui subpastas agrupadas pelo nome das pessoas Tabela 1.

Tabela 1 – Distribuição da base de dados RMFD.

Classes	**Total de Pessoas**	**Total de Imagens**
Sem Máscara	460	90458
Com Máscara	525	2156

Fonte – Produzido pelo autor.

Apesar de conter uma grande quantidade de imagens, é possível perceber que o grupo binário está desbalanceado tanto em número de pessoas quanto no total de imagens. Embora o grupo com máscara possua mais pessoas, a quantidade de imagens contidas é muito inferior, além de possuir ruídos, pastas vazias ou com uma quantidade muito pequena de imagens.

4.1.1 Pré-processamento

Como o objetivo final é classificar pessoas independente do uso de máscaras, faces com máscara possuem poucas regiões reconhecíveis e a principal delas são os olhos. Por conta disso e do desbalanceamento, foi filtrado na base com máscaras pessoas que não estejam de óculos com quantidade de imagens maiores ou iguais a 10. Após a filtragem, foi gerada uma nova base em que as classes são as pessoas resultantes da filtragem com suas

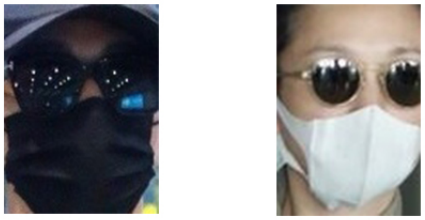
respectivas imagens com e sem máscaras. Assim foram separadas 30 classes de pessoas com um total de 6770 imagens distribuídas de acorda com a Tabela 2.

Após a separação das classes a serem treinadas foi realizada a remoção de ruídos de duas formas: automática (Figura 10) e manual (Figura 11). Primeiramente foi realizada a remoção automática que consistiu na utilização da biblioteca *MTCNN* do *PyTorch* para verificar quais imagens sem máscara correspondem a uma face humana. Já na remoção manual, foram retiradas as imagens restantes que contivessem a maior parte do rosto coberto.

Figura 10 – Imagens removidas pela MTCNN

Fonte: Produzido pelo autor

Figura 11 – Imagens removidas manualmente



Fonte: Produzido pelo autor

Além da base com imagens reais de pessoas com máscara, foi gerada uma nova base de dados com máscaras inseridas de forma artificial na base de dados sem máscara. Dessa forma, busca-se explorar a capacidade de geração de imagens artificiais para reconhecimento de faces no mundo real e contornar a escassez de dados de pessoas com máscara até o momento. Comprovada a eficiência dessa metodologia, poderá ser uma alternativa para adaptação de modelos já existentes.

Para a geração da base de dados com máscaras simuladas foi testado o script *MaskTheFace* (ANWAR; RAYCHOWDHURY, 2020). Como deseja-se comparar com o modelo gerado pelas imagens da base RMFD, apenas foram geradas imagens para as mesmas 30 classes do primeiro dataset. As configurações escolhidas para isso foram máscaras pretas do tipo pano (Figura 12), que são as que apresentaram maior semelhanças com as reais presentes na base de dados original. Como é possível perceber na Tabela 2, a quantidade de imagens de faces com máscara obtidas de forma simulada foi muito maior, porém algumas faces reais não foram detectadas pelo *MaskTheFace* devido a sua baixa qualidade.

Por último, em ambos datasets, as imagens foram redimensionadas para a resolução de 160x160 com adição de pixels nulos nas bordas para conservação das proporções originais de cada face e evitar distorções, como mostrado na Figura 13. Dessa forma é possível obter melhores precisões nos filtros gerados durante o treinamento por conservar o tamanho e dimensões dos olhos.

Figura 12 – Imagem com máscara inserida pelo *MaskTheFace*.

Fonte: Produzido pelo autor

Figura 13 – Imagem redimensionada com bordas laterais nulas.

Fonte: Produzido pelo autor

4.1.2 Balanceamento de classes

Como pode ser observado nas distribuições de dados após o pré-processamento, há uma grande discrepância na quantidade de imagens entre as classes, este tipo de base de dados é dita desbalanceada. Modelos de classificação são geralmente avaliados pela métrica de acurácia e o uso de bases desbalanceadas tendem a uma maior acentuação do erro causado pela priorização de classes predominantes (HOENS; CHAWLA, 2013).

Para contornar este problema existem duas principais técnicas. A primeira consiste na atribuição de pesos para diferentes classes com objetivo de aumentar a priorização de classes com menor quantidade de dados. A segunda, e mais utilizada, consiste na re-amostragem do conjunto de dados com técnicas de sobre-amostragem (*oversampling*) das classes minoritárias ou sub-amostragem (*undersampling*) das classes majoritárias. Neste trabalho, foram explorados dois algoritmos de re-amostragem: *Random Under Sampler* para sub-amostragem e *Synthetic Minority Over-sampling Technique* (SMOTE) (CHAWLA et al., 2002) para sobre-amostragem.

O *Random Under Sampler* consiste em reduzir a quantidade de dados de todas as classes de forma randômica até se igualarem a quantidade de dados da classe minoritária. Esta técnica não é recomendada para este trabalho já que deseja-se classificar muitas classes e CNN apenas são capazes de generalizar melhor com uma quantidade massiva de dados. Porém o uso desta metodologia foi utilizado apenas na base de dados real para comparação de resultados.

O SMOTE consiste na criação de dados sintéticos na vizinhança dos dados já existentes da classe minoritária. Seu funcionamento consiste em escolher aleatoriamente um dado A presente na classe minoritária atual e buscar seus k vizinhos mais próximos desta classe, então é traçado um seguimento de reta entre A e um vizinho aleatório B, por fim as instâncias sintéticas são geradas com uma combinação convexa das duas instâncias escolhidas em A e B. Esta técnica foi utilizada tanto na base de dados com imagens reais quanto na base de dados com máscaras computadorizadas.

Após a execução do *Random Under Sampler* a base de dados com imagens reais reduziu o número de imagens para 4230. Enquanto a aplicação do *Synthetic Minority Over-sampling Technique* (SMOTE) aumentou a quantidade de dados desta mesma base para 9270. Além disso, a base de dados com máscaras simuladas aumentou sua quantidade de imagens para 11160. Como a quantidade de imagens com máscara na primeira base está com um desbalanceamento pequeno, a técnica de *Synthetic Minority Over-sampling Technique* (SMOTE) foi aplicada na base completa. Enquanto na segunda base o desbalanceamento de dados entre imagens com e sem máscara era muito grande em ambos os casos, por isso a técnica de *Synthetic Minority Over-sampling Technique* (SMOTE) foi aplicada duas vezes de forma individual e depois os dados foram concatenados.

4.1.3 Aumento de dados

Imagens de faces humanas possuem uma infinidade de variações que podem limitar a capacidade de generalização de modelos CNN. Entre essas variações pode ser citado: mudanças de angulação, a forma de recorte, variações de zoom, além de mudanças na iluminação e saturação. Redes neurais convolucionais são sensíveis a estas mudanças, por isso bases com quantidades reduzidas de dados ou com poucas variações entre eles podem gerar modelos impraticáveis para o mundo real. Para reduzir estes impactos pode ser utilizado a técnica de aumento de dados.

A técnica de aumento de dados consiste em gerar novas imagens para a base a partir de transformações das imagens presentes na base. Então umas mesma imagem poderá gerar várias imagens diferentes para a rede se aplicadas transformações como espelhamento, rotação, variações do brilho, recorte e até mesmo uma combinação dessas e outras opções possíveis. Com isso, o modelo poderá prevenir overfitting por ser forçado a ajustar os filtros para as diversas possíveis variações que determinado problema pode ter. Como pode ser observado na Figura 14, neste trabalho foi utilizado a técnica de aumento de dados com as transformações de rotação, espelhamento, redimensionamento, zoom e cisalhamento.

Figura 14 – Imagens geradas pelo aumento de dados

Fonte: Produzido pelo autor

4.2 Frameworks e bibliotecas

A linguagem escolhida para desenvolvimento deste trabalho foi *Python*[1] com *Jupyter Notebook*[2]. Esta combinação é muito comum para implementação de modelos de aprendizado de máquina pela simplicidade e disponibilidade de ferramentas que auxiliem a documentação. Outro fator importante é a existência de *frameworks* robustos e uma vasta quantidade de bibliotecas disponíveis para esta linguagem.

Na geração do modelo, o *framework* escolhido foi o *TensorFlow*[3] em conjunto com as bibliotecas *Keras*, *Sklearn*, *Imblearn* e *Pillow*. Já para validação com dados reais, optou-se pela utilização dos *frameworks TensorFlow* e *PyTorch*[4] e das bibliotecas *MTCNN* e *OpenCV*. Neste caso, apenas foi utilizado mais de um *framework* porque o *PyTorch* apresentou-se mais performático no reconhecimento de faces e o *TensorFlow* foi necessário para carregamento do modelo treinado.

Com intuito de avaliar os custos de geração de um modelo para redes profundas em equipamentos do cotidiano optou-se por treinar este modelo em um notebook com CPU Intel Core i7-8750H, GPU NVIDIA GeForce GTX 1050 Ti, Memória RAM de 16 gb e SSD 480 gb. Apesar de ter sido classificado 30 classes distintas com uma base de dados grande, o treinamento apenas se tornou viável pela realização do processamento na GPU. Para isso é necessário o uso da biblioteca *NVIDIA CUDA* que já possui integração com as bibliotecas utilizadas. Além disso, a quantidade de RAM disponível não é suficiente e por isso foi reservado 35 gb para *SWAP*. O tempo para os diferente tipo de processamento pode ser visto na Tabela 3.

4.2.1 Keras

Keras[5] é uma biblioteca de código aberto robusta e de fácil implementação para redes neurais integrada ao *TensorFlow*. Por contar com uma vasta comunidade sua implementação é confiável, otimizada e recebe constates atualizações. Dentre as diversas funções disponíveis, para este trabalho foi utilizado as funções para aumento de dados (*ImageDataGenerator*), callbacks para salvar modelos (*ModelCheckpoint*) e reduzir a taxa de aprendizado (*ReduceLROnPlateau*), geração de uma matriz binária para classes (*to_categorical*), importação do modelo VGG (*VGG16*), criação do modelo com suas opções de camadas nativas (*Sequential*), otimizadores (*optimizers*) e funções para treinamento do modelo (*model*).

[1] <https://www.python.org/>
[2] <https://jupyter.org/>
[3] <https://www.tensorflow.org/>
[4] <https://pytorch.org/>
[5] <https://keras.io/>

4.2.2 Sklearn

Sklearn[6] é uma biblioteca de código aberto para *machine learning* com diversos algoritmos e utilitários de simples utilização. Para este trabalho, foi utilizada as funções para separação da base em treinamento e teste (*train_test_split*), codificação das classes em números de 0 a 29 (*LabelEncoder*) e visualização de métricas (*metrics*) para avaliação de desempenho da classificação.

4.2.3 Imblearn

Imblearn[7] é uma biblioteca de código aberto para lidar com desbalanceamento em bases de dados. Ela possui diversas opções para aplicação de sobre-amostragem e sub-amostragem na base e para este trabalho foi explorado os algoritmos sub-amostragem *Random Under Sampler* e sobre-amostragem SMOTE.

4.2.4 Pillow

Pillow[8] é uma biblioteca de código aberto em *Python* para realizar processamento de imagens. Ela foi utilizada para carregar as imagens da base de dados, redimensiona-las para 160x160 com inserção de bordas para não haver distorção e retornar um vetor de três dimensões.

4.2.5 MTCNN

MTCNN[9] é uma biblioteca de código aberto para detecção de faces em imagens e vídeos. Esta biblioteca ao receber uma imagem como entrada contendo um rosto retorna a posição da face, olhos, nariz, boca e a porcentagem de confiabilidade. Ela foi utilizada para validar o modelo treinado com imagens e vídeos que não pertencem a base.

4.2.6 OpenCV

OpenCV[10] é uma biblioteca, inicialmente desenvolvida pela Intel, com mais de 500 funções voltadas para a área de visão computacional. Em conjunto com a biblioteca *MTCNN*, ela foi utilizada para carregar e reproduzir a imagem ou video, desenhar um quadrado na imagem ou frame do video baseado na região da face detectada e exibir sua classificação.

[6] <https://scikit-learn.org/>
[7] <https://imbalanced-learn.org/stable/>
[8] <https://pillow.readthedocs.io/en/stable/>
[9] <https://github.com/ipazc/mtcnn>
[10] <https://opencv.org/>

4.3 Modelo

Após o pré-processamento apresentado na subseção 4.1.1 e o aumento de dados na subseção 4.1.3, em cada modelo a base de dados foi separada em 20% dos dados para teste e o restante para treinamento. Em seguida, foi definido os valores para os hiperparâmetros através de experimentações. Os valores que apresentaram melhor resultado podem ser vistos na Tabela 4. Entre os hiperparâmetros, pode-se destacar a necessidade de 800 épocas para melhor aproveitamento da camada de *dropout* e garantir épocas suficientes para ações dos *callbacks*.

Então foi definidos dois *callbacks*: *ModelCheckpoint* e *ReduceLROnPlateau*. O primeiro irá salvar a última época que obtiver maior valor de acurácia para a base de validação. Dessa forma, é gerando um arquivo h5 que poderá ser usado para classificar imagens desconhecidas pelo modelo. Enquanto o segundo *callback* irá reduzir a taxa de aprendizado em 0.1 a cada 100 épocas sem melhorias na taxa de acerto dos dados de validação. Esta técnica é útil para extrair ao máximo o aprendizado principalmente no final do treinamento que os ganhos são reduzidos.

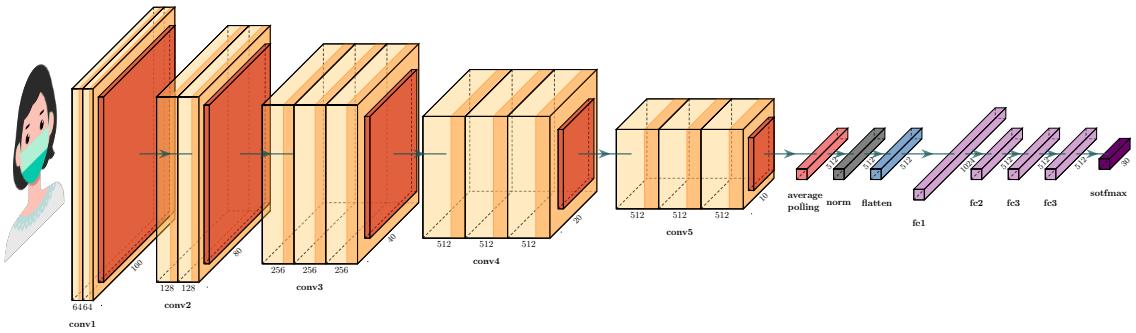
Então é carregado o modelo da *VGG16* e neste momento deve-se definir quais camadas convolucionais serão retreinadas. Como o reconhecimento de faces foi uma das classes treinadas nesta arquitetura, os filtros obtidos nas primeiras camadas são gerais e reaproveitáveis. Por conta disso, apenas a partir do quarto bloco de convoluções foi realizado o retreinamento dos filtros, já que os mesmos são mais específicos e importante para definir as diferenças entre faces humanas. Dessa forma, a rede conservará 1.735.488 parâmetros e irá retreinar 12.979.200 parâmetros. Porém, antes de realizar o treinamento, foi inserida camadas adicionais a *VGG16*.

As primeiras camadas inserida foram *GlobalAveragePooling2D* e *BatchNormalization*. A *GlobalAveragePooling2D* permite a reduzir as dimensões espaciais através da aplicação de agrupamento médio para cada dimensão, reduzindo assim a quantidade de parâmetros. Em seguida, o *BatchNormalization* é utilizado para que a saída possuía média próxima de 0 e desvio padrão próxima a 1. Dessa forma, impede que as mudanças simultâneas das diversas funções aplicadas gerem resultados inesperados que possam atrapalhar o treinamento. Ambas as camadas são capazes de aumentar a velocidade do aprendizado da rede.

Após a redução dos parâmetros e normalização, os dados de entrada são reduzidos para um vetor de dimensão única pela camada *flatten* para serem treinados por camadas totalmente conectadas. Após cada camadas totalmente conectadas (*dense*) com função de ativação ReLU foi inserido uma camada de *dropout*. A primeira camada dense foi definida com 1024 neurônios e *dropout* com taxa 0.5, seguida por 3 camadas dense de tamanho 512 e *dropout* com taxa 0.4. Então, para retornar a probabilidade de determinada entrada

corresponder a cada classe treinada, é inserida uma camada dense com função de ativação *softmax*. A estrutura geral da arquitetura pode ser observada na Figura 15.

Figura 15 – Arquitetura criada utilizando VGG16 como base



Fonte: Modificado de Iqbal (2018)

Finalmente foi realizado o treinamento da rede seguindo algumas configurações importantes para maximizar a precisão do modelo. Como deseja-se treinar uma grande quantidade de dados e classes, o treinamento foi dividido em lotes (*batch_size*) de 128 imagens a serem carregadas na memória a cada iteração de uma época. O total de iterações por época pode ser definido pela divisão do total de dados pelo tamanho do lote. Além disso, foi utilizada a função de custo para multiclasses *categorical crossentropy* e otimizador *Adam*. Todas as implementações realizadas estão disponíveis em um repositório no GitHub[11].

[11] <https://github.com/ItaloTN10/TCC2>

Tabela 2 – Classes e quantidades de imagem obtidas após pré-processamento.

Classes	**Sem máscara**	**Com máscaras reais**	**Com máscaras simuladas**
chenqiaoen	228	13	55
chenweiting	225	12	42
duhaitao	183	25	39
guanxiaotong	253	15	68
houminghao	208	17	55
huangjingyu	195	14	46
jingtian	286	15	73
linxinru	247	18	71
linyoujia	144	12	44
liyitong	182	13	30
luhan	124	21	41
masu	210	19	79
matianyu	201	19	55
wangfei	95	16	14
wangjunkai	185	17	12
wangyuan	223	17	21
wuyifan	222	25	76
xuweizhou	145	9	19
yangmi	290	18	81
yangyang	211	22	32
yangzi	193	12	53
yuanshanshan	218	21	25
zhangruoyun	221	18	42
zhangyixing	160	49	26
zhangyuxi	183	20	58
zhangzifeng	253	10	69
zhengshuang	204	7	50
zhoujielun	122	19	18
zhouxun	256	12	82
zhouyumin	157	13	50

Fonte – Produzido pelos autores.

Tabela 3 – Comparação de tempo de treinamento entre CPU e GPU com *batch size* de 128 e 800 épocas

Processamento	**Tempo Médio por Época**	**Tempo Total**
CPU	736s	$\approx$ 7 dias
GPU	60s	13h56

Fonte – Produzido pelo autor.

Tabela 4 – Valores dos Hiperparâmetros para os modelos.

Hiperparâmetro	**Valor**
batch_size	128
input_shape	(160, 160, 3)
random_state	42
alpha	1e-5
epoch	800

Fonte – Produzido pelo autor.

5 Resultados

Neste capítulo serão apresentados os resultados obtidos para o modelos criado no Capítulo 4. Na Seção 5.1 será apresentado os testes realizados. Seguido da Seção 5.2 que apresentará resultados dos testes. Enquanto na Seção 5.3 será apresentado o comportamento do treinamento. Por fim, a Seção 5.4 irá demonstrar outras formas de comprovação dos resultados.

5.1 Testes

O modelo final foi definido por experimentação de algumas variações nos dados de entrada, mudanças nos hiperparâmetros, camadas retreinadas da *VGG16* e números de camadas totalmente conectadas. A começar pelas entradas foram separadas em duas bases, uma com imagens reais de máscara e outra com máscaras inseridas computacionalmente. A seguir será apresentado uma visão geral das experimentações realizadas e nas seções seguintes será apresentado os resultados.

O primeiro passo nas experimentações foi a variação do número de épocas e o valor *patience* para o *callback ReduceLROnPlateau*. Em uma rede com número elevado de parâmetros a serem ajustados, uma quantidade reduzida de épocas não irá permitir que o *dropout* atue no aumento da capacidade de generalização da rede. Além disso, o valor de *patience* quando muito pequeno irá reduzir a taxa de aprendizado prematuramente, impedindo que a precisão atinja seu valor máximo. Porém um valor muito alto para o *patience* exigirá proporcionalmente um aumento no número de épocas e consequentemente aumentar o tempo necessário para treinamento.

Tanto o número de épocas quanto o *patience* dependem do número e quantidade de neurônios das camadas totalmente conectados. Quanto menor esses parâmetros, menor o número de pesos a serem ajustados, então da mesma forma que a rede poderá ganhar em tempo de execução, poderá perder em capacidade de representação de características. Então foi ajustado o número de camadas a fim de maximizar a precisão e reduzir o tempo de treinamento. Esta quantidade de camadas e neurônios está diretamente ligadas também ao número de camadas a serem retreinadas no aprendizado por transferência.

Ao analisar a arquitetura da *VGG16* é possível perceber que quanto mais profunda é a rede, mais parâmetros específicos são passados para a convolução. Ao optar por retreinar todas as camadas ou as primeiras camadas, a rede terá que aprender diversas características globais limitados a base de dados utilizada. Com isso, o uso de mais pesos da *VGG16* permite reaproveitar filtros que todas as faces tem em comum e reduzir o custo

de treinamento da rede. Porém como deseja-se diferenciar diversas faces, as camadas mais específicas devem ser retreinadas.

Enfim, a última variação realizada foi a validação das opções de balanceamento da base comparadas ao não balanceamento. No balanceamento foi explorados os métodos *RandomUnderSampler* e SMOTE. Será possível observar na seção 5.2 que uso da técnica de sub-amostragem irá ser pior do que não balancear a base, isto porque será perdido muitas imagens. Já o uso da sobre-amostragem terá resultados significativamente superiores em relação aos testes anteriores.

5.2 Precisão dos modelos

Como poderá ser observado na Tabela 5, fica evidente a dependência das redes neurais convolucionais no aumento da quantidades de dados. O modelo 2 apresentou melhores resultados mesmo com máscaras geradas artificialmente por aumentar consideravelmente o número de imagens com essa característica. Além disso, em todos os casos a técnica de SMOTE foi crucial para obter resultados satisfatórios pelas limitações da base. É possível perceber que a não utilização de uma técnica de sobre-amostragem dos dados resultou em resultados insatisfatórios, por isso optou-se por apresentar apenas o melhor resultado para a base com dados artificiais. Como os modelos com SMOTE foram os únicos que apresentaram resultados satisfatórios, apenas eles serão considerados nas discussões seguintes.

Tabela 5 – Resultados obtidos nos treinamentos das bases de dados com máscaras reais e simuladas.

Base de dados	**Balanceamento**	**Acurácia**	**Total de imagens**	**Tempo de execução**
Real	-	82,75%	6898	12h44
Real	*RandomUnderSampler*	78,01%	4230	6h10
Real	SMOTE	91,94%	9240	13h56
Máscara Simulada	SMOTE	93,54%	11160	16h13

Fonte – Produzido pelo autor.

A Tabela 6 mostra um baixo valor no desvio padrão da acurácia do conjunto de teste para cinco treinamentos distintos. Já na Figura 16 e Figura 17 é apresentada as matrizes de confusão para as bases com imagens reais e geradas artificialmente, em que a ordem dos números correspondem a ordem da Tabela 2. É possível perceber a complexidade do problema de classificação de imagens com elevado número de classes, mas fica evidenciado que o modelo acertou a grande maioria dos nomes das pessoas. O grande problema é uma determinada classe poder apresentar pequenas semelhanças nas

imagens fornecidas com outras 29 classes e esse erro ser distribuído em pequenas porções ao longo da matriz, dessa forma o número de acertos é reduzido drasticamente.

Tabela 6 – Todos os resultados nas bases de dados com SMOTE de máscaras reais e simuladas.

Base de dados	**Acurácias para base de teste**					**Média**	**Desvio padrão**
Real	90,99%	91,29%	91,39%	91,77%	**91,94%**	91,47%	0,38%
Máscara Simulada	92,60%	92,65%	93,23%	93,28%	**93,54%**	93,06%	0.41%

Fonte – Produzido pelo autor.

5.3 Análise do treinamento

Nas Figura 18 e Figura 19 é possível observar quedas bruscas de acurácia para os dados de teste. Essas quedas ocorrem pela ação do *dropout* em desativar alguns neurônios da camada oculta, evidenciando neurônios que podem ser melhor ajustados. Dessa forma, ao longo do treinamento a rede será capaz de generalizar melhor se todos os neurônios tiverem menores dependências entre si. Outra informação a ser observada é o efeito da redução da taxa de aprendizado que ocorreram nas épocas 721 e 468 para as Figura 18 e Figura 19 respectivamente. Depois de algumas épocas a rede não conseguia convergir até a redução da taxa de aprendizado ocorrer.

5.4 Validação dos resultados

Como a base *RMFD* foi inicialmente pensada para reconhecimento em tempo real, foi criado um script em *Python* para validação de imagens e vídeos externos a base. Seu funcionamento consiste na extração das faces da imagem ou video através da biblioteca *MTCNN* e desenho da região classificada em conjunto com sua classificação com a biblioteca *OpenCV*. Com este teste, em vídeos fica evidente que apesar do modelo ser capaz de classificar a maioria dos casos, ele também erra principalmente em mudanças de angulação ou variações na iluminação. Nas Figura 20 e Figura 21 será possível observar estes testes.

5.5 Comparação com outros resultados

Como a base de dados proposta foi criada durante a pandemia, a maioria dos trabalhos buscam identificar a presença da máscara nas imagens. Por esse motivo, o trabalho de Hariri (2021) foi o único encontrado que se propõem em classificar multiclasses

da base RMFD. A principal diferença entre os modelos está no número de classes, já que o trabalho atual classifica 30 classes e Hariri classifica 50, 60, 70 e 100 classes.

Este trabalho optou por classificar um número menor de classes pela limitação da base em número de imagens de pessoas com máscara. Enquanto o trabalho de Hariri contornou isso com o alinhamento dos olhos com as bordas horizontais para em seguida recortar a imagem de forma que a máscara seja removida antes do treinamento. A abordagem proposta por Hariri é interessante porém aumenta o custo para aplicação do modelo no mundo real, pois exigirá a aplicação desse pré-processamento antes da classificação, algo que poderá tornar a identificação de pessoas através de vídeos inviável.

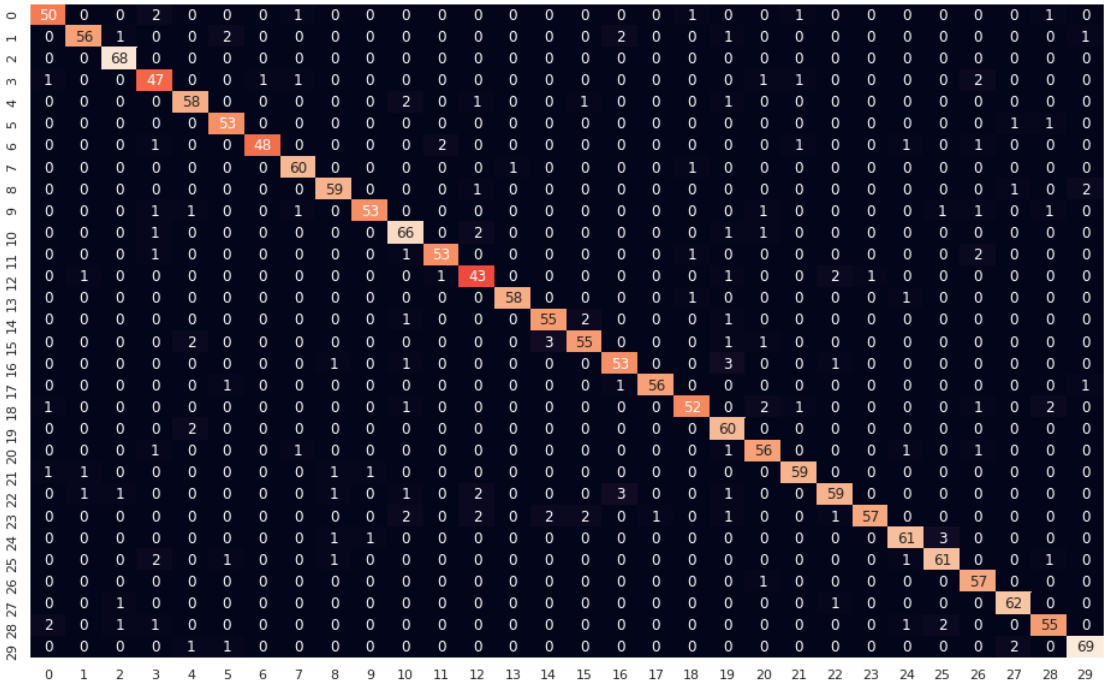
Como é possível perceber na Tabela 7, foram obtidos melhores resultados em ambos modelos (com máscaras reais e simuladas), com destaque para a base de dados com máscaras simuladas. Outro ponto a ser considerado é que o trabalho de Hariri mostra que o aumento de 50 para 60 classes trouxe melhores resultados e depois essa acurácia reduziu novamente, isso sugere que a escolha das classes e a qualidade das imagens teve mais impacto que o aumento de classes.

Tabela 7 – Comparação dos resultados entre modelos com máscaras reais e simuladas.

Metologia	**Número de Classes**	**Acurácia (Real)**	**Acurácia (Simulada)**
Este trabalho	30	**91,94%**	**93,54%**
Hariri (2021)	50	91,0%	83,5%
Hariri (2021)	60	91,3%	84,7%
Hariri (2021)	70	90,1%	88,9%
Hariri (2021)	100	89,8%	88,5%

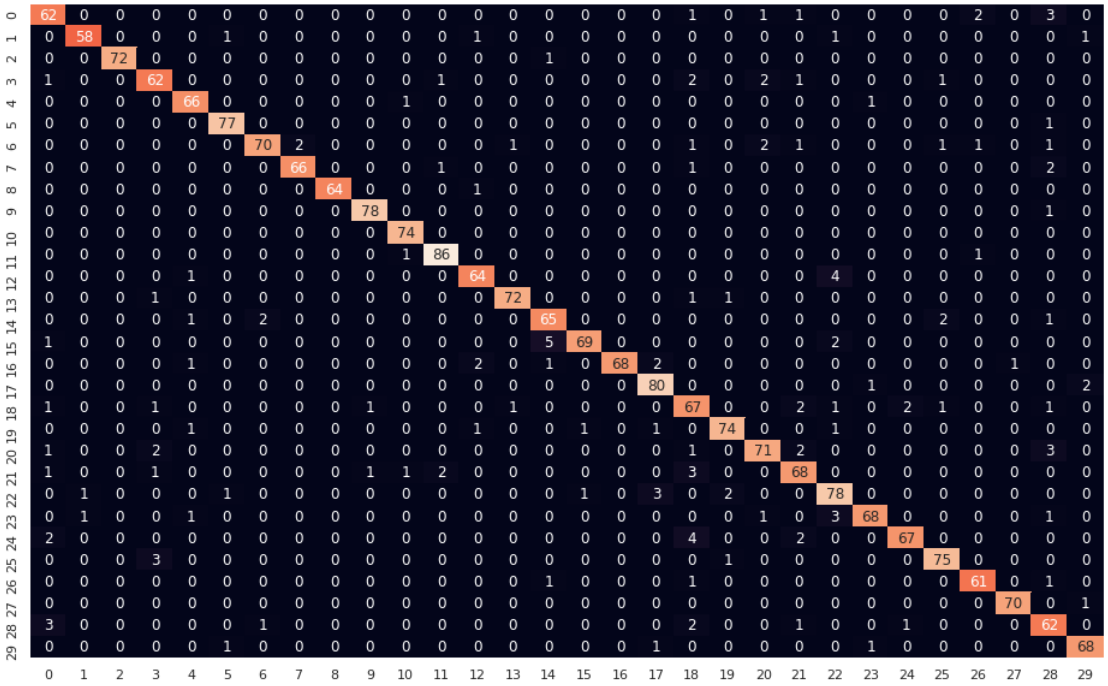
Fonte – Produzido pelo autor.

Figura 16 – Mapa de calor da matriz de confusão para base de imagens reais e SMOTE



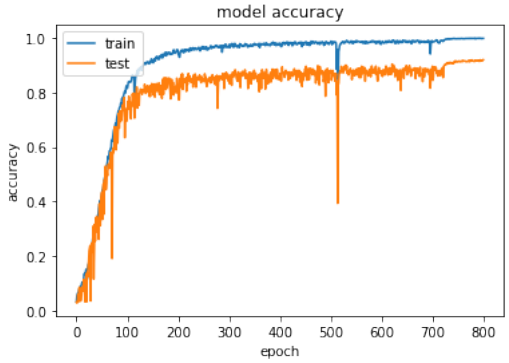
Fonte: Produzido pelo autor

Figura 17 – Mapa de calor da matriz de confusão para base com imagens artificiais e SMOTE



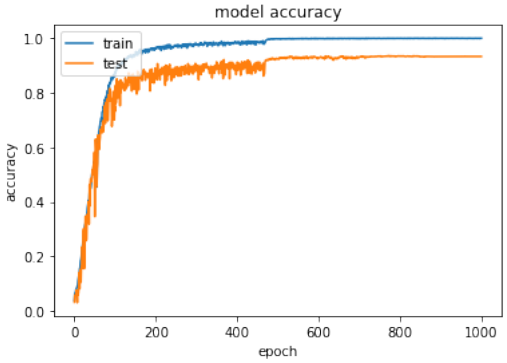
Fonte: Produzido pelo autor

Figura 18 – Gráfico de acurácia para base de imagens reais e SMOTE



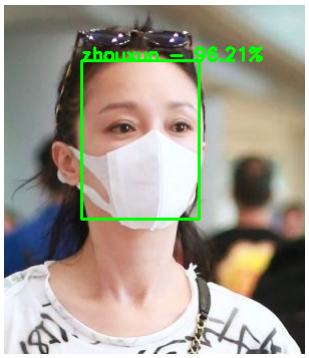
Fonte: Produzido pelo autor

Figura 19 – Gráfico de acurácia para base com imagens artificiais e SMOTE



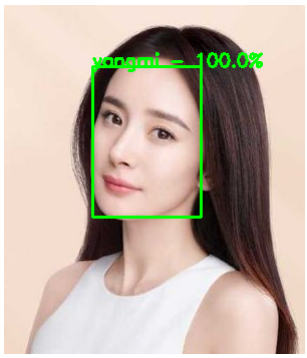
Fonte: Produzido pelo autor

Figura 20 – Detecção do rosto de Zhou Xun com o modelo de máscaras simuladas.



Fonte: Produzido pelo autor

Figura 21 – Detecção do rosto de Yang Mi com o modelo de máscaras simuladas.



Fonte: Produzido pelo autor

6 Conclusão

Este trabalho apresentou técnicas de pré-processamento de dados aplicadas a arquiteturas robustas e de simples implementação para avaliar a capacidade de reconhecimento facial em indivíduos com máscara. Foram classificadas 30 classes de nomes de indivíduos distribuídas em imagens de faces com e sem máscaras. O modelo foi criado utilizando redes neurais convolucionais com aprendizado por transferência da arquitetura *VGG16*. Além disso, a base de dados foi separada em um conjunto de imagens de máscaras reais e outro de máscaras simuladas computacionalmente. Em cada um destes conjuntos de imagens foram aplicadas técnicas de *oversampling* para balanceamento da base de dados *RMFD*.

Um ponto chave a ser destacado é a importância do pré-processamento para qualquer problema de rede neurais, já que na base haviam diversas imagens com o rosto completamente coberto ou até mesmo imagens que não tinham relação com o trabalho. Além disso, foi evidenciado a dependência de uma grande quantidade de dados para que as redes neurais convolucionais sejam capazes de diferenciar um grande número de classes e generalizar para cenários do mundo real. Por isso, técnicas de aumento de dados e balanceamento com SMOTE foram essenciais.

Em um problema de redes convolucionais profundas com multiclasses, os resultados obtidos foram capazes de classificar a grande maioria das imagens e ser coerente com o estado da arte atual. Porém quando aplicado para um problema de segurança real, a presença de máscaras ainda pode ser considerada um problema a ser mais explorado. O principal ponto de partida é a utilização de uma base com dados com menos ruídos e maior balanceamento de dados entre as classes. Como alternativa, o uso do script *MaskTheFace* se tornou uma alternativa viável para adaptar bases já existentes, entretanto ele apresenta algumas dificuldades em reconhecer algumas faces e por isso a quantidade de imagens que ele foi capaz de gerar foi reduzida.

Em trabalhos futuros será proposto a geração de uma base de dados melhor balanceada e em cenários mais controlados. Além disso, deseja-se comparar os resultados obtidos em redes neurais convolucionais com modelos que utilizam redes em cápsula (CapsNet) (SABOUR; FROSST; HINTON, 2017), a fim de reduzir a necessidade de grandes quantidades de dados para treinamento.

Referências

ANWAR, A.; RAYCHOWDHURY, A. *Masked Face Recognition for Secure Authentication*. 2020. Citado na página 29.

BLEDSOE, W. W. The model method in facial recognition. *Panoramic Research Inc., Palo Alto, CA, Rep. PR1*, v. 15, n. 47, p. 2, 1966. Citado na página 17.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado na página 30.

CHENG, V. C.-C. et al. The role of community-wide wearing of face mask for control of coronavirus disease 2019 (covid-19) epidemic due to sars-cov-2. *Journal of Infection*, Elsevier, v. 81, n. 1, p. 107–114, 2020. Citado na página 14.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1251–1258. Citado na página 26.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255. Citado na página 17.

GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1. Citado na página 23.

HACKELING, G. *Mastering Machine Learning with scikit-learn*. [S.l.]: Packt Publishing Ltd, 2017. Citado na página 21.

HARIRI, W. Efficient masked face recognition method during the covid-19 pandemic. *arXiv preprint arXiv:2105.03026*, 2021. Citado 3 vezes nas páginas 18, 40 e 41.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 26.

HOENS, T. R.; CHAWLA, N. V. Imbalanced datasets: from sampling to classifiers. *Imbalanced learning: Foundations, algorithms, and applications*, Wiley Online Library, p. 43–59, 2013. Citado na página 30.

IQBAL, H. *HarisIqbal88/PlotNeuralNet v1.0.0*. 2018. Disponível em: <https://doi.org/10.5281/zenodo.2526396>. Citado na página 35.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 22.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, p. 1097–1105, 2012. Citado 2 vezes nas páginas 17 e 18.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 22.

LI JUSTIN JOHNSON, S. Y. F.-F. *Standford - Deep Learning Lectures: CNN architectures*. 2017. Disponível em: <http://cs231n.stanford.edu/slides/2017/>. Citado na página 27.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 19.

MELO, A. *Data Science e Machine Learning na Prática - Introdução e Aplicações na Indústria de Processos*. 2020. Disponível em: <https://www.kaggle.com/afrniomelo/epv-peq-aula-3-classifica-o>. Citado na página 25.

NGAN, M. L.; GROTHER, P. J.; HANAOKA, K. K. Ongoing face recognition vendor test (frvt) part 6a: Face recognition accuracy with masks using pre-covid-19 algorithms. 2020. Citado na página 14.

OKEREAFOR, K. et al. Fingerprint biometric system hygiene and the risk of covid-19 transmission. *JMIR Biomedical Engineering*, JMIR Publications Inc., Toronto, Canada, v. 5, n. 1, p. e19623, 2020. Citado na página 14.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2009. Citado na página 26.

PHILLIPS, P. J. et al. An introduction evaluating biometric systems. *Computer*, IEEE, v. 33, n. 2, p. 56–63, 2000. Citado na página 17.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 20.

SABOUR, S.; FROSST, N.; HINTON, G. E. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017. Citado na página 44.

SILVA, S. R.; SCHIMIDT, F. Redução de variáveis de entrada de redes neurais artificiais a partir de dados de análise de componentes principais na modelagem de oxigênio dissolvido. *Química Nova*, SciELO Brasil, v. 39, p. 273–278, 2016. Citado na página 20.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 27.

SIROVICH, L.; KIRBY, M. Low-dimensional procedure for the characterization of human faces. *Josa a*, Optical Society of America, v. 4, n. 3, p. 519–524, 1987. Citado na página 17.

SOBREIRO, V. A. et al. Uma estimação do valor da commodity de açúcar utilizando redes neurais artificiais. *Pesquisa & Desenvolvimento em Engenharia de Produção*, v. 7, p. 36–52, 2008. Citado na página 21.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Citado 2 vezes nas páginas 25 e 26.

SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9. Citado na página 26.

TAIGMAN, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 1701–1708. Citado na página 17.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. *Journal of cognitive neuroscience*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 3, n. 1, p. 71–86, 1991. Citado na página 17.

WANG, Z. et al. Masked face recognition dataset and application. *arXiv preprint arXiv:2003.09093*, 2020. Citado na página 28.

ZHANG, A. et al. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021. Citado na página 24.