PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

REDES NEURAIS CONVOLUCIONAIS

Análise De Imagens De Vídeo Em Tempo Real Para Identificação De Faces E Verificação Do Uso De Máscaras De Proteção

ANDRÉ LUIZ ASTROL DE OLIVEIRA

GOIÂNIA
2021

ANDRÉ LUIZ ASTROL DE OLIVEIRA

REDES NEURAIS CONVOLUCIONAIS

Análise De Imagens De Vídeo Em Tempo Real Para Identificação De Faces E Verificação Do Uso De Máscaras De Proteção

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência de Computação.

Orientador:

Prof. Me. Max Gontijo de Oliveira

GOIÂNIA

2021

ANDRÉ LUIZ ASTROL DE OLIVEIRA

REDES NEURAIS CONVOLUCIONAIS

Análise De Imagens De Vídeo Em Tempo Real Para Identificação De Faces E Verificação Do Uso De Máscaras De Proteção

Este Trabalho de Conclusão de Curso julgado adequado para obtenção o título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, em ____/____/_____.

_____

Profa. Ma. Ludmilla Reis Pinheiro dos Santos

Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientador: Prof. Me. Max Gontijo de Oliveira

_____

Profa. Ma. Lucília Ribeiro

_____

Prof. Me. Fernando Gonçalves Abadia

,

GOIÂNIA

2021

Dedico este trabalho a minha família, parentes e amigos que sempre me apoiaram com meus estudos e me amaram incondicionalmente.

AGRADECIMENTOS

Agradeço aos meus pais, Maria Cristina e Wandir Gonçalves, e ao meu irmão Miguel Enrique, pela confiança no meu progresso e pelo apoio ao longo do curso.

Ao meu orientador Max Gontijo, embora sua vida acadêmica seja intensa, ele concordou em me orientar nesta monografia. Suas valiosas recomendações fizeram toda a diferença.

Agradeço a todos os professores e funcionários da Pontifícia Universidade Católica de Goiás, que sempre transmitiram seus conhecimentos com grande profissionalismo e empenho.

Agradeço também a todos os meus amigos do curso, por terem me dado a oportunidade de conviver e colaborar ao longo de todos esses anos.

“Em momentos de crise, não se desespere,
aproveite, saiba tirar vantagem”
(Kratos)

RESUMO

Este trabalho tem como objetivo desenvolver um modelo computacional para identificar faces e classificar se elas estão fazendo o uso de máscaras de proteção. Com o surgimento da Covid-19, métodos alternativos para a minimização das infecções causadas pelo coronavírus vêm sendo amplamente utilizados ao redor do mundo, tais como utilização de luvas, disponibilização de álcool em gel em ambientes públicos e privados, distanciamento social e obrigatoriedade do uso das essenciais máscaras de proteção respiratória. Juntamente a esses métodos, tecnologias podem desempenhar um papel no combate a essa doença, reportando quais indivíduos estão ou não cumprindo seu papel perante as leis de proteção. Sendo assim, com a utilização de uma Rede Neural Convolucional (CNN), o trabalho em questão visa desenvolver um modelo para classificar se os indivíduos capturados em imagens de vídeo estão ou não fazendo o uso de máscaras protetoras, dando liberdade e flexibilidade para que o modelo desenvolvido seja utilizado em trabalhos futuros.

Palavras-chave: Identificação de Faces. Rede Neural Convolucional. Máscara de Proteção

ABSTRACT

This work aims to develop a computational model to identify faces and classify they are using protective masks. With the emergence of Covid-19, alternative methods for minimizing those caused by the coronavirus being widely used around the world, such as the use of gloves, availability of alcohol gel in public and private environments, social distance and mandatory use of essentials respiratory protection masks. Along with these methods, technologies can play a role in fighting this disease, reporting which ones are not or fulfilling their role under protective laws. Thus, with the use of a Convolutional Neural Network (CNN), the work in question is to develop a model to classify whether those captured in video images are or are not using protective masks, giving freedom and flexibility for the model developed is used in future work.

Keywords: Face Identification. Convolutional Neural Network. Protection Mask
.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
RNA	Rede Neural Artificial

SUMÁRIO

1. INTRODUÇÃO

No final de 2019, a humanidade se deparou com o anúncio de um novo vírus com potencial pandêmico, nomeado como SARS-CoV-2, responsável pela COVID-19. Ainda no primeiro semestre de 2021, o Brasil contabilizava cerca de 15,6 milhões de casos registrados e aproximadamente 436 mil vítimas fatais.

O uso de máscaras faciais passou a ser adotado como uma medida de contingência da disseminação do vírus. Segundo estudos (G1 RS, 2021) realizados por pesquisadores da Universidade Federal do Rio Grande do Sul (UFRGS), da Universidade das Ciências da Saúde (UFCSPA), da Unisinos e da Universidade Federal de Pelotas (UFPel), o uso de máscaras reduz em até 87% as chances de infecção por coronavírus. Infectologistas também ressaltaram a importância de saber utilizar as máscaras de forma adequada, devendo tapar o nariz e a boca por completo e não deixando vazar ar.

Apesar do sancionamento da Lei nº 14.019/2020 que torna obrigatório o uso de máscara em ambientes públicos e privados, as autoridades governamentais ainda possuem bastante dificuldade em fazer o controle dos indivíduos que ferem a mencionada lei. Mesmo com alguns estados decretando multa para cidadãos ou estabelecimentos que descumprirem a lei, a adesão na utilização de máscaras em espaços comuns vem cada vez mais diminuindo. Neste contexto, sistemas de reconhecimento eficientes para verificar se os rostos das pessoas estão mascarados de forma segura e aceitável seriam de grande apoio às autoridades para assegurar o cumprimento dessa lei.

A proposta do presente trabalho é desenvolver um modelo que associado à um *software*, realize a detecção de faces e classifique se estão ou não utilizando máscaras. Caso essa solução seja aplicada em um ambiente de teste real, poderá ser adaptada em um circuito de câmeras de monitoramento e até mesmo integrada em um serviço de mensageria para reportar quais indivíduos do recinto estão infringindo as regras estabelecidas. Em uma escala maior, poderia ser integrada a um sistema do governo, onde detectaria a face, classificaria se está ou não de máscara e logo em seguida, emitiria uma multa automaticamente para o infrator.

Para que o principal objetivo do trabalho seja alcançado, estudos e pesquisas sobre Redes Neurais Convolucionais, OpenCV e Keras foram demandados.

1.1 METODOLOGIA ADOTADA

Quanto a natureza deste projeto, pode-se classificá-lo como pesquisa aplicada, pois tem como objetivo gerar conhecimento para aplicações práticas de problemas específicos.

No que diz respeito aos objetivos do projeto, a pesquisa exploratória se encaixa melhor como ferramenta do trabalho. Apesar de não buscar descrever um tipo novo de conhecimento, o projeto tem como finalidade aplicar na prática os fundamentos teóricos que foram adquiridos em artigos e trabalhos acadêmicos já desenvolvidos na área.

Referente aos procedimentos técnicos, este projeto define-se como pesquisa bibliográfica e documental.

Logo de início foram realizados estudos de teses, artigos, livros, etc. E com as informações obtidas com estes estudos foi possível organizar uma base bibliográfica para contribuir com o projeto, o que é definido como pesquisa bibliográfica.

1.2 ORGANIZAÇÃO DOS CAPÍTULOS

No primeiro capítulo foram introduzidas as inspirações e motivações para realização desse trabalho, assim como também as tecnologias e possíveis melhoras que poderão ser realizadas futuramente.

O Capítulo 2 proporciona fundamentação teórica para que os objetivos deste trabalho sejam alcançados, apresentando conceitos de perceptron, perceptron multicamadas e arquitetura de redes convolucionais.

O Capítulo 3 irá apresentar as tecnologias que foram utilizadas no decorrer do desenvolvimento do trabalho. Além disso, será realizada uma explanação sobre os *datasets* que serviram de base para criação dos *datasets* principais.

O Capítulo 4 explicará quais foram os passos necessários para o desenvolvimento da solução proposta pelo trabalho.

Por fim, o Capítulo 5 será responsável por apresentar as conclusões finais do trabalho.

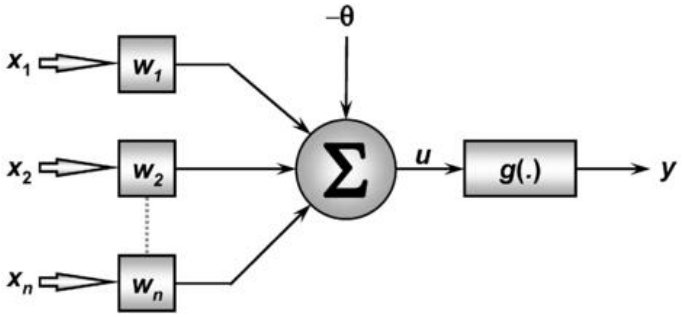
2. REFERENCIAL TEÓRICO

Neste capítulo, uma base teórica será apresentada para melhor compreensão dos tópicos abordados na pesquisa. Os fundamentos sobre *perceptron*, *perceptron* multicamadas e redes neurais convolucionais são extremamente relevantes para o desenvolvimento desse projeto.

2.1 PERCEPTRON

Redes neurais artificiais (RNAs) possuem um modelo computacional baseado no sistema nervoso dos organismos vivos, tendo como principal objetivo adquirir e manter conhecimento por meio de treinamento e experiência. O nome RNA vem de sua inspiração no sistema nervoso biológico, e na prática o modelo traz um conceito de neurônios artificiais conectados entre si. As RNAs apresentam como principais características a capacidade de aprendizagem por meio de treinamento, adaptação da experiência adquirida (ajustando os pesos das sinapses com base na entrada de informações) e a generalização de informações treinadas pelo modelo.

Figura 1 - Modelo de um Neurônio Artificial



Fonte: (Lama, 2016)

A imagem apresentada na Figura 1 ilustra um tipo de RNA chamada de *perceptron*. Uma RNA é um sistema interconectado de *perceptrons*. Sendo assim, é possível afirmar que os *perceptrons* são a base de qualquer rede neural. O *perceptron* pode ser visto como um bloco de camada única em uma rede neural, consistindo em quatro partes diferentes, sendo elas: valores de entrada, pesos e bias, célula somatória e função de ativação.

Tomando como base o exemplo na Figura 1, x_1, x_2....x_n representam os valores de entrada da rede. Entre os valores de entrada e a célula de somatório existem as chamadas sinapses, onde cada uma carrega um determinado peso w_i, que por sua vez é multiplicado por seu respectivo valor de entrada e o produto dessa multiplicação é agregado ao somatório. O resultado do somatório pode ser acrescido ou decrescido do bias, que tem por finalidade tornar o modelo mais flexível e adaptável.

Por fim, o valor adquirido é processado em uma função de ativação que tem como objetivo limitar o valor de saída do neurônio, fazendo com que ele corresponda às entradas da rede. A saída gerada pela função de ativação é classificada como correta ou não. Caso o resultado seja o esperado, os próximos dados de entrada são processados na rede, caso contrário, os pesos da rede são reajustados e as novas entradas são processadas. Todo o processo acima pode ser representado na Equação 1.

Equação 1 - Formulação do Perceptron

$$y = \begin{cases} 1, se \sum_j w_j x_j - \theta > 0 \\ 0, se \sum_j w_j x_j - \theta \leq 0 \end{cases}$$

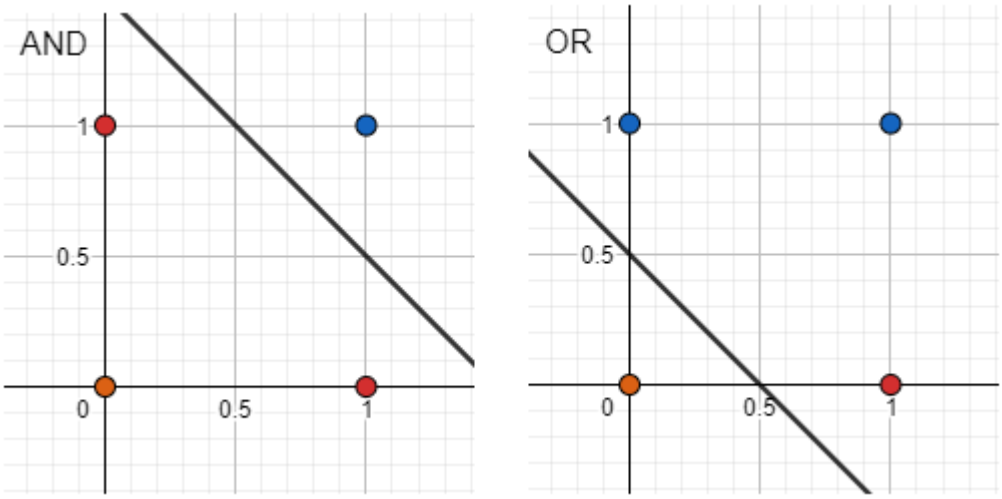
Fonte: Próprio Autor

onde:

x_j representa a entrada;
w_j representa o peso;
Θ representa o bias;
y representa a saída.

O problema do *perceptron* é a sua limitação em conseguir resolver apenas problema linearmente separáveis, como os operadores lógicos AND e OR ilustrados na Figura 2:

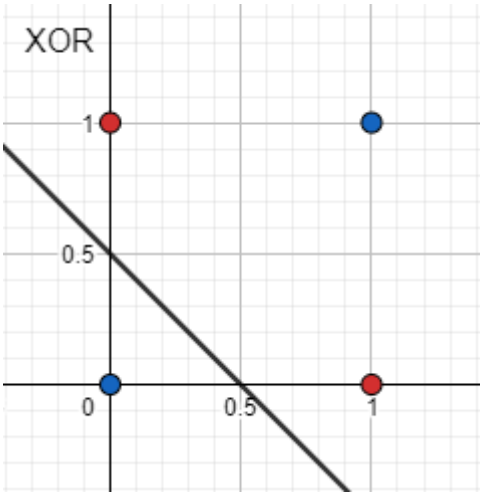
Figura 2 - Problemas Linearmente Separáveis



Fonte: Próprio Autor

Um exemplo de problema que o *perceptron* não consegue resolver é o do operador XOR:

Figura 3 - Problemas Linearmente Separáveis



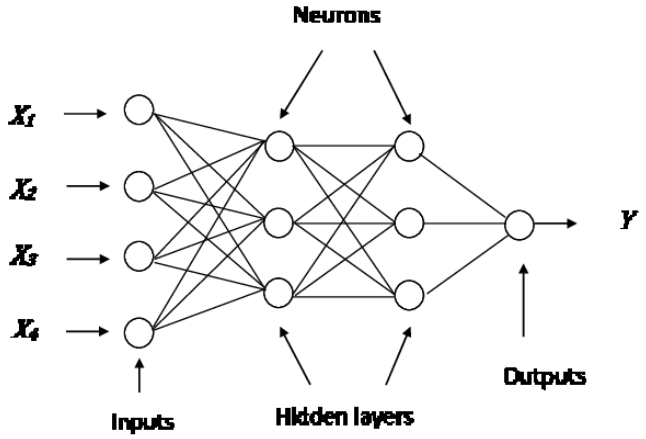
Fonte: Próprio Autor

Problemas dessa natureza requerem uma arquitetura mais sofisticada para serem resolvidos. Uma abordagem mais robusta e versátil é o uso de múltiplos *perceptrons* para a realização do aprendizado.

2.2 PERCEPTRON MULTICAMADAS (MLP)

Uma MLP é uma rede neural artificial *feedforward* (sentido único) que gera um conjunto de resultados a partir de um conjunto de entradas. O MLP é definido por nós de entrada que se conectam com uma série de nós das chamadas camadas escondidas, onde essas, por sua vez, podem abstrair em inúmeros níveis as informações inseridas. No final das camadas escondidas existe uma camada de saída, que tem por função exibir o resultado do processamento. Nesse modelo de rede os neurônios de uma camada se conectam com todos os neurônios da camada seguinte, dessa forma caracterizando uma rede densamente conectada, trazendo assim uma exigência maior de processamento para o treinamento do modelo.

Figura 4 - Topologia de MLP



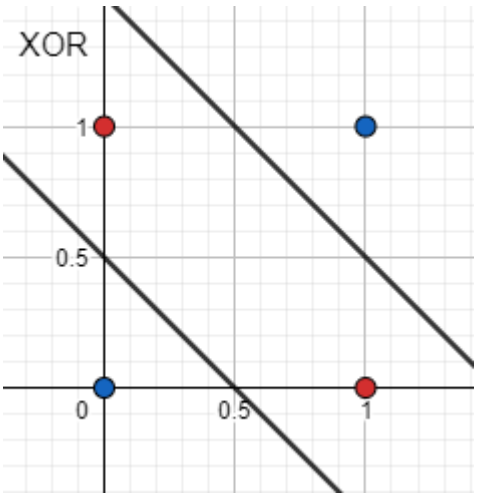
Fonte: (Mathur, 2016)

O treinamento desse tipo de rede é realizado utilizando o algoritmo *backpropagation*, visando encontrar o mínimo de erro possível na relação de entrada com saída. Resumidamente, a rede neural compara o valor da camada de saída com o resultado esperado, calcula o erro com base na diferença entre os valores e distribui a correção entre as sinapses dos nós adjacentes. O erro encontrado ajuda a recalcular os pesos sinápticos e consequentemente minimizar as chances de discrepâncias maiores nas próximas execuções, procurando também o melhor ajuste de pesos para relacionar todas as entradas com os resultados esperados.

Como mencionado na seção anterior, MLPs foram desenvolvidas para resolver problemas como o do operador lógico XOR, visto que uma rede *perceptron*

é capaz de criar apenas um hiperplano separador para dividir os dados presentes em um espaço. Em MLPs, cada neurônio presente na camada escondida é responsável por criar um hiperplano separador, logo, para o problema do XOR o resultado ótimo seria semelhante ao da Figura 5.

Figura 5 - Separação por Hiperplanos (XOR)



Fonte: Próprio Autor

2.3 O PROBLEMA

O reconhecimento de imagens é um problema clássico de visão computacional, e as chamadas Redes Neurais Convolucionais (CNN) ganharam destaque para resolução de problemas dessa área. Resumidamente, CNN é um algoritmo de aprendizado profundo que recebe uma imagem e atribui pesos às suas principais características. Tem como principal diferencial o baixo custo exigido em seu processamento e sua alta acurácia na classificação de um objeto-alvo.

Diferente de outros algoritmos classificadores, ConvNets (Redes Convolucionais) não visam reconhecer uma imagem por inteiro, mas apenas suas características fundamentais. Logo, isso justifica sua baixa exigência de processamento no treinamento de modelos. Além disso, os processos executados em uma rede convolucional também trazem como benefício a exclusão de alguns problemas comumente encontrados em outros tipos de redes, como: translação, escala e rotação.

Imagine uma rede neural treinada para classificar o elemento da Figura 6. Caso o objeto-alvo (sinal de X) por algum motivo seja transladado, redimensionado ou rotacionado, a probabilidade dessa rede classificar o elemento como um “X” cai consideravelmente.

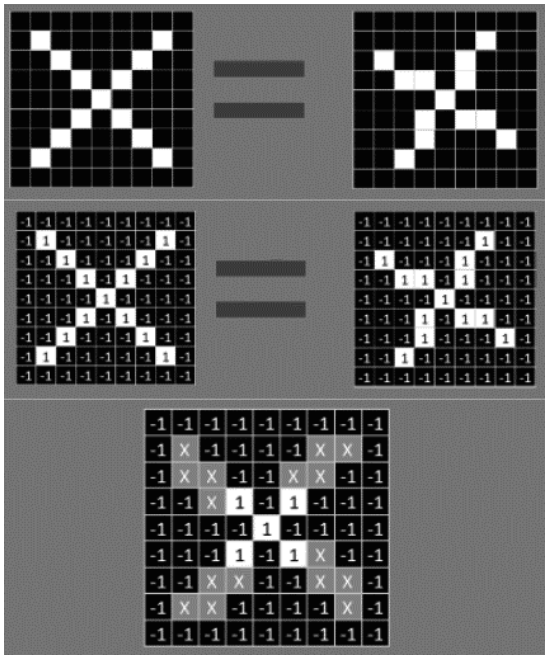
Figura 6 - Problema de Classificação

Fonte: (Soares, 2018)

A Figura 7 representa um dos clássicos problemas mencionados acima. Apesar de ambos os elementos apresentarem a mesma informação, o elemento mais à direita está levemente rotacionado para esquerda, fazendo com que a rede os identifique como objetos diferentes.

Um exemplo disso é o último elemento da Figura 7, onde um comparativo entre as paridades dos pixels é realizado, assim, ficando visível que a quantidade de pixels distintos acaba sendo bem maior.

Figura 7 - Comparação Rotacionada

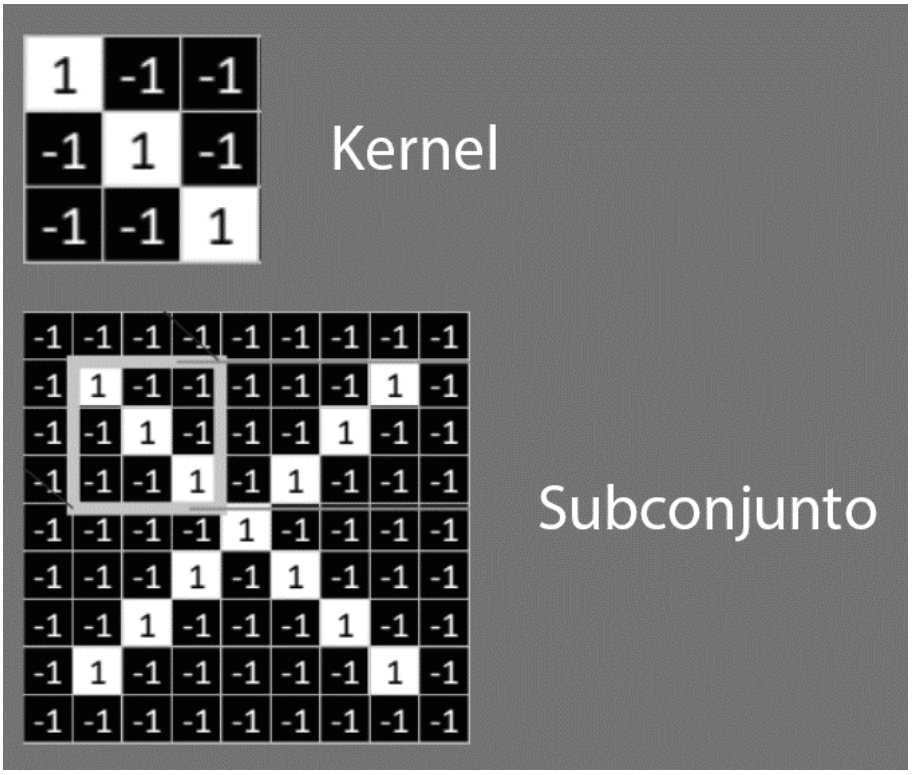


Fonte: (Soares, 2018)

2.4 CONVOLUÇÃO

Por definição, convolução é uma operação matemática que tem por objetivo convoluir duas funções, assim, obtendo como resultado uma terceira função, onde normalmente é vista como uma versão modificada de uma das funções originais. Todo esse processo tem como único objetivo destacar informações relevantes em um determinado conjunto de imagens. De forma menos abstrata, imagine que a primeira função é um subconjunto da imagem de treino (também conhecido como campo receptivo local) e a segunda função é um filtro (comumente chamado de *kernel*) que realça os principais pontos de tal imagem.

Figura 8 - Kernel e Subconjunto



Fonte: (Soares, 2018)

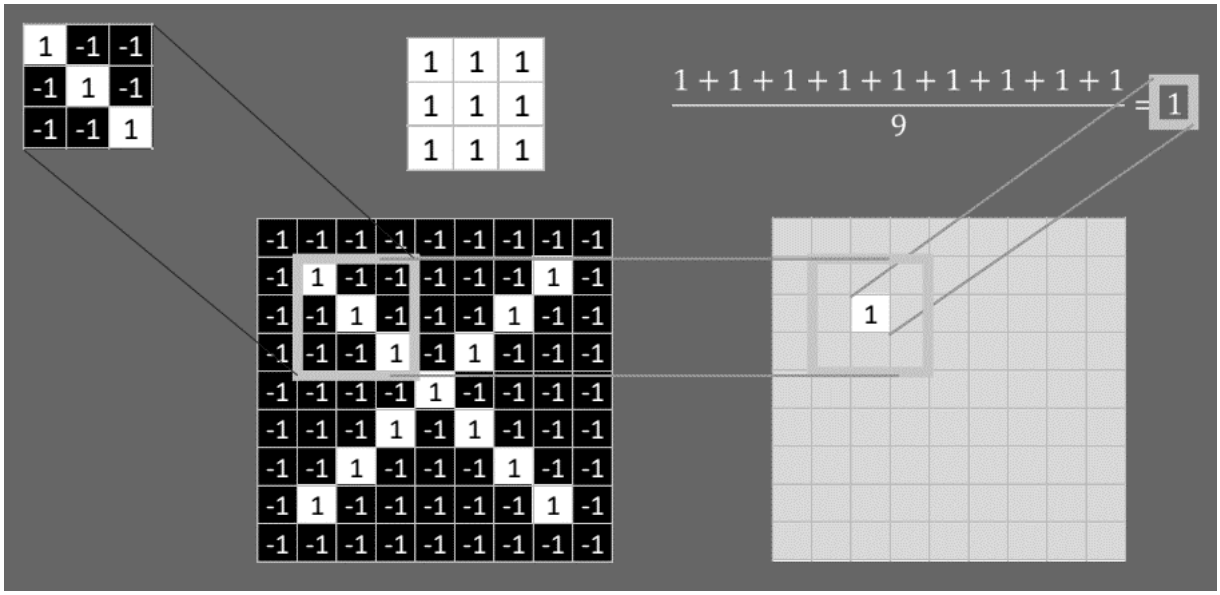
Sendo assim, tal subconjunto é selecionado pelo que em redes convolucionais é chamado de campo receptivo local. Esse campo tem como responsabilidade percorrer toda imagem da esquerda para direita e de cima para baixo, onde a cada deslocamento uma operação de convolução é realizada.

O primeiro passo para realizar a convolução é multiplicar cada pixel do subconjunto pelo pixel correspondente do *kernel*. O resultado dessa multiplicação deverá ser adicionado a um somatório, que por sua vez será dividido pela quantidade de pixels que foram multiplicados.

Assim como expresso na Figura 9, o resultado da convolução é o número 1, indicando que o *kernel* e o subconjunto da imagem analisada apresentam 100% de similaridade.

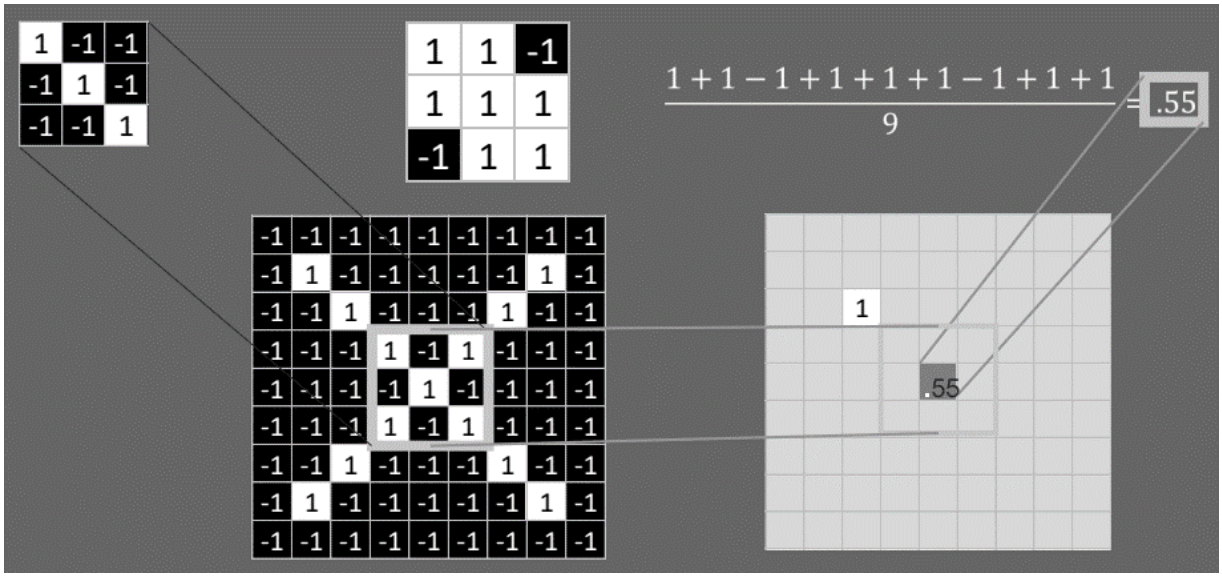
O mesmo procedimento pode ser observado na Figura 10, porém, o subconjunto analisado não possui total semelhança com o *kernel*, logo, o resultado encontrado é igual a 0.55, que representa o quanto a informação expressa pelo *kernel* está presente no subconjunto analisado. De forma visível, é possível inferir que o kernel em questão está buscando elementos na diagonal.

Figura 9 - Resultado da Convolução (Campo Receptivo Local 1)



Fonte: (Soares, 2018)

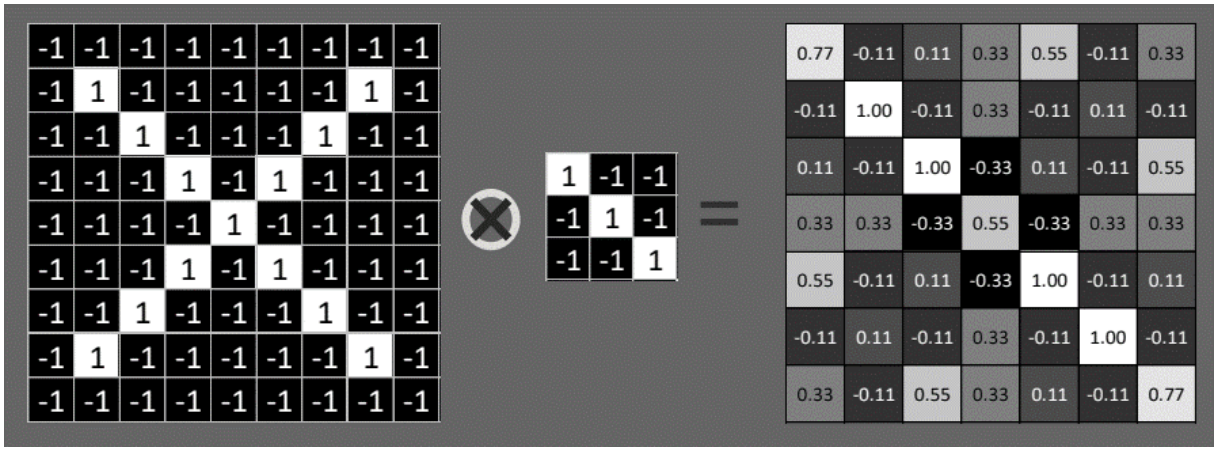
Figura 10 - Resultado da Convolução (Campo Receptivo Local 2)



Fonte: (Soares, 2018)

O resultado da convolução de toda imagem com o *kernel* que busca diagonais é apresentado na Figura 11. A saída desse processo de convolução é chamado de mapa de ativação. Note que as maiores intensidades (resultado 1) foram encontradas na diagonal principal, justamente a mesma informação emitida pelo *kernel*.

Figura 11 - Resultado da Convolução

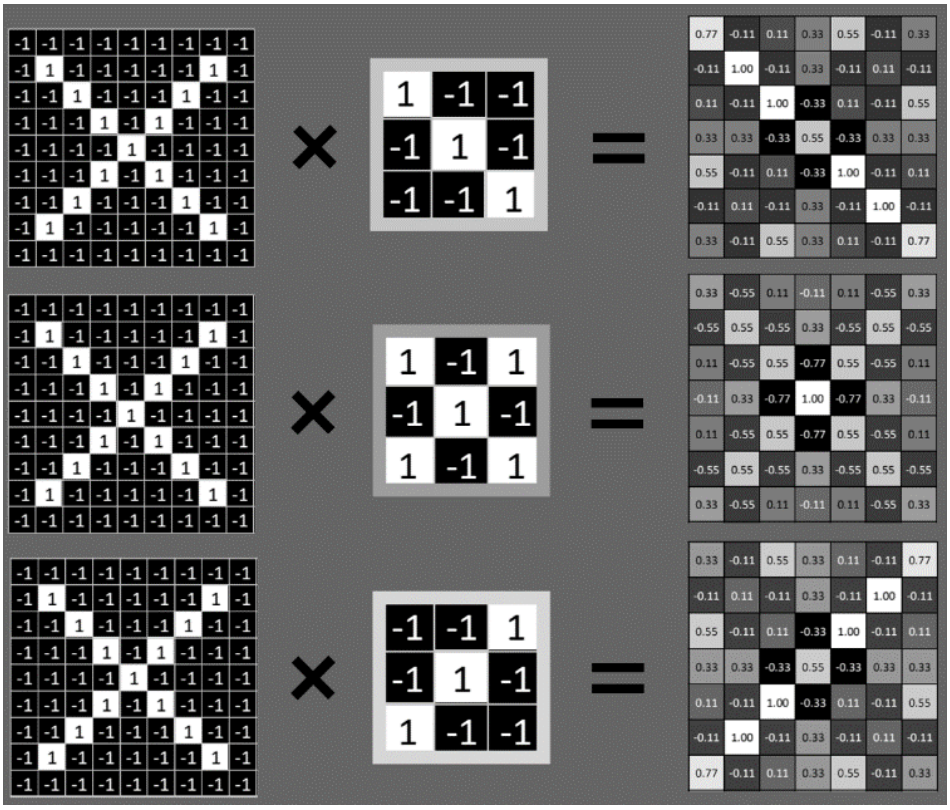


Fonte: (Soares, 2018)

É importante destacar que convoluir a imagem apenas com o *kernel* acima não é capaz de resolver o problema de classificação do objeto de estudo. Sendo assim, no mínimo serão necessários três mapas de ativação para chegar a um resultado satisfatório de classificação.

Observando a figura 7, note que o primeiro mapa é responsável por identificar diagonais principais, o segundo por identificar intersecções entre diagonais e o terceiro por identificar diagonais secundárias. Ao realizar o agrupamento destes três mapas o problema de classificação do sinal de X pode ser solucionado.

Figura 12 - Mapas de Características



Fonte: (Soares, 2018)

2.5 ARQUITETURA DE REDES CONVOLUCIONAIS

Redes Convolucionais recebem uma imagem em sua camada de entrada, ou também conhecida como *input layer*. Computacionalmente falando, imagens são nada mais do que matrizes de dimensões MxN. ConvNets não são capazes de receber matrizes como neurônios de sua camada de entrada. Sendo assim, um procedimento chamado *flatten* (Figura 13) é aplicado sobre a imagem de treino. Nesse contexto, o *flatten* é responsável por realizar o “achatamento” das imagens de entrada da rede, transformando uma matriz em um *array*. Nesse exemplo, o que antes era uma matriz 25x25 acaba-se transformando em um *array* 1x625.

Figura 13 - Flatten 3x3

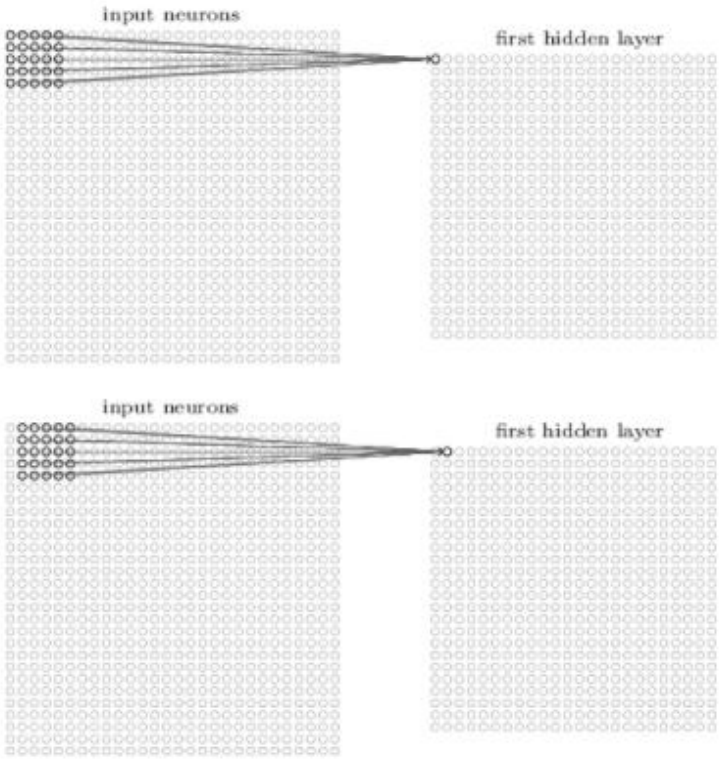
Fonte: Próprio Autor

Em redes neurais densamente conectadas, cada pixel da imagem irá se conectar com todos os neurônios da primeira camada escondida, em seguida, cada neurônio da primeira camada escondida irá se conectar com todos os neurônios da segunda camada escondida, e o processo se repetirá até os neurônios da última camada escondida.

Por sua vez, em Redes Convolucionais, cada neurônio da primeira camada escondida irá se conectar a uma pequena parte dos pixels da camada de entrada. Imagine uma janela 5 por 5, o que totaliza 25 pixels na camada de entrada. A essa janela é dado o nome de campo receptivo local.

Após cada conexão da camada realizar o cálculo de peso e o bias dos seus respectivos neurônios, a janela se desloca para o lado e repete o processo novamente. Esse movimento de deslocamento é chamado de *stride*, podendo ser parametrizado pelo criador da rede. É importante ressaltar que quanto maior o *stride*, maior a velocidade de treinamento do modelo, porém, mais características do objeto-alvo são perdidas na execução do treinamento, assim como ilustrado na Figura 14.

Figura 14 - Stride

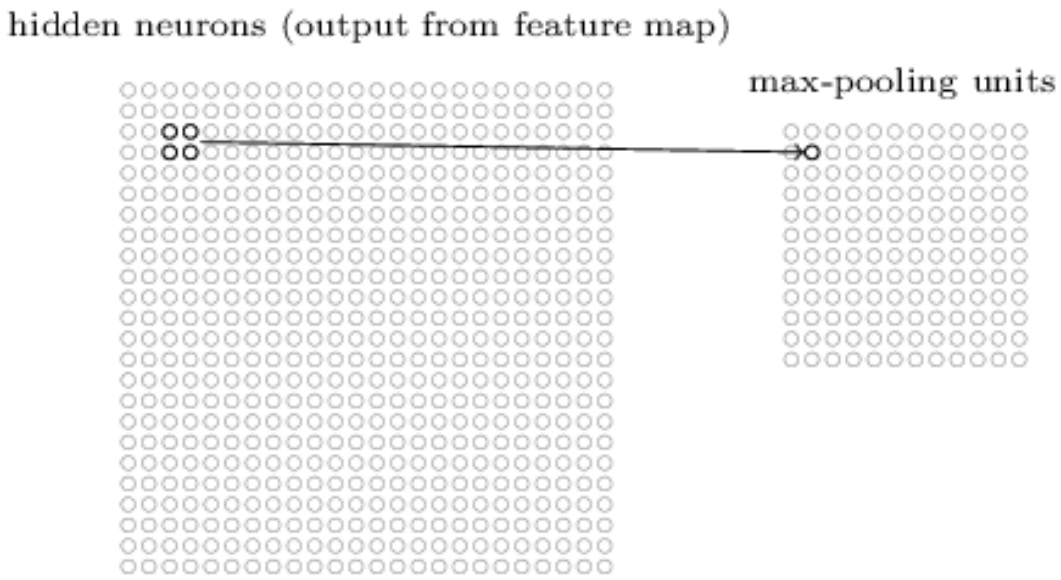


Fonte: (Deep Learning Book)

Ao fim do processo de varredura da imagem é obtido o chamado mapa de características. Cada mapa de característica é responsável por identificar uma informação na imagem, entretanto, tal informação poderá ser identificada independente de ter sido transladada, escalada ou até mesmo rotacionada, como será visto a seguir com a aplicação de um método chamado *Max-Pooling*.

Após realizar a execução da varredura, os mapas de características são submetidos ao procedimento chamado *pooling*. Tal processo tem como objetivo simplificar e condensar as informações obtidas na saída das camadas convolucionais, o que consequentemente reduz drasticamente a quantidade de parâmetros que deverão ser treinados pela rede. No exemplo da Figura 15 é utilizado o procedimento de *Max-Polling*, onde o maior elemento de cada região da janela 2x2 é processado como um neurônio de saída. De forma geral, existem outros dois tipos de *pooling*, sendo eles o *Min-Polling* e *Avarage-Polling*, onde respectivamente visam selecionar o menor elemento da janela e a média dos elementos presentes na janela.

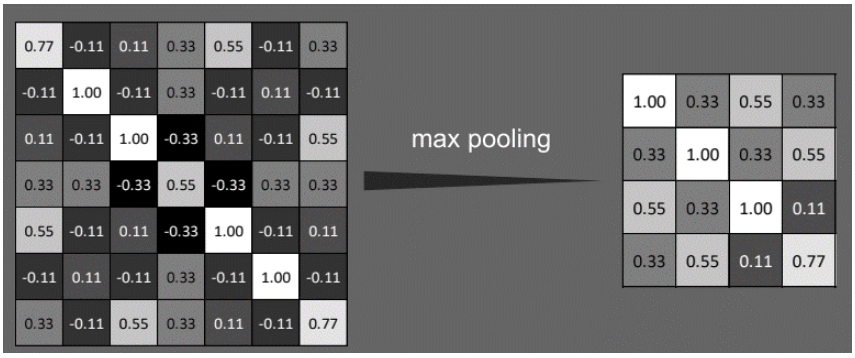
Figura 15 - Camada de Pooling



Fonte: (Deep Learning Book)

Ao observar a saída da camada de *pooling* da Figura 16, percebe-se que mesmo reduzindo o mapa em aproximadamente 3 vezes, as características presentes, no geral, foram preservadas. Essa camada também tem por responsabilidade reduzir a quantidade de ruídos ao redor da área analisada, garantido que apenas os neurônios máximos perpetuem na saída do processo. Sendo assim, independentemente de onde a característica esteja localizada na imagem, ela será identificada, visto que esta estará acentuada no mapa de ativação e principalmente na saída do mencionado processo.

Figura 16 - Max-Pooling



Fonte: (Soares, 2018)

Por fim, todas as saídas da camada de *pooling* serão sujeitas ao mesmo processo de *flatten* mencionado no início do capítulo, porém aqui, este processo visa agrupar as saídas em apenas um vetor, que por sua vez será adicionado como camada de entrada de uma rede densamente conectada, onde finalmente irá realizar o procedimento de classificação.

3. MATERIAIS E MÉTODOS

Este capítulo aborda todos os materiais utilizados e os métodos adotados para o desenvolvimento deste projeto. De forma detalhada, serão especificados os recursos utilizados e os meios para alcançar os objetivos propostos por este trabalho.

3.1 TECNOLOGIAS

Essa seção irá dar uma breve introdução ao leitor sobre as tecnologias que foram utilizadas neste trabalho e serão abordadas posteriormente.

3.1.1 OpenCV

Tendo a Intel Corporation como sua primeira desenvolvedora, *OpenCV* (Open Source Computer Vision) é uma biblioteca de programação, de código aberto, que disponibiliza uma grande quantidade de ferramentas destinadas à visão computacional. Foi desenvolvida com a premissa de fazer da visão computacional uma área acessível para desenvolvedores.

De acordo com Gary Bradski e Adrian Kaehler (2008) no livro *Learning Opencv: Computer Vision with the Opencv Library*, o código fonte da biblioteca é escrito em C e C++, podendo ser executado nos principais sistemas operacionais fornecidos pelo mercado. Também, de acordo com o site oficial da biblioteca, atualmente existem interfaces de comunicação com as linguagens Python, Java e MATLAB.

O *OpenCV* possui módulos de processamento de imagens e vídeo, além de mais de 350 algoritmos de visão computacional como, filtro de imagem, reconhecimento de objetos, faces, entre outros. Sendo assim, esta foi uma ferramenta indispensável para construção do projeto proposto.

3.1.2 Keras

De acordo com o próprio site da ferramenta, Keras é definida como uma API de redes neurais de alto nível, escrita em Python e capaz de rodar sobre TensorFlow, CNTK (*Microsoft Cognitive Toolkit*) ou Theano, sendo esta última uma biblioteca de otimização para manipular expressões matemáticas. Além de modular e extensiva, tem como objetivo principal permitir com que seu usuário desenvolva de forma rápida e amigável.

O Keras possui alguns módulos de fundamental importância para a implementação da solução desse trabalho. Observe a seguir a definição de alguns desses módulos segundo a documentação oficial da biblioteca:

- Models.Sequential: O módulo sequencial é apropriado para uma pilha de camadas onde cada camada tem exatamente um tensor (unidade escalar) de entrada e um tensor de saída;

- Layers.Conv2D: Essa camada cria um *kernel* de convolução que é aplicado (convoluído) à camada de entrada para disponibilizar um tensor de saídas;

- Layers.Activation: Esse módulo é responsável por representar uma determinada função de ativação. Essas funções podem ser utilizadas por meio do módulo layers.activation ou podem ser passadas como argumento de qualquer camada forward;

- Layers.MaxPooling2D: Reduz a resolução da entrada ao longo de suas dimensões (altura e largura), obtendo o valor máximo sobre uma janela de entrada (parametrizável através do pool_size) para cada canal da entrada. A janela é deslocada por strides ao longo de cada dimensão;

- Layers.Flatten: Realiza a conversão de uma matriz em um único array;

- Layers.Dropout: A camada dropout define aleatoriamente as unidades de entrada para 0 com uma frequência de taxa em cada etapa durante o tempo

de treinamento, o que ajuda a evitar o overfitting. As entradas não definidas para 0 são aumentadas em 1/(1 - taxa) de forma que a soma de todas as entradas permaneça inalterada.

- Callback.ModelCheckpoint: O retorno da chamada do método é usado em conjunto com model.fit() para salvar um modelo ou pesos (em um arquivo de pontos de verificação) em algum intervalo, para que o modelo ou pesos possam ser carregados posteriormente para continuar o treinamento do estado salvo.

3.2 DATASETS

3.2.1 MaskedFace-NET

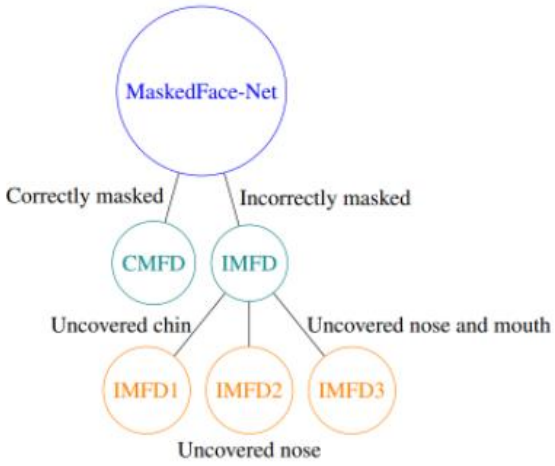
Para realizar o treinamento do modelo, um conjunto de dados de faces mascaradas e não mascaradas foi requisitado, assim possibilitando o aprendizado profundo da rede. Dessa forma, um dos *datasets* que serviu como base para criação do *dataset* definitivo do projeto foi o MaskedFace-NET (MFD). O *dataset* é bastante robusto, e conta com 67.049 imagens de faces corretamente mascaradas, contra 66.734 imagens de faces incorretamente mascaradas, gerando um total de 133.783 imagens e aproximadamente 38 gigabytes de dados para serem processados.

No que diz respeito ao conjunto de imagens com faces incorretamente mascaradas, essas são divididas em três classes, sendo:

- Nariz e boca descobertos (10%);
- Queixo descoberto (10%);
- Nariz descoberto (80%);

Na Figura 17 é possível observar o fluxo em que o *dataset* é dividido. *MaskedFace-Net* pode ser acessado através do seguinte link: https://github.com/cabani/MaskedFace-Net

Figura 17 - Fluxos Dataset(MaskedFace-NET)



Fonte: (Cabani, Hammoudi, Benhabiles, & Melkemi, 2020)

Os desenvolvedores do *MaskedFace-NET* criaram seu próprio *dataset* a partir de um outro chamado Flickr-Faces-HQ (FFHQ). As máscaras são inseridas digitalmente utilizando um classificador em cascata baseados em regiões de interesse. Resumidamente falando, 12 pontos ao redor da área de interesse são pivotados em cada imagem do *dataset*. Em seguida, outros 12 pontos correspondentes que foram pivotados na máscara digital são utilizados como referência para posicionar a máscara sobre a imagem do *dataset*. A Figura 18 ilustra todo o processo detalhado acima.

Figura 18 - Fluxo de Criação do Dataset

Fonte: (Cabani, Hammoudi, Benhabiles, & Melkemi, 2020)

3.2.2 Flickr-Faces-HQ MaskedFace-NET

Flickr-Faces-HQ é um *dataset* de faces humanas com variações de idades, etnias, ângulos e planos de fundo. Todas suas 70.000 imagens possuem uma resolução de 1040 x 1040 pixels.

Além da variação étnica mencionada anteriormente, o *dataset* também possui uma grande variedade de indivíduos portadores de acessórios, variando entre óculos de sol, óculos de grau, chapéu, entre outros. Essa diversificação de informações ajuda a fazer com que o modelo se generalize o máximo possível em suas classificações.

Todas as imagens deste *dataset* foram extraídas de um banco de imagens chamado Flickr, e foram recortadas utilizando uma biblioteca de processamento de imagens chamada dlib. Importante ressaltar que apenas imagens sob licenças permissivas foram coletadas.

Flickr-Faces-HQ pode ser encontrada acessando o seguinte endereço: https://github.com/NVlabs/ffhq-dataset

3.2.3 Data Augmentation

O *MaskedFace-NET* (MFD) é um *dataset* de extrema robustez, porém, implementa a inserção digital de apenas um modelo de máscara, o que pode acabar influenciando consideravelmente a não generalização da classificação realizada pelo software. Pensando nisso, um método conhecido como *data augmentation* foi aplicado nos dados fornecidos pelo *dataset*.

Data Augmentation é o nome de uma técnica utilizada para promover o aumento e a diversificação das informações que serão processadas na rede convolucional. Dessa forma, uma determinada imagem pode ser transformada em outras imagens derivadas. Geralmente, operações de rotação, escalamento e translação já são suficientes para transformar completamente uma imagem na perspectiva de determinadas redes.

Como mencionado no capítulo 2.5, redes convolucionais apresentam uma característica chamada invariância. Logo, conseguem realizar o treinamento e classificação independente das variações mencionadas no parágrafo anterior. Apesar disso, máscaras de proteção podem apresentar características que variam de uma máscara para outra, como cor, formato, material e tamanho, como apresentado na Figura 19.

Figura 19 - Modelos de Máscaras

Fonte: Próprio Autor

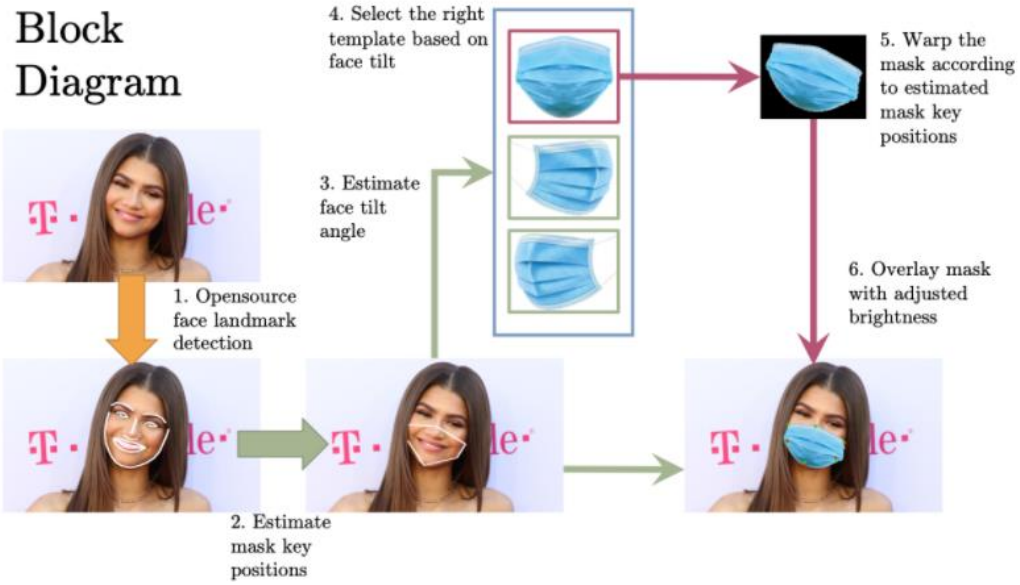
Na seção 3.2.1 é informado que o *dataset* MaskedFace-NET faz uso de um outro *dataset* chamado Flickr-Faces-HQ. Por sua vez, este segundo foi utilizado como base para realizar o aumento de dados. Um procedimento semelhante ao do MFD foi realizado, porém, máscaras de características diversas foram inseridas digitalmente na mesma imagem utilizando o script *MaskTheFace*, abordado na seção 3.2.4. Esse procedimento tem por objetivo fazer com que o software generalize o maior conjunto de máscaras de proteção possível.

3.2.4 MaskTheFace

MaskTheFace é um script baseado em visão computacional para mascarar rostos em imagens. Esse script utiliza um detector de pontos de referência facial baseado na dlib para identificar a inclinação do rosto e seis características principais necessárias para a aplicação da máscara. O fluxo completo do script pode ser visualizado na Figura 20.

Primeiramente é utilizado um identificador de pontos de interesse fornecido pela biblioteca dlib. Em seguida os seis pontos de interesse para o posicionamento da máscara são demarcados. Com base na angulação da face identificada no frame, um determinado template de máscara é selecionado. Por fim, a máscara é ajustada ao rosto e uma correção de brilho é aplicada sobre ela.

Figura 20 - Fluxo MaskTheFace



Fonte: (Aqeel & Raychowdhury, 2020)

A utilização desta biblioteca para a criação de um *dataset* é justificada devido a dificuldade em coletar um conjunto de dados de máscara sob várias condições. A seção a seguir irá apresentar os três *datasets* utilizados no projeto e como estes estão particionados.

3.2.5 Criação dos Datasets

Como mencionado na seção 3.2.3, houve a necessidade de realizar a criação de um *dataset* personalizado para este projeto, visando obter uma generalização maior na classificação do modelo.

Ao todo foram desenvolvidos três tipos de *datasets*, tendo como objetivo observar o comportamento do modelo em ambientes com uma variação relativamente grande de imagens. A tabela 1 apresenta a divisão dos *datasets* em quantidade e tipos de máscaras:

Tabela 1 - Divisão do Dataset

Tipo	Dataset 1	Dataset 2	Dataset 3
KN95	0	1	2
N95	0	1	1
Pano	4	9	20
Cirúrgica	1	2	2

Fonte: Próprio Autor

4. DESENVOLVIMENTO DA SOLUÇÃO

Esse capítulo será responsável por descrever como e quais tecnologias foram utilizadas no desenvolvimento do software de classificação.

4.1 PRÉ-PROCESSAMENTO

Após escolhido o *dataset* sobre o qual a rede convolucional foi treinada e aplicado o aumento de dados, um pré-processamento foi realizado nas imagens fornecidas, visto que a máquina utilizada para treinar o modelo não possui hardware suficiente para trabalhar com uma quantidade excessiva de imagens em uma resolução de 1040 x 1040 pixels.

O pré-processamento aplicado tem como objetivo reduzir a resolução da imagem para 128 x 128 pixels e converter os três canais de cores (RGB) para tons de cinza, assim, diminuindo drasticamente os requisitos computacionais de armazenamento da máquina.

No que diz respeito a resolução de saída, essa não foi escolhida por acaso, visto que 128 pixels em ambos os eixos são suficientes para expressar características fundamentais do objeto-alvo, resistindo também as camadas de *pooling* que simplificaram ainda mais as informações contidas na imagem.

É importante ressaltar que o *dataset* selecionado fornece os dados de forma bruta, logo, elementos dispensáveis para o treinamento do modelo precisaram ser

removidos no pré-processamento. Sendo assim, foi necessária uma forma de extrair apenas a região das imagens que dizem respeito às faces dos indivíduos.

Figura 21 - Dado Bruto x Dado Pré-Processado

Fonte: (Hellsten, Kuosmanen, & Jäni, 2018)

Como mencionado na seção 3.1.1, a biblioteca OpenCV oferece diversas ferramentas quando o assunto é pré-processamento de imagens. Para esse caso foi utilizado o módulo de detecção de faces. O referido módulo fornece para seu usuário um modelo em XML treinado com Haar Cascade, um algoritmo de aprendizado que seleciona características visuais critica para realiza detecção de objetos, onde caso aplicado a uma imagem, retorna as coordenadas que possuem uma face presente. Importante mencionar que o modelo fornecido consegue retornar as coordenadas inclusive de faces mascaradas.

4.2 TREINAMENTO E PARAMETRIZAÇÃO

Após realizar os procedimentos de aumento de dados e pré-processamento, um algoritmo foi desenvolvido e implementado para realizar o treinamento do modelo de classificação.

Primeiramente, todas as bibliotecas e módulos mencionados no capítulo 3.1 foram importados de forma global na aplicação. Em seguida, as variáveis data e target são instanciadas na memória. Essas variáveis buscam respectivamente carregar os dados de treinamento (imagens) e definir quais são os alvos que cada imagem deseja atingir (com máscara ou sem máscara).

O segundo passo foi definir as configurações de parâmetros do modelo. Nessa etapa foram criados três perfis de parametrização com o objetivo de gerar

dados comparativos. Na tabela 2 são apresentados os valores utilizados na parametrização de cada um dos perfis.

No modelo em questão foram implementadas duas camadas convolucionais, onde cada uma é seguida de uma função de ativação e uma camada de *pooling*. A escolha de duas camadas convolucionais foi realizada devido a resolução da imagem e o tamanho do objeto de estudo presente nela. A mesma justificativa é utilizada na escolha do tamanho do campo receptivo local.

Após parametrizadas as duas camadas convolucionais, o resultado obtido na saída da segunda camada de *polling* é adicionado como entrada de um método responsável por realizar o procedimento de *flatten* dos dados. A partir do momento que os dados estão sendo representados em uma estrutura de dados unidimensional, cada posição dessa estrutura pode simbolizar um neurônio.

Antes de processar os novos neurônios em uma rede densamente conectada, uma camada de *dropout* é adicionada como intermediária. A camada de *dropout* é utilizada como uma forma de evitar o *overfitting* do modelo treinado, ou seja, um modelo especialista em classificar apenas os dados de treinamento. Uma exclusão temporária e aleatória de alguns neurônios é realizada, fazendo assim com que o modelo aprenda a realizar classificações mesmo que algumas características fundamentais estejam faltando no treinamento.

Por fim, uma rede densa com X neurônios é configurada, fazendo com que no final haja apenas dois neurônios que representarão a camada de saída.

Tabela 2 - Perfis de Parametrização

Camadas		**Perfis**		
		1	**2**	**3**
1ª Camada Convolucional	Mapas de Características	4	8	16
	Campo Receptivo Local	6x6	8x8	10x10
	Stride	1x1	1x1	1x1
	Pooling	2x2	3x3	2x2

Continua

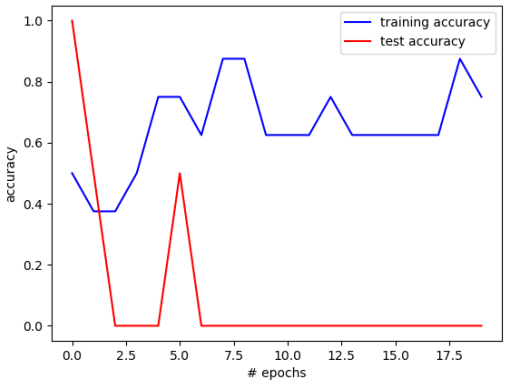
Cont. Tabela 2

2ª Camada Convolucional	Mapas de Características	2	4	8
	Campo Receptivo Local	3x3	4x4	5x5
	Stride	1x1	1x1	1x1
	Pooling	2x2	2x2	2x2
Dropout		0.5	0.5	0.5
Neurônios Densamente Conectados		50	50	50
Neurônios Densamente Conectados		2	2	2

Fonte: Próprio Autor

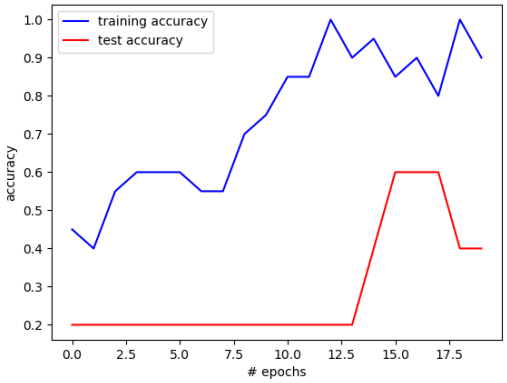
Ao final do treinamento de cada um dos modelos, um arquivo de registro e um gráfico de acurácia foram gerados. Os gráficos mencionados podem ser observados nas figuras ilustradas abaixo.

Figura 22 - Gráfico de Acurácia (Perfil 1 x Dataset 1)



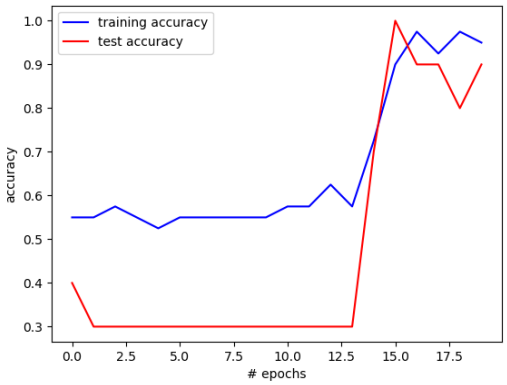
Fonte: Próprio Autor

Figura 23 - Gráfico de Acurácia (Perfil 1 x Dataset 2)



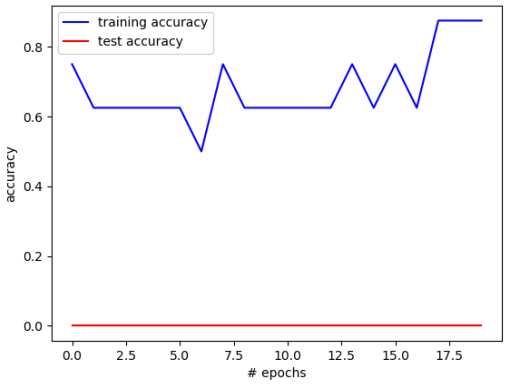
Fonte: Próprio Autor

Figura 24 - Gráfico de Acurácia (Perfil 1 x Dataset 3)



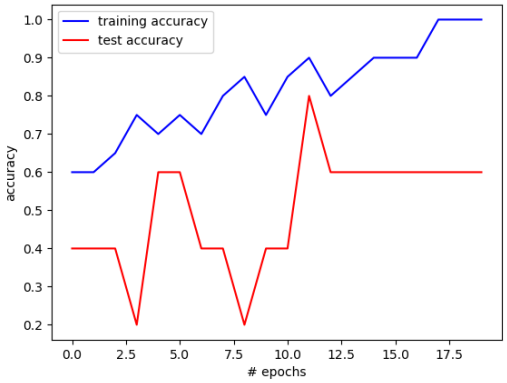
Fonte: Próprio Autor

Figura 25 - Gráfico de Acurácia (Perfil 2 x Dataset 1)



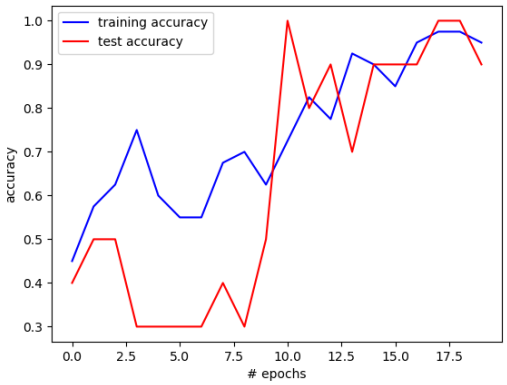
Fonte: Próprio Autor

Figura 26 - Gráfico de Acurácia (Perfil 2 x Dataset 2)



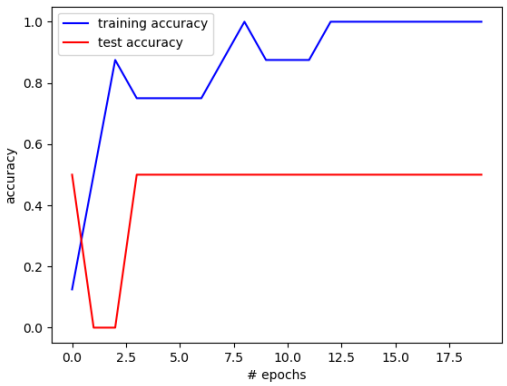
Fonte: Próprio Autor

Figura 27 - Gráfico de Acurácia (Perfil 2 x Dataset 3)



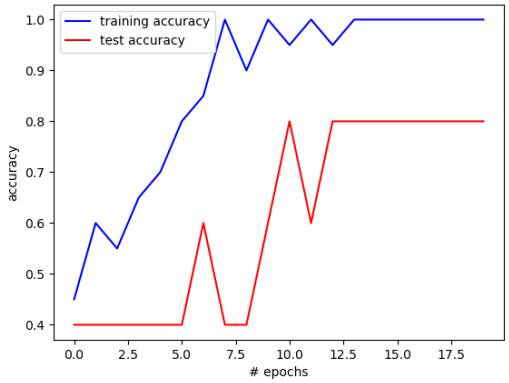
Fonte: Próprio Autor

Figura 28 - Gráfico de Acurácia (Perfil 3 x Dataset 1)



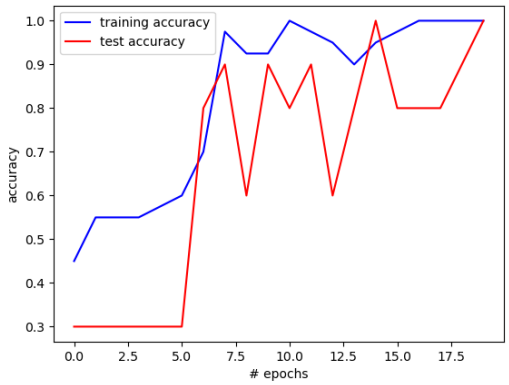
Fonte: Próprio Autor

Figura 29 - Gráfico de Acurácia (Perfil 3 x Dataset 2)



Fonte: Próprio Autor

Figura 30 - Gráfico de Acurácia (Perfil 3 x Dataset 3)



Fonte: Próprio Autor

Como apresentado nos gráficos, o modelo é treinado durante 20 épocas. Um recurso disponibilizado pela biblioteca de treinamento permite que ao fim de cada época um modelo seja disponibilizado. Por sua vez, esse modelo é responsável por processar as imagens de teste e disponibilizar sua acurácia.

Muitas vezes, nas primeiras épocas a acurácia de teste acaba sobrepondo a acurácia de treino. Isso acontece pois os primeiros modelos ainda estão se adaptando aos dados de treino que foram submetidos. A acurácia de treino leva em consideração todas as iterações e reajustes do modelo. Por sua vez, a acurácia de teste leva em consideração apenas o resultado do modelo já treinado. Um outro fator que contribui para esse fenômeno é que as imagens de teste correspondem a apenas 20% do *dataset* utilizado, logo, para o *dataset* 1 isso representa apenas 2 imagens. Essa discrepância é facilmente observada nos modelos que foram

treinados utilizando o *dataset* 1, provido apenas de 10 imagens. Apesar disso, conforme o volume do *dataset* aumenta, a discrepância apresentada acaba sendo suavizada.

Ao final do processo de treinamento dos modelos, apenas aqueles que apresentaram melhores resultados ao longo das épocas é selecionado. Todos os modelos selecionados foram submetidos a um teste com imagens exclusivas. Apesar dos modelos treinados com o *dataset* 1 possuírem um volume de dados reduzido, vale destacar que alguns desses modelos ainda assim acabaram por apresentar resultados satisfatórios.

O modelo treinado utilizando as parametrizações do perfil3 e os dados fornecidos pelo *dataset* 1, acabou sobressaindo em relação aos demais modelos que utilizaram o mesmo *dataset*, porém, perfis diferentes. De qualquer forma, em alguns momentos a classificação acaba apresentando estados intermitentes, principalmente nos momentos em que o rosto analisado muda sua angulação.

Uma relação de assertividade e quantidade de imagens de treinamento pôde ser observada durante a realização das classificações. Os modelos treinados com os *datasets* 2 e 3 acabara minimizando as classificações equivocadas que foram presentes nos modelos treinados com o *dataset 1*.

Com base no que foi relatado anteriormente, fica explícito que a quantidade de imagens disponibilizadas para o treinamento do modelo é um fator chave nas classificações de sucesso. Também deve ser levado em consideração que *datasets* maiores são carregados com mais informações de angulação facial e características étnicas, pontos que são grandes candidatos a serem a causa raiz das classificações erradas providas pelos modelos treinados com o *dataset 1*.

Para que a performance dos modelos fosse testada em um ambiente totalmente desconhecido, um vídeo foi gravado, onde nele é realizada a captura de vários ângulos de uma face humana. Levando em consideração que um vídeo é nada mais do que um conjunto de imagens que se sucedem, essa filmagem produziu um total de 2736 imagens para teste. Do total de imagens produzidas pela filmagem, apenas 2170 foram passivas de atuação do algoritmo de detecção facial. Isso se deve a fatores como iluminação, perca do objeto-alvo (face) no frame, desfoque da imagem e até mesmo a instabilidade da câmera no momento de realizar a captura, assim, gerado o que é chamado de borrão.

Figura 31 - Detecção - Modelo Dataset1/Perfil3

Fonte: Próprio Autor

Figura 32 - Detecção - Modelo Dataset2/Perfil3

Fonte: Próprio Autor

Figura 33 - Detecção - Modelo Dataset3/Perfil3

Fonte: Próprio Autor

Ao submeter os 9 modelos produzidos nesse novo *dataset*, foi compreensível que os modelos treinados com o *dataset 1* acabaram por apresentar um número exorbitante de falsos negativos (indicando com máscara, quando na verdade a face encontra-se sem). Em relação aos modelos que melhor performaram, é possível observar na tabela 3 que esses foram: *dataset* 3/perfil2 e *dataset* 3/perfil 3.

Tabela 3 - Resultados dos Modelos

	Dataset 1/Perfil 1	**Dataset 1/Perfil 2**	**Dataset 1/Perfil 3**	**Dataset 2/Perfil 1**	**Dataset 2/Perfil 2**	**Dataset 2/Perfil 3**	**Dataset 3/Perfil 1**	**Dataset 3/Perfil 2**	**Dataset 3/Perfil3**
Acerto	19,93%	25,68%	37,98%	47,87%	96,11%	99,87%	87,69%	99,26%	100%

Fonte: Próprio Autor

Ao fim, é possível concluir que a parametrização do perfil 3 acabou se sobressaindo levemente em relação a parametrização do perfil 2, visto que ao utilizar o *dataset* 3 o modelo treinando com os parâmetros do perfil 3 conseguiu detectar 100% das imagens disponibilizadas pela captura de vídeo.

5. CONSIDERAÇÕES FINAIS

No início do trabalho, um dos grandes objetivos era realizar um estudo sobre redes neurais convolucionais e desenvolver um modelo que classificaria se determinada face está ou não mascarada. Após a realização dos estudos, constatado que o uso dessa tecnologia tem proporcionado grandes avanços na área de classificação de imagens.

Estabelecido que no estudo e desenvolvimento da aplicação seriam utilizadas as bibliotecas *OpenCV* e *Keras*, onde ambas apresentaram uma grande valia no que diz respeito a detecção facial e treinamento de modelos de *machine learning*, respectivamente. Essas bibliotecas, associadas a linguagem Python, possibilitaram que o objetivo do trabalho fosse concluído com êxito.

A fim de enriquecer o trabalho e realizar a criação de um material de treinamento autoral, vários *datasets* foram explorados e tiveram serventia como inspiração para o desenvolvimento do *dataset* definitivo. *MaskedFace-NET* e *Flickr-Faces-HQ* foram os escolhidos para serem abordados no capítulo 3.

Um fator importante no desenvolvimento desse trabalho foi a utilização do script *MaskTheFace*, responsável por realizar as detecções de áreas de interesse em uma face humana, e em seguida executar o posicionamento dos modelos de máscaras na imagem alvo. A não utilização desse script faria com que o processo de criação do *dataset* se tornasse lento e desgastante.

No treinamento utilizando as redes convolucionais, os resultados obtidos nos modelos foram capazes de classificar a grande maioria das imagens disponibilizadas para teste. Entretanto, o modelo treinado utilizando o *dataset* 3 e as parametrizações do perfil 3 acabou se sobressaindo em relação aos outros, assim, tornando-se o modelo definitivo desse trabalho.

Apesar do modelo definitivo conseguir bons resultados em acurácia de treino, acurácia de teste e validação com o processamento de imagens externas, seria interessante que em trabalhos futuros fosse realizada a aplicação de um pré-processamento mais intenso nas imagens avaliadas, visto que o modelo poderá estar atuando em diferentes tipos de ambientes, com diferentes pontos de captura das imagens e iluminação com variação constante.

REFERÊNCIAS

AQEEL, Anwar; RAYCHOWDHURY, Arijit. (2020). **Masked Face Recognition for Secure Authentication**. Fonte: Github: https://github.com/aqeelanwar/MaskTheFace

CABANI, Adnane; HAMMOUDI, Karim; BENHABILES, Halim. (3 de Novembro de 2020). **MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19**. Smart Health, 2-6.

Deep Learning Book. (s.d.). **Campos Receptivos Locais em Redes Neurais Convolucionais**. Fonte: Deep Learning Book: https://www.deeplearningbook.com.br/campos-receptivos-locais-em-redes-neurais-convolucionais

G1 RS. (5 de Março de 2021). **Uso de máscara reduz chance de infecção por coronavírus em 87%, diz estudo de universidades do RS**. Fonte: G1: https://g1.globo.com/rs/rio-grande-do-sul/noticia/2021/03/05/uso-de-mascara-reduz-chance-de-infeccao-por-coronavirus-em-87percent-diz-estudo-de-universidades-do-rs.ghtml

GOV.BR. (3 de Julho de 2020). **Lei que torna obrigatório o uso de máscara é sancionada**. Fonte: gov.br: https://www.gov.br/planalto/pt-br/acompanhe-o-planalto/noticias/2020/07/lei-que-torna-obrigatorio-o-uso-de-mascara-e-sancionada

GOV.BR. (08 de Abril de 2021). **O que é a Covid-19?** Fonte: gov.br: https://www.gov.br/saude/pt-br/coronavirus/o-que-e-o-coronavirus

HELLSTEN, Janne., KUOSMANEN, Tero; JANI, Pekka. (2018). **Flickr-Faces-HQ Dataset (FFHQ)**. Fonte: Github: https://github.com/NVlabs/ffhq-dataset

LAMA, Rafael. (2016). **Neurônio artificial**. Acesso em 21 de Novembro de 2021, disponível em Wikipédia: https://pt.wikipedia.org/wiki/Perceptron_multicamadas#/media/Ficheiro:Sadssa.png

MATHUR, Pankaj. (1 de Maio de 2016). **A Simple Multilayer Perceptron with TensorFlow**. Fonte: Medium: https://medium.com/pankajmathur/a-simple-multilayer-perceptron-with-tensorflow-3effe7bf3466

ROSEBROCK, Adrian. (4 de Abril de 2020). **COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning**. Fonte: Pyimagesearch: https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

Secretaria Especial de Comunicação. (29 de Junho de 2020). **Decreto permite multa a pessoas ou estabelecimentos que desrespeitarem o uso de máscaras**. Fonte: Cidade de São Paulo: http://www.capital.sp.gov.br/noticia/vigilancia-sanitaria-vai-multar-pessoas-ou-estabelecimentos-que-desrespeitarem-o-uso-de-mascaras

SOARES, Anderson. (2018). **Redes Neurais Convolucionais**. Fonte: Deep Learning Brasil: https://ww2.inf.ufg.br/~anderson/deeplearning/20181/Aula%20-%20Redes%20Neurais%20Convolucionais%20Parte%20I.pdf

STAITE, Tom. (16 de Agosto de 2020). **Face Mask Detection algorithm using Convolutional Neural Network — AI — Computer Vision**. Fonte: Tom Staite: https://medium.com/@tomstaite1/face-mask-detection-algorithm-using-convolutional-neural-network-ai-computer-vision-15f08988533

UNZUETA, Diego. (13 de Novembro de 2020). **Convolutional Layers vs Fully Connected Layers**. Fonte: Towards data science: https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante André Luiz Astrol de Oliveira do Curso de Ciência da Computação, matrícula 201710028024 9-4 telefone: 62 981670584 e-mail alaodev@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Redes Neurais Convolucionais: Análise de Imagens de Vídeo em Tempo Real p/ Ident. de Face e Máscaras, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 15 de Dezembro de 2021.

Assinatura do(s) autor(es): _____

Nome completo do autor: André Luiz Astrol de Oliveira

Assinatura do professor-orientador: _____

Nome completo do professor-orientador: Max Gontijo de Oliveira