UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS DE DOIS VIZINHOS
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

JOSÉ LUIZ NEVES VOLTAN

O EMPREGO DE REDES NEURAIS CONVOLUCIONAIS NA IDENTIFICAÇÃO DE MÁSCARAS DE PROTEÇÃO NO CONTEXTO DA COVID-19

TRABALHO DE CONCLUSÃO DE CURSO DE ESPECIALIZAÇÃO

DOIS VIZINHOS
2022

JOSÉ LUIZ NEVES VOLTAN

O EMPREGO DE REDES NEURAIS CONVOLUCIONAIS NA IDENTIFICAÇÃO DE MÁSCARAS DE PROTEÇÃO NO CONTEXTO DA COVID-19

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Orientador: Prof. Dr. Jefferson Tales Oliva

DOIS VIZINHOS
2022

JOSÉ LUIZ NEVES VOLTAN

O EMPREGO DE REDES NEURAIS CONVOLUCIONAIS NA IDENTIFICAÇÃO DE MÁSCARAS DE PROTEÇÃO NO CONTEXTO DA COVID-19

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Data de aprovação: 08/junho/2022

Jefferson Tales Oliva
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

Dalcimar Casanova
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

Marcelo Teixeira
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

DOIS VIZINHOS
2022

AGRADECIMENTOS

Agradeço à minha esposa, pela paciência e por me apoiar em meus sonhos, sendo meu porto seguro sempre presente.

Aos meus pais, minha fonte de inspiração e altruísmo.

Aos professores do Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, que conduziram o curso de forma primorosa, sempre preocupados em transmitir o conhecimento da melhor maneira.

Em especial, meus agradecimentos ao Prof. Dr. Jefferson Tales Oliva pela gentileza e paciência em aceitar ser orientador deste trabalho e a forma como me orientou.

RESUMO

Durante a pandemia do Coronavírus (COVID-19), o uso de máscaras de proteção se mostrou um importante recurso para frear a contaminação entre pessoas. A máscara é uma recurso não-farmacológico, de baixo custo, mas com grande impacto. Mesmo assim, algumas pessoas não a utilização, ou o fazem de forma incorreta, sendo assim importante a identificação e orientação nesses casos. O trabalho apresenta uma proposta de utilização de Redes Neurais Convolucionais (CNN) como forma de identificar pessoas que estão utilizando máscara de proteção corretamente, incorretamente ou estão sem a máscara (3 classes). Para treinamento e teste, utilizou-se um *dataset* com imagens geradas de forma artificial. Duas arquiteturas de CNN foram propostas e comparadas, gerando 5 modelos cada uma. Apesar dos modelos terem arquiteturas simples, obtiveram uma acurácia superior a 99% no conjunto de teste. As arquiteturas apresentadas pode ser utilizadas para o controle de acesso em ambientes fechados, como forma de se garantir o correto uso da máscara de proteção.

Palavras-chave: CNN. Redes Neurais Convolucionais. COVID-19. máscara.

ABSTRACT

During the Coronavirus (COVID-19) pandemic, the use of protective masks proved to be an important resource to reduce contamination between people. The mask is a non-pharmacological, low-cost resource, but with great impact. Even so, some people do not use it, or do it incorrectly, so identification and guidance in these cases is important. This work presents a proposal for the use of Convolutional Neural Networks (CNN) as a way to identify people who are using a protective mask correctly, incorrectly or are without a mask (3 labels). For training and testing, a *dataset* with artificially generated images was used. Two CNN architectures were proposed, generating 5 models each. These models, despite having a simple architecture, achieved an accuracy greater than 99% in the test set. The architectures presented can be used for access control in indoor environments, as a way of guaranteeing the correct use of the protection mask.

Keywords: CNN; Convolutional Neural Network; COVID-19; mask.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

No final do ano 2019, um novo vírus (SARS-CoV-2) começou a assolar o mundo, desencadeando uma pandemia. O vírus SARS-CoV-2 pertence a família de vírus conhecida como Coronavírus. Ao infectar humanos, o vírus causa a doença denominada COVID-19 (FREITAS et al., 2021). Em fevereiro de 2020, o Ministério da Saúde brasileiro declarou emergência em saúde pública de importância nacional. Já em março de 2020, foi declarado o estado de transmissão comunitária do coronavírus (COVID-19) em todo território nacional (BRASIL. MINISTÉRIO DA SAÚDE, 2020). Esses eventos demonstraram a rápida evolução da pandemia que traria diversos impactos para a sociedade, tais como, fechamento de lojas e escolas, quebra da cadeia produtiva, ocasionada pelo fechamento temporário de fábricas, e consequente aumento de preços.

Diante desse cenário e o aprendizado gradual sobre o vírus SARS-CoV-2 e sobre a doença COVID-19, a Organização Mundial da Saúde (OMS) e a Organização Pan-Americana da Saúde (OPAS) recomendaram o uso de máscaras de proteção como forma de diminuir a transmissão do vírus SARS-CoV-2. Na orientação provisória, de 06 de abril de 2020, foi destacado que o uso de máscaras cirúrgicas pode evitar a disseminação de gotículas infecciosas entre pessoas e o ambiente (OPAS, 2020).

Com a evolução da doença e aumento do número de óbitos, muitos estados e municípios decidiram por tornar obrigatório o uso de máscaras em logradouros públicos, estabelecimentos comerciais, dentre outros locais.

Posteriormente, o Governo Federal, através da Lei nº 14.019, de 2 de julho de 2020 (BRASIL, 2020), tornou obrigatório o uso de máscaras de proteção individual. Da lei, destaca-se que o uso da máscara é obrigatório de forma que a boca e o nariz estejam cobertos. Essa obrigação existe tanto em espaços públicos como privados tais como, vias públicas, transportes públicos coletivos, veículos de transporte remunerado privado individual de passageiros por aplicativo ou por meio de táxis, e ainda, em ônibus, aeronaves ou embarcações de uso coletivo fretados (BRASIL, 2020).

O uso de máscara é considerado uma medida não-farmacológica com capacidade de diminuir a chance de contágio entre as pessoas. Para tanto, ela deve ser confeccionada com material que tenha a capacidade de filtrar partícula do vírus e posicionada de forma adequada no rosto. Desse modo, OPAS (2020, p. 4) recomenda a colocação cuidadosa, de forma que a máscara cubra a boca e o nariz do indivíduo, também é recomendada que a máscara esteja amarrada de forma a minimizar a folga entre ela e o rosto.

A Figura 1 ilustra exemplos de tipos de máscara de proteção. A máscara (a) é do modelo N-95, a (b) é de modelo esportivo, a (c) do tipo cirúrgica descartável, a (d) e a (e) são modelos comuns. Nas imagens, é possível verificar o correto posicionamento das máscaras, cobrindo a região de nariz, boca e queixo, estando bem fixadas no rosto.

Figura 1 – Diversidade de modelos de máscaras

Fonte: Autoria própria (2022)

O uso da máscara é uma medida simples com impacto positivo. Mesmo assim, algumas pessoas não a utilizam ou o fazem de forma incorreta, sem que a posicionem adequadamente no rosto. Assim, é importante reconhecer se uma pessoa está ou não usando corretamente. Uma alternativa para fazer isso de forma automatizada é através de métodos de aprendizado de máquina que realização a tarefa de classificação em imagens. Dessa forma, ao se realizar a detecção de que a pessoa não está com a máscara corretamento no rosto, ela seria orientada para vestir a máscara ou corrigir o posicionamento.

1.1 Problema de Pesquisa

O uso correto da máscara de proteção pode ajudar a diminuir a transmissão do vírus SARS-CoV-2. Diante desse cenário, surge o questionamento de como seria possível detectar de forma automatizada através de algoritmos de visão computacional e métodos de aprendizado de máquina, se uma pessoa está utilizando a máscara e se ela está corretamente posicionada. Essa informação pode ser utilizada, por exemplo, para somente permitir o acesso de pessoas que estejam utilizando corretamente a máscara ou mesmo como orientação para que a pessoa reposicione-a de forma correta.

1.2 Objetivos

1.2.1 Objetivo Geral

Implementar um modelo de classificação, utilizando métodos de aprendizado de máquina, para identificar pessoas sem máscara, pessoas com máscara incorreta e pessoas com máscara correta (três classes).

1.2.2 Objetivos Específicos

- Identificar ou construir um banco de imagens (*dataset*) que possa ser utilizado para treinamento e teste para a tarefa de classificação quanto ao uso da máscara nas classes pessoas sem máscara, com máscara incorreta ou com máscara correta;
- Realizar uma fundamentação teórica sobre inteligência artificial, visão computacional, e as redes neurais convolucionais;
- Comparar o desempenho de métodos de aprendizagem de máquina para essa tarefa;
- Identificar e avaliar formas de automatizar a identificação de pessoas com máscaras.

1.3 Justificativa

O trabalho é relevante por estar relacionado a um problema de saúde global, a pandemia causada pelo vírus SARS-CoV-2. Nesse contexto, o uso de máscara tem sido uma recomendação, por diferentes entidades ligadas à saúde, como forma de diminuir a propagação do vírus. Assim, o trabalho auxilia a propor uma forma eficiente de fiscalização e controle sobre o uso da máscara de proteção.

A hipótese é de que seria possível realizar o controle do uso da máscara de forma automatizada por meio de algoritmos de aprendizado de máquina.

No cotidiano, as pessoas passam por diversos equipamentos que possuem câmeras, ou que poderiam receber uma e assim orientá-las quanto ao uso da máscara. Por exemplo, uma porta automática de shopping, uma *self* antes de iniciar uma viagem em um aplicativo ou mesmo uma catraca. A técnica ora proposta combinada com esses tipos de equipamentos poderia incentivar e corrigir o uso da máscara. O uso das três classes propostas 'sem máscara', 'com máscara incorreta' e 'com máscara correta' permitiria uma orientação mais assertiva. Dessa forma, o estudo proposto também guarda aplicação prática.

1.4 Organização do Trabalho

O trabalho apresenta ainda os seguintes capítulos: o Capítulo 2 aborda conceitos e fundamentos importantes e visa contextualizar o leitor na área da visão computacional e da inteligência artificial; o Capítulo 3 apresenta os principais trabalhos relacionados com a pesquisa, envolvendo a detecção de faces e a classificação quanto ao uso da máscara. O Capítulo 4, explica sobre os materiais utilizados e a metodologia adotada. Em seguida, o Capítulo 5 apresenta um debate sobre os resultados encontrados e compara os modelos propostos. Por fim, o Capítulo 6 discute as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, é feita uma revisão de conceitos importantes para o trabalho. São abordados conceitos referentes à Inteligência Artificial (IA), *Deep Learning*, visão computacional, imagem e Redes Neurais Convolucionais. As métricas de desempenho também são apresentadas por serem um importante tópico que permite comparar os resultados obtidos.

2.1 A Inteligência Artificial

Muitas são as definições para Inteligência Artificial (IA), uma delas seria dizer que a IA é a área que estuda como fazer com que computadores trabalhem, isto é, realizem tarefas de forma similar aos humanos.

Lenz et al. (2020) destacan a diferença entre humanos e máquinas. Enquanto os homens tem uma forma de agir e pensar complexa, nem sempre racional, as máquinas necessitam receber instruções claras do que e como fazer.

Em sua obra, Luger, Vieira e Tavares (2013) utilizam a definição da IA como sendo uma área da ciência da computação focada no estudo da automação do comportamento inteligente. Eles pontuam ainda a dificuldade em se definir uma área tão nova e ao mesmo tempo ampla.

Ao longo dos anos, a IA evoluiu, desenvolvendo diversos conceitos e ganhou maturidade. De acordo com Russell, Norvig e Macedo (2013), o primeiro período da IA foi de gestação e ocorreu entre os anos de 1943 a 1956. Nele, Russell, Norvig e Macedo (2013) destacam o primeiro trabalho que foi considerado como sendo de IA, escrito por Warren McCulloch e Walter Pitts (1943) e que tratou sobre um modelo de neurônios artificiais, os quais poderiam estar ligados ou desligados. Nesse período, também se destacou o trabalho de Alan Turing, e seu artigo intitulado "*Computing Machinery and Intelligence*".

Com o passar dos anos, muitos paradigmas de IA, e mais especificamente de aprendizagem de máquina, surgiram, cresceram e depois foram esquecidos. A computação também evoluiu aumentando sua capacidade de armazenamento e processamento, esses dois fatores fizeram com que tecnologias de IA anteriormente deixadas de lado fossem novamente utilizadas, por exemplo as redes neurais artificiais.

2.1.1 Aprendizado de máquina

Segundo Lenz et al. (2020), a área de aprendizado de máquina (em inglês, *Machine Learning* - ML) vem crescendo de importância devido ao aumento do volume de dados e a dificuldade em se realizar uma análise em um curto intervalo de tempo de forma que a informação extraída ainda seja válida (validade da informação). Métodos de aprendizado de

máquina são utilizados em diversas aplicações, como diagnóstico médico, veículos autônomos, entre outras (LENZ et al., 2020).

Dentre as inúmeras capacidades humanas desejáveis que as máquinas tenham, está o aprendizado. Nesse contexto, surgiu uma sub-área dentro da IA denominada aprendizagem de máquinas (AM). Nessa área, um algoritmo é treinado para identificar características (padrões) em um conjunto de dados e posteriormente, com novas entradas conseguir realizar tarefas de classificação, previsão ou agrupamento de dados (FACELI et al., 2021).

O presente trabalho aborda o problema da classificação, onde as imagens foram classificadas entre 3 possíveis classes (com máscara correta, com máscara incorreta ou sem máscara). Em problemas de classificação, busca-se, por meio de um classificador ($\hat{f}$), prever o atributo alvo (classe) de uma instância. As possíveis classes formam um conjunto finito de hipóteses ($C_1, C_2, ..., C_n$). Caso no problema sejam abordadas apenas duas classes, fala-se em classificação binária, por exemplo classificar se um e-mail é ou não spam. Quando se tem mais do que duas classes, fala-se em multiclasse. Em um tipo especial de classificação, as instâncias podem assumir mais do que uma classe, esse tipo é denominado multirrótulo.

FACELI et al. (2021) destacam o problema da classe majoritária, em que a amostragem desproporcional (desbalanceada) de uma classe pode induzir o classificador a privilegiar a classificação nessa classe. É possível dizer que o classificador tende a atribuir a classe majoritária ao classificar um novo item. Existem diversas abordagens que tentam solucionar esse problema, como a amostragem com mesmo número de exemplares para cada uma das classes, de forma a não privilegiar nenhuma, tornando o conjunto de exemplos balanceados por classe. Outra opção é realizar a amostragem estratificada em relação ao conjunto original, dessa forma, a amostragem respeita a proporção das classes existentes no conjunto original (FACELI et al., 2021). Para evitar o citado problema, este trabalho adotou a distribuição proporcional das imagens entre as três classes, conforme é abordado na Subseção 4.1.1.

Lenz et al. (2020) destacam que o processo geral de aprendizado de máquina pode ser dividido em sete etapas, são elas:

- coleta (seleção) de dados: consiste em coletar os dados atentando-se para sua qualidade e quantidade. Um jargão da área diz que "*garbage in, garbage out*", ou seja, um conjunto de dados de baixa qualidade produz um modelo também de baixa qualidade. Além disso, um conjunto de dados pequeno pode não possibilitar a identificação de padrões relevantes. Essas duas características interferem diretamente na assertividade do modelo;
- preparação dos dados: os dados podem não estar prontos para o uso, assim, sendo necessário o seu pré-processamento, por meio de técnicas de padronização (padronização de unidades), normalização (mesma escala), limpeza (remoção de ruídos e erros), adaptação (problema do dado faltante) ou mesmo balanceamento. Essa etapa compreende ainda a separação do conjunto de dados em treinamento e teste;
- seleção do modelo: existem diversos modelos de aprendizagem de máquina que podem ser empregados, Lenz et al. (2020) cita por exemplo a aprendizagem profunda, a regressão

logística, e o agrupamento;

- treinamento: consiste em treinar o modelo utilizado o conjunto de dados separado para isso. Nessa parte, características são extraídas;
- avaliação: o modelo treinado deve ser avaliado ao que se propõe a fazer. É a forma de se verificar a assertividade do modelo, e poder comparar o desempenho entre modelos;
- ajuste de (hiper)parâmetros: alguns parâmetros que controlam a etapa de treinamento podem ser alterados, tais como épocas e taxa de aprendizagem;
- aplicação: consiste na aplicação de fato do modelo treinado e testado.

As etapas anteriores são complexas, por vezes a preparação dos dados é uma das que mais demanda tempo. Por exemplo, bases de dados com atributos em diferentes unidades de medidas, como a distância em quilômetros e em milhas, que necessitam serem convertidas para uma mesma base, diferentes padrões de resposta necessitam ser padronizadas, dentre inúmeros outros problemas. Também, é importante a escolha das métricas de desempenho. É através delas que se pode comparar os modelos, ou mesmo ajustar parâmetros em um modelo, com a finalidade de melhorar aquela métrica escolhida. O estudo das métricas é feito no final deste capítulo na Seção 2.3.

Este trabalho adotou as seis primeiras etapas dentre as sete apontadas por Lenz et al. (2020). Inicialmente foi construído o *dataset* com as imagens de pessoas sem máscara, com máscara corretamente posicionada e com máscara incorretamente posicionada. No Capítulo 3, foi feito um estudo sobre os trabalhos relacionados e sobre os conjuntos de imagens utilizados, apontando suas limitações. Na Subseção 4.1.1, foi explicado como o *dataset* utilizado foi montado. Depois, na Seção 4.2 foi abordado como se deu a amostragem desse *dataset*, e preparação dos dados. Também foi feita nessa seção a construção das arquiteturas propostas. Em seguida ocorreu o treinamento (Subseção 4.2.2) e avaliação das arquiteturas através do seu conjunto de modelos (Seção 5.2 e Seção 5.3). Esses procedimentos se tornarão mais claros a medida que o leitor progredir na leitura do trabalho.

Géron e Contatori (2019, p.35) propõem outra abordagem, um pouco mais prática, sobre os passos de um projeto de aprendizado de máquina de ponta a ponta, estabelecendo 8 passos. O primeiro é observar o quadro geral, estudar qual é o problema que se busca solucionar, se já existe uma solução vigente e as razões pelas quais ela não atende totalmente. Com essas informações, pode-se pensar em qual será a métrica de desempenho para nortear a comparação de soluções. Outro ponto é verificar as hipóteses feitas, sobre as necessidades, alinhando conhecimento e expectativas entre os envolvidos.

O segundo passo é obter os dados, inicialmente criar um espaço de trabalho, instalando e configurando ambientes de desenvolvimento e bibliotecas. Os dados podem ser oriundos de bancos de dados relacionais, tabelas, arquivos *csv* dentre outros. É importante observar a estrutura e modelagem desses dados, entender o significado dos campos e relacionamentos entre as tabelas, se houver. Géron e Contatori (2019, p.35) sugerem uma exploração inicial nos dados, apenas para se situar. Pode-se utilizar por exemplo medidas estatísticas ou gerar

gráficos (GéRON; CONTATORI, 2019).

Depois disso, deve-se descobrir e visualizar os dados para obter informações mais específicas. Nesse momento a análise será mais aprofundada. Pode se buscar correlações entre os atributos, usando um coeficiente de correlação padrão, gerar matrizes de dispersão, experimentar a combinação de atributos, dentre outros (GéRON; CONTATORI, 2019).

O quarto passo é preparar os dados para os algoritmos de ML. Uma ideia é criar funções para isso, além de otimizar o tempo gasto no processo com preparações recorrentes, isso também auxilia na padronização de procedimentos. Deve-se realizar a limpeza de atributos desnecessários, conversão de tipos categóricos e numéricos, a depender do algoritmo a ser utilizado (GéRON; CONTATORI, 2019).

O quinto é selecionar e treinar um modelo. As etapas anteriores ajudaram a entender qual método de ML poderia auxiliar na solução do problema, se seria uma classificação, ou uma regressão, se existe a necessidade de entender a decisão tomada pelo modelo, ou se a extração de características seria difícil. Depois de escolher o(s) método(s), deve-se escolher como será feita a separação entre conjunto de treinamento, validação e teste, e se ocorrerá apenas uma rodada, ou mais de treinamentos e teste. Ao final desse passo, se terá um ou mais modelos (GéRON; CONTATORI, 2019).

Em seguida, com base no desempenho do modelo, deve-se ajustá-lo, alterando hiperparâmetros, funções de otimização, buscando a melhoria no desempenho do modelo. Nessa etapa utiliza-se o conjunto de validação para esses ajustes. O sétimo é avaliar o modelo em um conjunto de teste. Normalmente o desempenho nesse conjunto é pior, pois o modelo foi ajustado para o conjunto de treino e validação. Mas é justamente uma boa capacidade de generalização do modelo que o fará ter um bom desempenho com dados não vistos antes. Por fim, o último passo é lançar, monitorar e manter o sistema. Nessa etapa é importante monitorar a solução, e o desempenho, verificar se ocorreu uma mudança de cenário, queda de desempenho, ou deterioração (GéRON; CONTATORI, 2019).

Após estudar o processo geral de aprendizado de máquina, é importante entender a divisão entre os tipos de métodos. Eles podem ser divididos nas seguintes categorias (LENZ et al., 2020):

- aprendizagem supervisionada: durante o treinamento, tem-se o conhecimento prévio da saída esperada. Por exemplo, ao se ter um conjunto de fotos de animais (entrada), e a tarefa de classificar a imagem por espécie, por exemplo cães e gatos, os rótulos (*labels*) seriam previamente conhecidos. Esse foi o tipo utilizado neste trabalho. As imagens utilizadas no treinamento eram rotuladas com suas classes (pessoas sem máscara, com máscara corretamente ou com máscara incorretamente posicionada), o rótulo foi inferido pelo nome da pasta em que as imagens se encontravam, como é explicado na Subseção 4.2.2.
- aprendizagem não supervisionada: o rótulo das entradas não é conhecido. Buscam-se características similares nas entradas. Por exemplo, as fotografias não possuem a

identificação da espécie e dessa forma, tem-se de escolher características para agrupar as fotografias, por exemplo quantidade de patas ou cor do pelo.

- aprendizagem semisupervisionada: é uma mistura das duas anteriores. No exemplo das fotografias, teriam-se fotos em que a espécie está identificada e outras em que não.
- aprendizagem por reforço: utiliza recompensas e punições como estímulo ao aprendizado. A tarefa consiste em buscar hipóteses, para diminuir as punições e aumentar as recompensas recebidas ao final das tentativas, quando o modelo apresenta as decisões ou previsões feitas (LENZ et al., 2020). Encontra inspiração em teorias da psicologia humana.

No aprendizado de máquina, existem diversos paradigmas, tais como: o simbólico, o evolutivo ou genético, o estocástico, e o conexionista (LUGER; VIEIRA; TAVARES, 2013). No simbólico, empregam-se representações simbólicas e utilizam-se linguagens formais. No paradigma genético/evolucionário, a inspiração veio das teorias evolutivas e da programação genética, tendo grande importância inspiratória a teoria da seleção natural de Darwin. Já para abordagem estocástica se recorre a ferramentas probabilísticas como a distribuição de probabilidade. Por fim, no conexionista, encontra-se inspiração no cérebro humano e nos neurônios e suas conexões. Nele, durante o processo de aprendizado, a rede neural se adapta ajustando suas ligações, modificando-se de acordo com o conhecimento (LUGER; VIEIRA; TAVARES, 2013). Esse último paradigma é alvo de análise mais profunda, através das rede neurais, devido a ligação com as redes neurais convolucionais e a aplicação na classificação de imagens, tarefa proposta no presente trabalho.

2.1.2 Redes Neurais

O paradigma conexionista encontrou inspiração no cérebro humano. Esse é uma complexa estrutura biológica, composto por neurônios (célula nervosa) que se comunicam através de sinapses nervosas. Ele permite a realização de tarefas como andar, enxergar e falar, que apesar de simples para humanos, são complexas para máquinas. Dessa forma, serviu de inspiração para os métodos conexionistas (FACELI et al., 2021).

Segundo Silverthorn et al. (2017), o Sistema Nervoso é constituído, em análise primária, por neurônios e por células da glia. O neurônico é considerado como uma unidade básica, eles recebem sinais pelos dendritos e emitem pelos axônios. Existem diversos tipos de neurônios que variam em sua forma e constituição (extensão, tamanho dos dendritos e axônios). A região de contato entre um neurônio e seu alvo chama-se sinapse, existem sinapses químicas (maioria) e sinapses elétricas, essa classificação considera o tipo de sinal que passa na região.

As Redes Neurais Artificiais (RNA) buscam de forma artificial recriar mecanismos presentes no cérebro humano. Segundo FACELI et al. (2021), os primeiros estudos remontam à década de 40 com McCulloch e Pitts (1943) e a Unidade Lógica com Limiar, em inglês conhecida como LTU, seguidos por Rosenblatt (1958) e o modelo Perceptron.

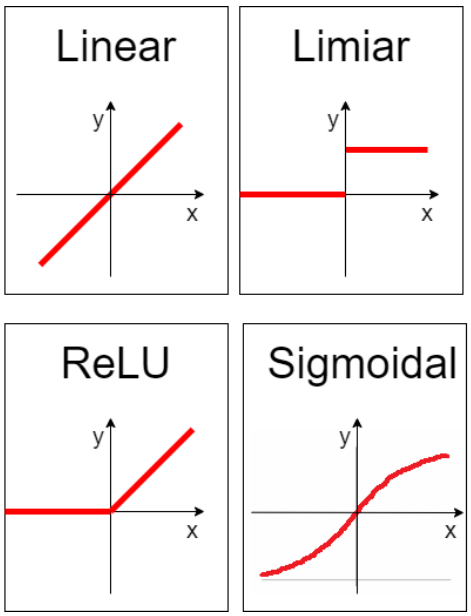
Nesse contexto, surge o neurônio artificial, ele simula o comportamento de um neurônio. Nesta subseção, a palavra neurônio será utilizada para referenciar o neurônio artificial. Assim, o

neurônio recebe sinais de entrada, que são ponderados com o uso de pesos. Esses pesos podem tornar uma entrada mais forte ou fraca a medida que assumem valores maiores ou menores. A Equação (1) ilustra essa combinação, em que *W* representa o peso e *X* o sinal de entrada. Os pesos são ajustados durante a etapa de treinamento.

$$H = \sum_{i=1}^{n} (w_i.x_i) \tag{1}$$

Além dos pesos, aplica-se uma função de ativação à H, e assim, obtém-se a saída. Existem diversas funções de ativação, a Figura 2 ilustra algumas delas. A seguir encontram-se suas definições (FACELI et al., 2021):

Figura 2 – Gráfico de algumas funções de ativação



Fonte: Autoria própria(2022).

- linear: corresponde a uma identidade, isto é, a saída é idêntica à entrada. Sua fórmula é definida na Equação (2).

$$f(x) = a.x \tag{2}$$

- limiar: foi a função de ativação adota no neurônio de McCulloch e Pitts (1943). Assume valores iguais a 1, quando *x* é maior ou igual ao valor adotado como limiar, ou iguais a 0, caso contrário. Essa saída corresponde à ativação ou inativação do neurônio. Sua fórmula é definida na Equação (3), no exemplo adotou-se como limiar o valor 0.

$$f(x) = \begin{cases} 1 & \text{se x} \geq 0 \\ 0 & \text{caso contrário} \end{cases} \tag{3}$$

- sigmoidal: É uma das funções mais utilizadas, por ser diferenciável e contínua. Ela corresponde a uma adaptação da função limiar. Sua saída são valores dentro do intervalo (0,1). Sua fórmula é definida na Equação (4)

$$f(x) = \frac{1}{1 + e_{-x}} \tag{4}$$

- tangente hiperbólica: é um tipo (ou variação) da função sigmoidal. Sua saída está no intervalo (-1,1). Sua fórmula é definida na Equação (5)

$$f(x) = \tanh(x) \tag{5}$$

- função linear retificada, também conhecida como ReLU (do inglês, *Rectified Linear Unit*): é uma das mais recentes segundo FACELI et al. (2021). Ela retorna 0 para valores menores que 0, e retorna o próprio valor para valores maiores ou iguais a 0. Neste trabalho, ela foi utilizada nas camadas convolucionais das duas arquiteturas propostas, como é explicado nas Subsubseção 4.2.2.1 e Subsubseção 4.2.2.2. Sua fórmula é definida na Equação (4)

$$f(x) = \begin{cases} x & \text{se x} \geq 0 \\ 0 & \text{caso contrário} \end{cases} \tag{6}$$

- softmax: segundo Géron e Contatori (2019, p. 279) ela é indicada para tarefas de classificação na camada de saída, em que as classes são mutuamente exclusivas. Por essa razão, ela foi utilizada neste trabalho, na última camada densa nas duas arquiteturas propostas, como é explicado nas Subsubseção 4.2.2.1 e Subsubseção 4.2.2.2. Sua fórmula é definida na Equação (7). Banerjee et al. (2020) explicam que na parte de cima da fração, a função exponencial é aplicada nos elementos de z_i (elementos do vetor z), depois os valores resultantes são normalizados através da divisão pela soma de todas as exponenciais dos elementos de z.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad para\ i = 1,2,\ldots,K \tag{7}$$

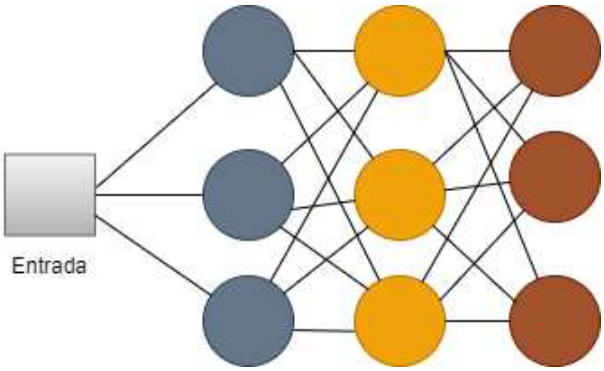
Assim, a saída de um neurônio artificial pode ser descrita como na Equação (8), em que G representa a função de ativação utilizada.

$$S = G(\sum_{i=1}^{n}(w_i.x_i)) \tag{8}$$

Nas RNA, os neurônios artificiais se distribuem entre as camadas. Cada camada pode ter um ou mais neurônios e a RNA pode possuir uma ou mais camadas. Sendo que quando possui mais de uma camada, recebe o nome de multicamada. As camadas podem ser de saída ou intermediárias/ocultas/escondidas, ou ainda de entrada (FACELI et al., 2021). Na Figura 3 é exemplificada uma RNA com 3 camadas, sendo a primeira de entrada (azul), seguida de

uma intermediária (amarela) e por fim uma de saída (marrom). As três camadas do exemplo possuem cada uma três neurônios.

Figura 3 – Arquitetura genérica de uma RNA



Fonte: Autoria própria(2022)

As camadas se comunicam de forma similar ao que ocorre com as sinapses nervosas. Quanto às formas de conexões, tomando como referência um neurônio, ele pode ser completamente conectado, quando recebe/faz conexões de/com todos neurônios da camada anterior ou seguinte, como é o caso de todos os neurônios da Figura 3. Ou pode ser ainda parcialmente conectado, quando ligado a apenas alguns, ou ainda, localmente conectado, quando a apenas alguns em uma região bem definida (LENZ et al., 2020).

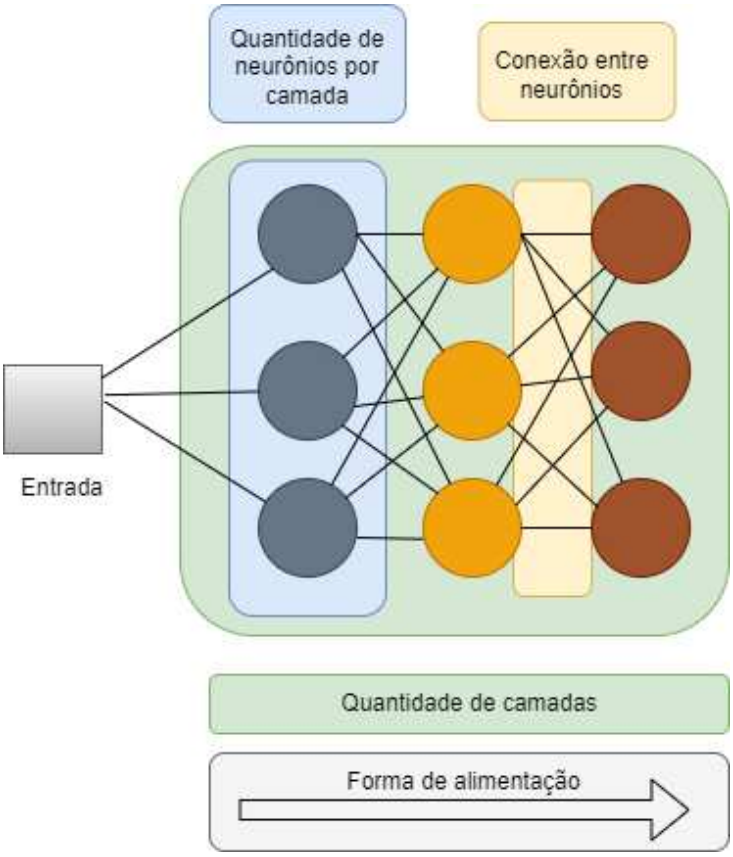
As RNA podem ainda ser do tipo recorrente ou *feedforward*. Nesse, o sinal flui sempre da entrada para a saída. Enquanto que naquele ocorre a retropropagação, isto é, a saída de um neurônio artificial de camada posterior, saída de um neurônio da mesma camada, ou mesmo a própria saída do próprio neurônio, pode servir para alimentar a sua entrada (FACELI et al., 2021).

Dessa forma, as características apresentadas para uma RNA formam a sua topologia. A Figura 4 ilustra essas opções, mostrando a quantidade de neurônios por camada, conexão entre os neurônios, quantidade de camadas e forma de alimentação.

FACELI et al. (2021) explicam que o número de neurônios nas camadas intermediárias de uma RNA depende de questões envolvendo o conjunto de treinamento, tais como volume, quantidade de ruído e distribuição estatística, além da complexidade da função que se deseja aprender. FACELI et al. (2021) explicam ainda que a escolha da arquitetura não é simples, e por vezes se baseia na tentativa e erro, tomando como referência a acurácia preditiva de cada rede testada.

O ajuste dos pesos sinápticos é feito através do algoritmo *Back-propagation*. Ele é baseado no algoritmo do gradiente descendente. Uma das necessidades para seu uso é que a função de ativação seja contínua, diferenciável e, preferencialmente, não decrescente. Uma função que atende essas características é a função do tipo sigmoidal. O treinamento usando o algoritmo *Back-propagation* é dividido em duas fases, a primeira em que o sinal segue para frente, e segunda em que volta (FACELI et al., 2021).

Figura 4 – Escolhas ao montar uma RNA



Fonte: Autoria própria (2022)

No primeiro momento, as entradas são multiplicadas pelos pesos sinápticos e depois se aplica a função de ativação. Esse resultado se torna entrada da próxima camada, que repete o processo. Isso continua a se repetir até que se chegue na camada de saída. O erro é calculado como a diferença entre o valor obtido e o esperado em cada neurônio dessa última camada (FACELI et al., 2021).

Na fase de volta, o erro é utilizado para ajuste dos pesos, conforme a Equação (9). O termo $W^{jl}(t+1)$ identifica o valor do peso entre os neurônios *j* e *l* no momento *t+1*. Já $W^{jl}(t)$ representa o mesmo peso, mas no momento *t*. η indica a taxa de aprendizagem, x^j é a entrada recebida por esse neurônio e σ^l o erro associado ao neurônio *l*. Dessa forma o ajuste do peso considera seu antigo valor, o erro, a taxa de aprendizagem e a entrada (FACELI et al., 2021).

$$W^{jl}(t+1) = W^{jl}(t) + \eta x^j \sigma^l \tag{9}$$

O erro das camadas intermediárias é estimado usando o erro da camada posterior. Assim, o erro de um neurônio *d* pertencente a uma camada intermediária é a soma ponderada dos erros dos neurônios da camada seguinte que se encontram conectados a ele usando como ponderação o valor dos pesos dessas conexões. A Equação (10) mostra como esse erro é

calculado, nela n^l representa um neurônio da camada *l*, o índice *k*, denota os neurônios da camada seguinte ligados a ele, f' representa a derivada parcial da função de ativação dele, por fim E^l representa o erro quadrático desse neurônio e pode ser calculado pela Equação (11) (FACELI et al., 2021).

$$\sigma^l = \begin{cases} f' E^l & \text{se } n_l \in \text{camada\_de\_saida} \\ f' \sum W^{jl} \sigma^l & \text{caso contrário} \end{cases} \tag{10}$$

$$E^l = \frac{1}{2} f' \sum_{i=1}^{k} (y^q - \hat{\mathrm{f}}^q)^2 \tag{11}$$

2.1.3 Aprendizagem profunda

O termo ganhou destaque nos últimos anos como sendo uma solução para problemas difíceis em que a extração de características ou identificação de padrões não parece ser algo simples, como ocorre com dados não-estruturados. Segundo FACELI et al. (2021), as redes profundas (*Deep Networks*), treinadas por algoritmos de aprendizagem profunda (*Deep Learning*) ganharam destaque devido à sua eficiência sobretudo em processamento de imagens e de linguagem natural. FACELI et al. (2021) explicam que já na década de 70 começavam a surgir trabalhos envolvendo o reconhecimento de caracteres em imagens como as RNAs Cognitron (FUKUSHIMA, 1975).

LeCun, Bengio e Hinton (2015) pontuam que o aprendizado profundo melhorou consideravelmente o estado da arte em diversas áreas, como o reconhecimento de fala, reconhecimento de objetos visuais dentre outros. Eles afirmam ainda que esse tipo de aprendizagem permite que modelos computacionais de múltiplas camadas aprendam representações de dados com diversos níveis de abstração. Além disso, o aprendizado profundo permite ainda a descoberta de estruturas complexas em grandes volumes de dados utilizando o algoritmo *backpropagation* para o ajuste de pesos (LECUN; BENGIO; HINTON, 2015).

Abadi et al. (2016) destacam que as redes neurais profundas alcançaram um desempenho inovador em tarefas de visão computacional, como reconhecer objetos em fotografias.

Modelos de aprendizagem profunda utilizam o conhecimento em níveis hierárquicos. A composição simples, não-linear, de níveis mais básicos produz níveis mais abstratos (LECUN; BENGIO; HINTON, 2015). Assim, pensando na classificação de imagens, uma primeira camada pode trabalhar com a extração de características simples, como as bordas, e linhas. A camada seguinte através da composição dessas formas básicas, trabalha com formas mais complexas como os contornos. Assim, a complexidade aumenta camada a camada.

2.1.4 Redes Neurais Convolucionais

Um exemplo de redes profundas são as Redes Neurais Convolucionais, também conhecidas como CNN (do inglês, *Convolutional Neural Network*). Segundo Géron e Contatori

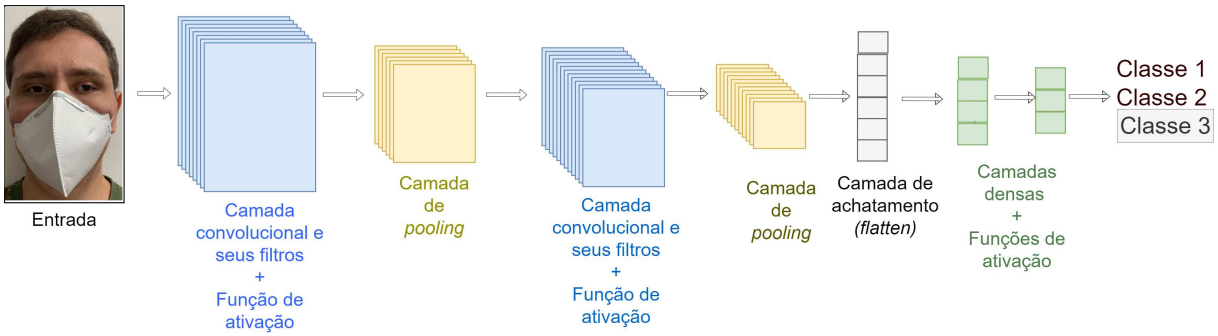
(2019), as CNN surgiram a partir dos estudos sobre o córtex visual do cérebro e ganharam destaque no reconhecimento de imagens desda década de 80. Esses estudos mostraram que os neurônios no córtex visual tem uma área de atuação (reação a estímulos) limitada a um pequeno campo visual, e quando juntos formam o campo visual completo. Também se concluiu que eles não se comportam da mesma forma (reação a estímulos visuais), alguns neurônios são mais sensíveis a algumas formas geométricas (por exemplo linhas verticais ou horizontais) do que outros. Outro ponto importante foi à ideia de que os neurônios de nível superior utilizam as saídas dos neurônios vizinhos de nível inferior. Isto é, alguns neurônios recebem as saídas de outros que reagem a formas básicas (GéRON; CONTATORI, 2019). Essa forma de organização dos neurônios lembra uma arquitetura hierárquica em camadas.

Em 1998, foi proposta a arquitetura LeNet-5 por Yann LeCun, Léon Bottou, Yoshua Bengio e Patrick Haffner (LECUN et al., 1998). A tarefa proposta era reconhecer números em cheques manuscritos. A LeNet-5 contava com camadas totalmente conectadas e funções de ativação sigmoide, além das camadas convolucionais e das camadas de pooling (GéRON; CONTATORI, 2019).

As CNN ganharam destaque em problemas de visão computacional e, após um período de relativa estagnação até meados do ano 2000, voltaram a crescer com o aumento de dados e com o uso de GPUs (acrônimo do inglês *Graphics Processing Units* para unidade de processamento gráfico) (GéRON; CONTATORI, 2019). Devido a sua aplicação em problemas de classificação de imagens, elas foram escolhidas para serem utilizadas neste trabalho.

Uma CNN é composta por uma sequência de camadas, de diferentes tipo. Nas subseções seguintes, elas serão apresentadas. A Figura 5 mostra uma visão geral sobre uma CNN, com a arquitetura sendo bem similar com a que foi desenvolvida neste trabalho.

Figura 5 – Exemplo de uma CNN com suas camadas



Fonte: Autoria própria (2022)

2.1.4.1 Camada convolucional

A convolução é uma operação matricial que combina duas entradas (matrizes), sendo uma a representação matricial de uma imagem e a outra, um filtro. A saída é uma representação matricial de uma imagem processada (GéRON; CONTATORI, 2019).

Observando a Figura 5, é possível perceber que a imagem é a entrada para a 1ª camada convolucional. Isso ocorre através da representação matricial da imagem, que é ligada aos neurônios dessa 1ª camada convolucional. Nessa ligação, surge um conceito importante que são os campos receptivos, área de atuação dos neurônios na representação matricial da imagem. Assim, um neurônio não trabalha com todos os pixels da imagem, e poderá ter pixels em comum com neurônios vizinhos. Nessa primeira camada, ocorre a extração de características básicas da imagem com a aplicação de filtros convolucionais. Os neurônios da segunda camada convolucional e das posteriores também têm uma área de atuação limitada. A primeira camada se concentra em características e formas simples, que vão sendo refinadas de forma hierárquica pelas camadas convolucionais seguintes (GéRON; CONTATORI, 2019).

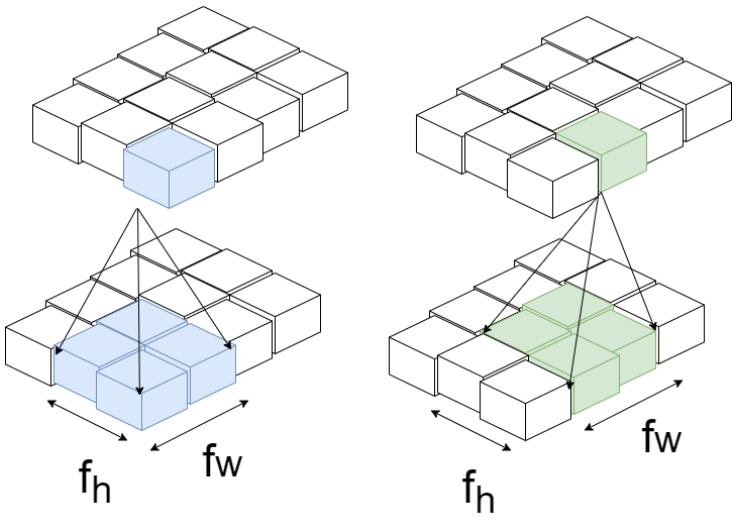
A Figura 6 ilustra esse processo entre camadas. Um neurônio que ocupe a posição (i,j) se ligará com as saídas dos neurônios do intervalo (i : i + f_h - 1, j : j + f_w - 1), em que f_h e f_w representam respectivamente a altura e largura do campo receptivo. Por exemplo, observando a Figura 5, os neurônios da primeira camada convolucional se ligam com a representação matricial da imagem, os neurônios da primeira camada de *pooling* com a saída dos neurônios da primeira camada convolucional, e assim por diante. Perceba que o conceito de campo receptivo se aplica tanto as camadas convolucionais como as camadas de *pooling*, que serão apresentadas na subseção seguinte(GéRON; CONTATORI, 2019).

Perceba que os neurônios destacados de azul e verde, na Figura 6, estão ligados a alguns neurônios em comum. Eventualmente, os neurônios das bordas seriam prejudicados por não possuírem seu campo na camada anterior completo. Para esses casos é comum completar com zero os elementos faltantes (*zero padding*) ou realizar algum procedimento de preenchimento usando medidas estatísticas, como a média (GéRON; CONTATORI, 2019). Quando a camada seguinte é muito menor que a anterior, utiliza-se algo conhecido como *stride*, de forma que seja possível cobrir toda camada. O *stride* seria o deslocamento do campo de receptivo entre neurônios vizinhos, isto é, a distância entre esses campos vizinhos e possui as componentes s_h e s_w (GéRON; CONTATORI, 2019). Na Figura 6, o s_w é igual a 1.

2.1.4.2 Filtros ou *kernels* de convolução

Os filtros são usados para detectar características na imagem. O conjunto de pesos de um neurônio tem o tamanho do seu campo receptivo. Géron e Contatori (2019) citam, por exemplo, o filtro vertical, que seria composto por zeros, exceto em sua coluna principal que contaria com uns. Ao aplicá-lo em todos os neurônios, em relação a uma imagem, as linhas verticais seriam destacadas em detrimento ao resto da imagem. É durante a fase de treinamento que a rede aprende quais filtros são mais importantes. Uma camada convolucional pode ter centenas de filtros. A Figura 7 ilustra a aplicação de dois filtros, um de linhas verticais e outro de linhas horizontais, e foi gerada através da API Keras (KERAS, 2022a) e da biblioteca Numpy (HARRIS et al., 2020). Optou-se por converter as saídas para cinza com a finalidade de facilitar a visualização das características destacadas por cada um dos dois filtros. A seguir

Figura 6 – Relacionamento entre camadas

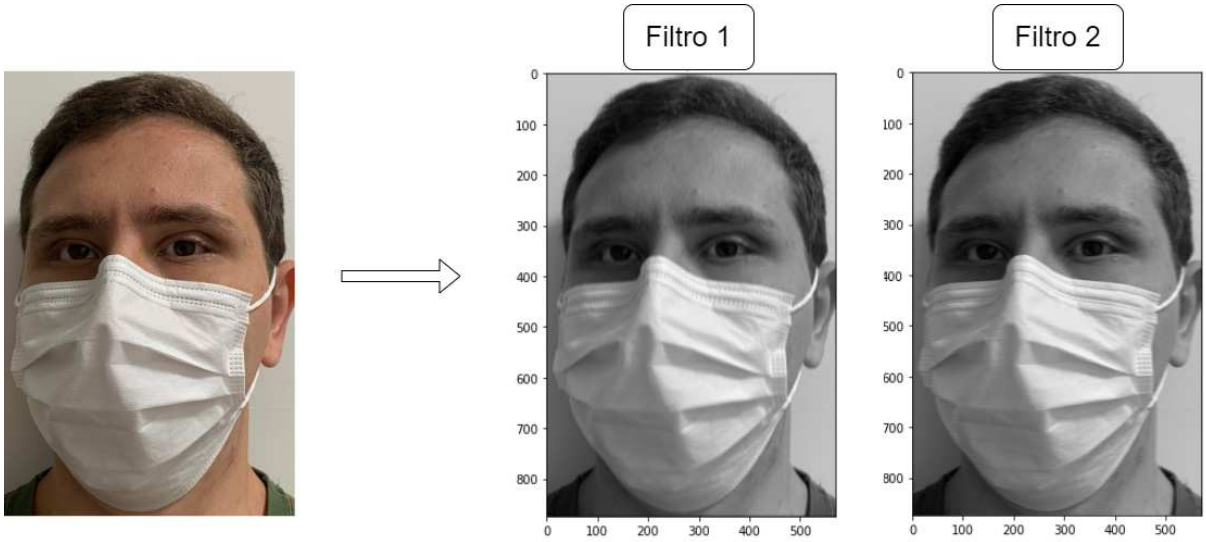


Fonte: Autoria própria (2022)

apresenta-se os comandos para criação dos dois filtros. Depois eles foram passados através de uma camada convolucional para aplicação na imagem de entrada.

- filtro1 = np.zeros(shape=(7, 7, 3, 1), dtype=np.float32)
- filtro1[:, 3, :, 0] = 1 #*linha vertical*
- filtro2 = np.zeros(shape=(7, 7, 3, 1), dtype=np.float32)
- filtro2[3, : , :, 0] = 1 #*linha horizontal*

Figura 7 – Exemplo da aplicação de um filtro vertical e outro horizontal

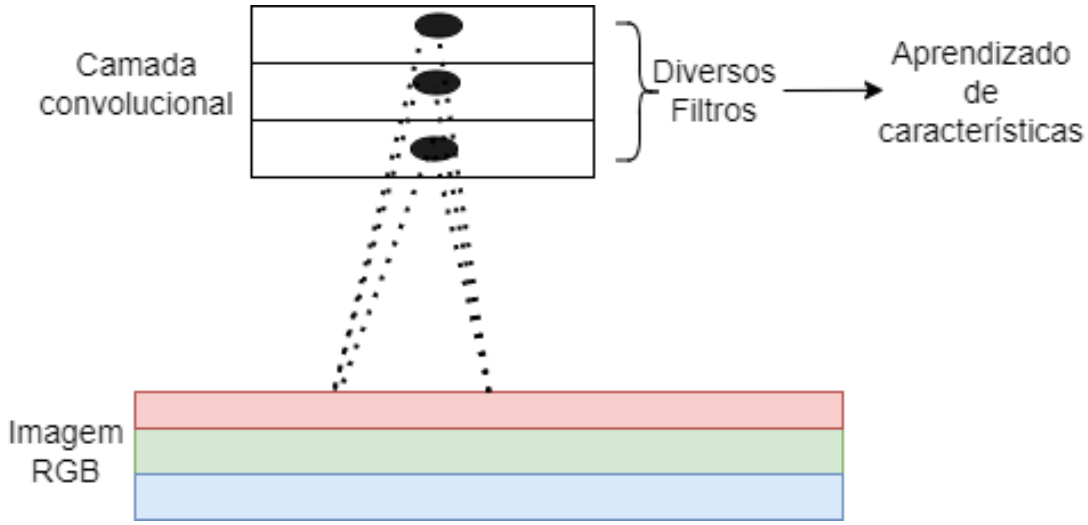


Fonte: Autoria própria (2022)

A utilização de um filtro, em uma camada convolucional, aplicado em uma imagem fornece um mapa de características (GéRON; CONTATORI, 2019). Deve-se observar que uma camada convolucional pode ser representada em 3 dimensões, conforme ilustrado na Figura 8.

Assim, uma camada convolucional é composta por diferentes filtros. Um mesmo mapa possui as mesmas características (pesos e polarizações) (GéRON; CONTATORI, 2019). Uma camada convolucional é capaz de aplicar de forma simultânea diversos filtros às suas entradas. Assim, consegue detectar várias características em qualquer lugar em suas entradas. Neste trabalho, são os filtros, com sua extração de características, que irão ajudar a classificar corretamente as pessoas que estão sem a máscara, as que estão com a máscara corretamente posicionada e as que estão incorretamente posicionada.

Figura 8 – Camada convolucional e seus filtros



Fonte: Autoria própria (2022)

2.1.4.3 Camada *pooling*

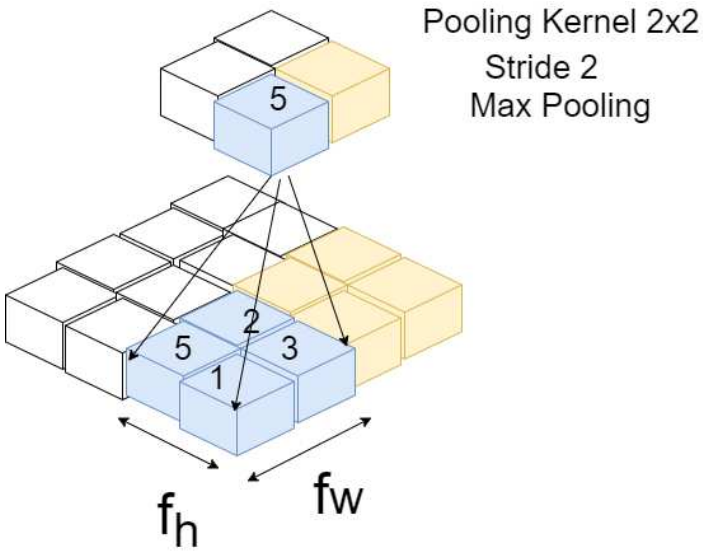
Essa camada serve para sub-amostrar os dados da camada anterior. Para isso, cada neurônio possui um campo de percepção limitado, de forma similar ao que ocorre com as camadas de convolução. Os neurônios dessa camada não possuem os pesos. Normalmente se alternam camadas de convolução e *pooling*. Além disso, essa agregação ocorre pegando o maior valor do campo (*max*) ou a média (*mean*). Elas são importantes por reduzirem o custo computacional a medida que diminuem o volume de dados (GéRON; CONTATORI, 2019).

A Figura 9 exemplifica uma camada de pooling de tamanho 2 x 2, que adota a função de agregação máximo (max), e que possui *stride* igual a 2. Nela, cada 4 entradas retornarão um elemento, reduzindo em 75% a quantidade de elementos.

2.1.4.4 Outras camadas e características

Normalmente, aplica-se uma camada com a função de ativação *ReLU* após cada camada convolucional. Ela serve para além de inserir uma não-linearidade no processo, acelerar a convergência do gradiente (GéRON; CONTATORI, 2019). Após o conjunto de camadas

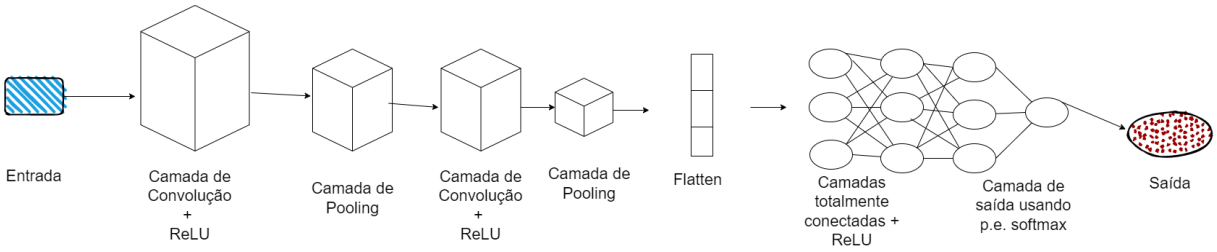
Figura 9 – Exemplo de camada de pooling



Fonte: Autoria própria (2022)

convolucionais e de *pooling*, tem-se uma etapa de achatamento (*flatten*) seguida por camadas totalmente conectadas acrescidas de funções de ativação *ReLU* e, por fim, a camada de saída. Essa última pode empregar uma função como a *Softmax* ou *Sigmoidal* (KERAS, 2022e). A Figura 10 busca exemplificar toda essa arquitetura mencionada. A saída poderia ser por exemplo a classe da imagem.

Figura 10 – Exemplo de uma arquitetura de CNN



Fonte: Autoria própria (2022)

Existem inúmeras arquiteturas de CNN voltadas para os mais diferentes problemas. Elas se distinguem pela quantidade de camadas, funções de ativação e camadas utilizadas.

2.2 Visão Computacional

Para o ser humano, algumas tarefas como enxergar cores e identificar objetos parecem ser algo simples, presente no dia-a-dia. Entretanto, para máquinas isso é desafiador.

A visão computacional pode ser sintetizada como a área capaz de fazer as máquinas/computadores enxergarem. Ela encontra inspiração nas capacidades do sistema de visão

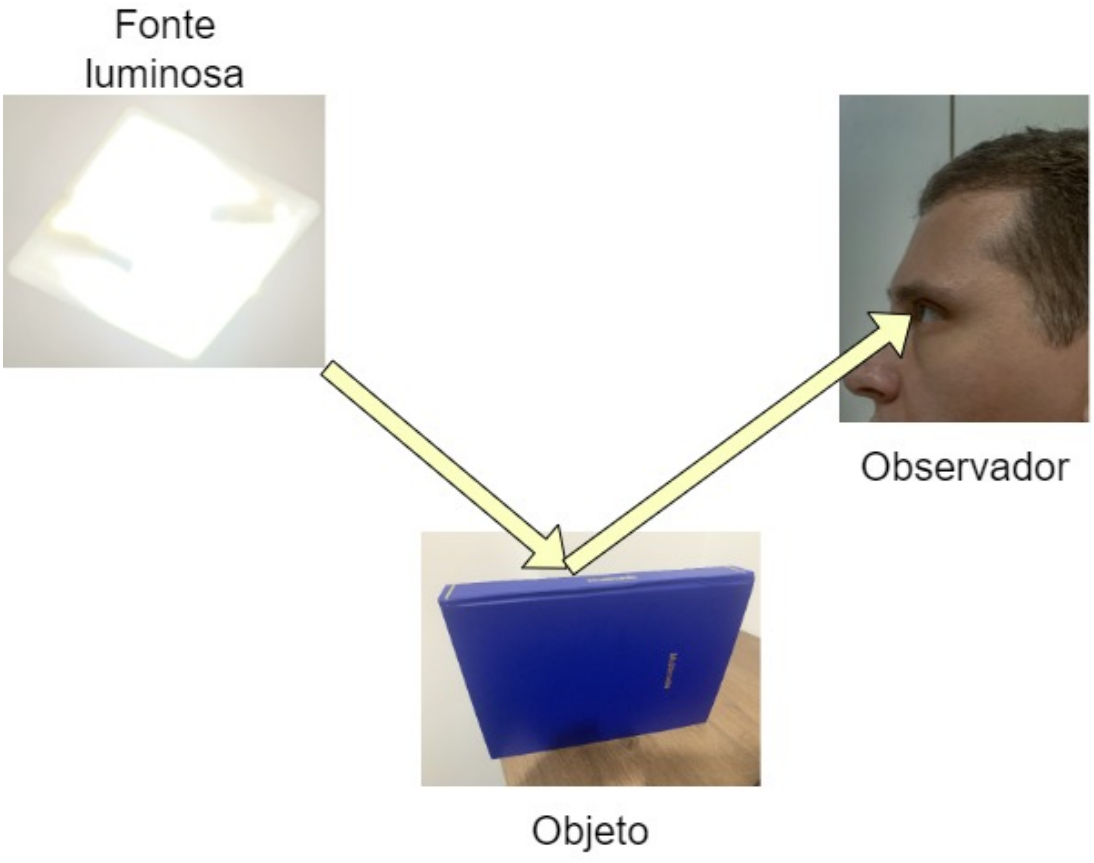
humana. Alguns problemas de estudo dessa área são a detecção, o reconhecimento, a segmentação e a análise de movimento. Entender como a visão humana funciona, como são os sistemas de cores e a resolução de imagens são pontos importantes nessa área (DAWSON-HOWE, 2014).

Para Neves, Neto e Gonzaga (2012), a visão computacional integra as áreas de IA e processamento de imagens e tem como propósito a obtenção de algoritmos voltados para a interpretação do conteúdo visual de imagens. Neves, Neto e Gonzaga (2012) mencionam ainda aplicações da visão computacional envolvendo o reconhecimento de padrões e controle inteligente, e a análise de imagens, em diversas áreas como agronomia, medicina, astronomia, biologia, dentre outras.

2.2.1 A visão humana

Os seres humanos enxergam as imagens devido ao sentido da visão. Nele, ocorre um processo em que a luz refletida no meio externo chega aos olhos, e é traduzida em uma imagem no encéfalo (SILVERTHORN et al., 2017). A Figura 11 ilustra este processo, em que a luz proveniente de uma fonte luminosa incide sobre um objeto, sendo refletida, chegando até um observador.

Figura 11 – A luz e o sentido da visão



Fonte: Autoria própria (2022)

O processo da visão ocorre da seguinte forma: a luz refletida adentra no olho, a pupila varia de tamanho permitindo a entrada de mais ou menos luz, e o cristalino, que funciona como uma lente, focaliza essa luz na retina. Os fotorreceptores localizados na retina fazem a conversão da luz para sinais elétricos, o que se denomina foto-transdução. Os principais fotorreceptores são os cones e os bastonetes. Os cones correspondem a visão mais apurada, são responsáveis pela visão colorida durante o dia, quando há maior presença de luz. Enquanto os bastonetes estão relacionados à visão noturna, menor presença de luz, e permitem enxergar objetos em preto e branco (ao invés de coloridos). A imagem projetada na retina fica de cabeça para baixo, a inversão só é revertida no encéfalo (SILVERTHORN et al., 2017).

Os sinais elétricos oriundos dos fotorreceptores seguem através dos nervos ópticos para o quiasma óptico, no encéfalo. Esses sinais são processados e convertidos em imagens, no córtex visual. Outro ponto é que ocorre o cruzamento das fibras nervosas para o outro lado do encéfalo. Assim, a informação proveniente da parte esquerda é processada no lado direito do encéfalo e vice-versa (SILVERTHORN et al., 2017).

A cor dos objetos depende do comprimento de onda da luz refletida (FILHO, 2011). Na Figura 11, o livro (objeto) reflete ondas de comprimento aproximado de 400 nm (luz azul), por isso é enxergado como azul (sensação de cor azul). Os cones, fotorreceptores relacionados com as cores, podem ser divididos em 3 tipos. Cada qual é mais sensível a um tipo específico de comprimento de onda (*long*, *medium* e *short*).

O Sistema Visual Humano (SVH) possui ainda algumas características fisiológicas e outras perceptuais (percepção/subjetiva) que são relevantes para a visão computacional e o processamento de imagens. Ele consegue detectar apenas uma parte do espectro de luz (comprimento de onda), isto é, apenas a faixa de luz visível. Essa faixa, compreende, aproximadamente, ondas com comprimento entre 380nm e 780 nm. O SVH possui ainda a sensação de variação contínua nos níveis de cinza a partir de 256 níveis, o que corresponde a 8 bits (PAGLIARI, 2018).

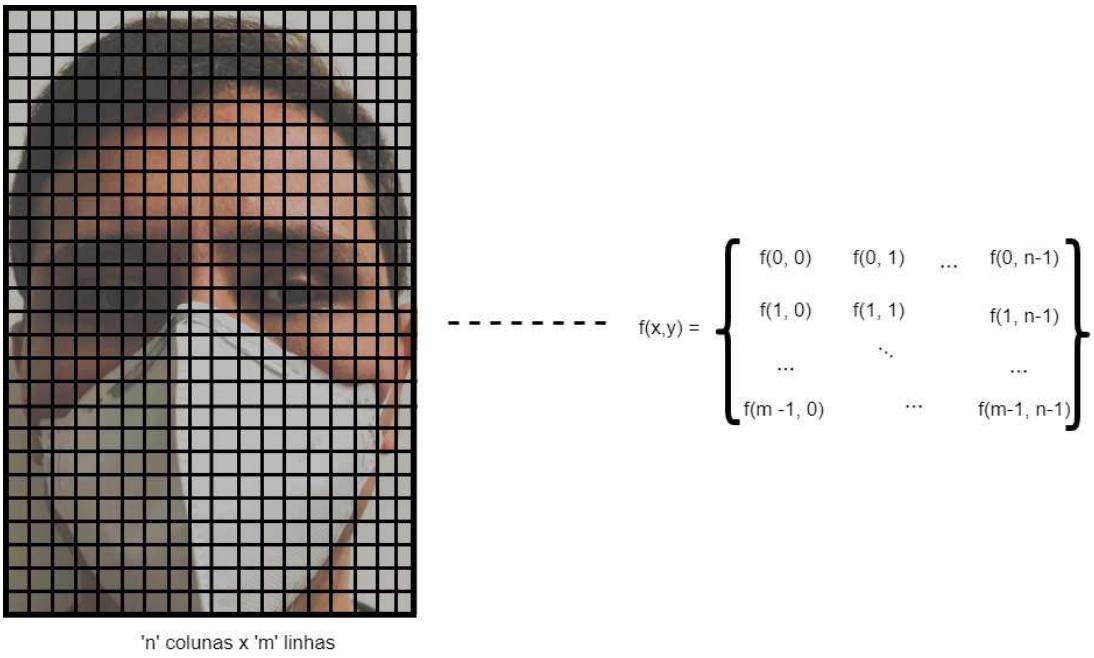
2.2.2 A imagem digital

A imagem digital é considerada como sendo um dado não-estruturado, pois não possui formato e seu tamanho é variável (SANTOS et al., 2021). Dela se pode extrair informações e características. Podendo ser definida como sendo uma função de dois parâmetros espaciais *f(x,y)*, em que x e y são as coordenadas espaciais, enquanto que o valor de *f(x,y)* correspondente equivale a intensidade da imagem naquele ponto de coordenadas (x,y). Frequentemente, uma matriz é utilizada para essa representação (matriz bidimensional quando se trabalha com um canal, ou tridimensional para imagens coloridas). Cada um de seus elementos é chamado de pixel. Segundo Filho (2011), o campo de visão humano equivale a uma matriz de 3000 x 3000.

Na Figura 12, a imagem foi dividida em 'n' colunas e 'm' linhas, o que equivaleria a *n*m* pixels. A imagem original está na resolução 617x906. Caso fosse feita a amostragem da imagem com base no número de linhas e colunas do exemplo, poderia ser atribuído um valor

entre 0 e 255 (caso fosse utilizado 256 níveis de cinza) para cada pixel em cada um dos canais RGB (um acrônimo em língua inglesa para vermelho (*Red*), verde (*Green*) e azul (*Blue*)). A resolução espacial está relacionada com a capacidade de se distinguir detalhes. A Figura 13 compara diferentes resoluções para uma mesma imagem. A fotografia mais a esquerda possui 617 colunas por 906 linhas, e apresenta mais detalhes do que as outras. Já, a mais a direita possui apenas 154 colunas e 226 linhas. Uma maior quantidade de linhas e colunas acarreta na necessidade de um maior espaço de armazenamento, pois haverá mais pixels e informações a serem armazenadas.

Figura 12 – Exemplo da representação matricial de uma imagem



Fonte: Autoria própria (2022)

Para as cores, existem diversos sistemas, dentre eles, o aditivo, o qual é um dos modelos de cores mais utilizados segundo Filho (2011). Ele se baseia na propriedade perceptual de representar as cores através da soma ponderada das luzes vermelha (*Red*), verde (*Green*) e azul (*Blue*) (RGB). Assim, determinada cor poderia ser construída a partir da combinação dessas três. Apesar de possuir a limitação na representação de cores saturadas, essas são raras. Para esse sistema, um eixo de 3 dimensões pode ser imaginado, onde cada um representa uma das cores primárias (vermelho, verde e azul). Um ponto por exemplo, em (1,1,1) representaria a cor branca, enquanto que (1,0,0), (0,1,0) e (0,0,1) vermelho, verde e azul respectivamente. Existem outros sistemas, como o CMY (*cyan-magenta-yellow*), HLS (*hue-luminance-saturation*) e o CIE (*Commission Internationale d'Eclairage*) (FILHO, 2011).

Voltando à imagem digital, para o sistema RGB, tem-se 3 canais, um para cada cor primária (Vermelho-Verde-Azul). Para cada pixel, tem-se L níveis de cor, quanto maior L, mais realista é a imagem. Sistemas de cores verdadeiras podem ter 16 milhões de cores (FILHO,

Figura 13 – Comparação de diferentes resoluções

Fonte: Autoria própria (2022)

2011). Pensando em uma imagem em escala cinza, ao se utilizar 8 bits para L, tem-se um valor entre 0 (preto) e 255 (branco) para cada pixel. Essa faixa de valores (0-255) é comumente utilizada devido à sensibilidade do olho humano (níveis de luminância)(FILHO, 2011).

2.3 Métricas de desempenho

Como foi abordado na Subseção 2.1.1, ao se tratar sobre o processo geral de aprendizado de máquinas, as métricas de desempenho tem um importante papel na avaliação dos modelos.

Após realizar a escolha do *dataset* e realizar o treinamento, ocorre a etapa de teste. Para comparação desses resultados, o que foi feito no Capítulo 5, é necessário o estudo de algumas métricas. Vale destacar que é de interesse deste trabalho as métricas relacionadas ao problema de classificação multi-classe, uma vez, que serão adotadas 3 possíveis classes quanto ao uso da máscara.

2.3.1 Taxa de erro

FACELI et al. (2021) explicam que ela corresponde a uma medida de desempenho usualmente empregada em problemas de classificação, ela apresenta a taxa de instâncias classificadas de forma incorreta. Seu valor pode variar entre 0 e 1. Quanto mais próximo de 0,

mais assertivo é o classificador.

Seja $\hat{f}(x)$ o classificador em análise. Para encontrar a taxa de erro, é utilizada a Equação (12). Nela, a função *I(a)* retorna 1 se a condição *a* for verdadeira e 0 se a condição for falsa. Dessa forma, o valor 1 é somado para cada instância em que o valor predito ($\hat{f}(x)$) difere do valor real (y), por fim o erro total é obtido (FACELI et al., 2021). Ao final, esse somatório é dividido pelo número total de instâncias, o que na Equação (12) corresponde ao *n*.

$$erro(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{f}(x_i)) \tag{12}$$

2.3.2 Acurácia

A acurácia é a taxa de acertos. Ela é uma das principais métricas e informa a proporção de amostras corretamente classificadas em relação à quantidade total (FACELI et al., 2021). Seu valor pode ser encontrado por meio da Equação (13).

$$acuracia(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{f}(x_i)) \tag{13}$$

A acurácia é o complemento da taxa de erro, assim caso já tenha sido calculada a taxa de erro, a Equação (14) pode ser utilizada (FACELI et al., 2021).

$$acuracia(\hat{f}) = 1 - erro(\hat{f}) \tag{14}$$

2.3.3 Matriz de Confusão e métricas relacionadas

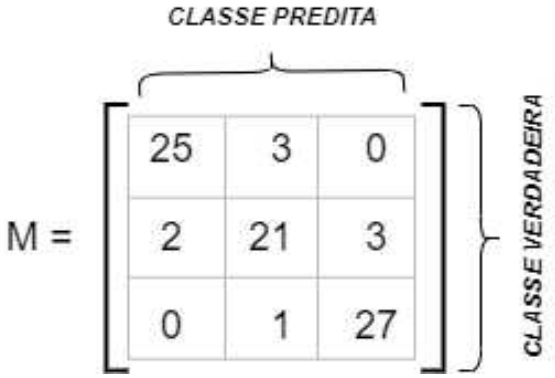
A matriz de confusão é uma importante métrica montada em forma de matriz, permitindo uma visão geral das classificações. Se o problema possui N classes, a matriz será de dimensão *N x N*. Ela irá relacionar a classe verdadeira (linhas) com a classe prevista pelo classificador (colunas). Tomando, por exemplo, o elemento da linha 1, coluna 2, ele representa quantas instâncias foram classificadas como sendo da classe 2, quando em verdade pertenceriam a classe 1. A diagonal principal dessa matriz representa as instâncias corretamente classificadas (GéRON; CONTATORI, 2019).

A Figura 14 exemplifica o que foi dito. Nela, 25 instâncias da classe 1 foram corretamente preditas, 21 e 27 da classe 2 e 3, respectivamente, também o foram. Porém 3 instâncias que deveriam ter sido previstas como sendo da classe 1, foram preditas como da classe 2.

Além das medidas apresentadas, outras são utilizadas para problemas de duas classes. Durante a revisão da literatura, que será apresentada no Capítulo 3, foi possível perceber que muitos autores utilizam essas métricas expandido sua definição para o problema da classificação com mais de duas classes.

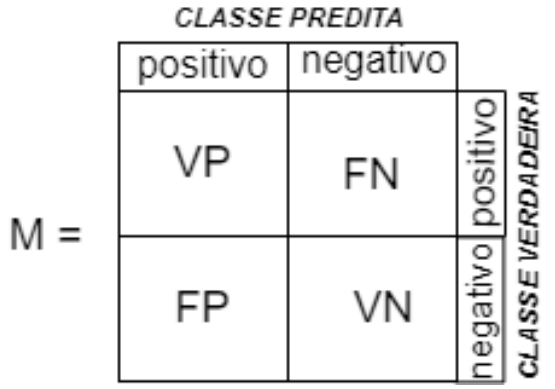
A Figura 15 apresenta essa forma de organizar a matriz de confusão com uma classe positiva e uma negativa. Assim, tem-se (FACELI et al., 2021):

Figura 14 – Exemplo de uma Matriz de Confusão



Fonte: Autoria própria (2022)

Figura 15 – Matriz de Confusão para classe positiva e negativa



Fonte: Autoria própria (2022)

- Verdadeiro Positivo (VP): Corresponde aos valores classificados como pertencentes à classe positiva e que realmente pertencem a ela. Dessa forma, a predição foi correta.
- Verdadeiro Negativo (VN): Corresponde aos valores classificados como pertencentes à classe negativa e que realmente pertencem a ela. Dessa forma, a predição foi correta.
- Falso Positivo (FP): Corresponde aos valores classificados como pertencentes à classe positiva, mas que pertencem a classe negativa. Dessa forma, a predição foi incorreta.
- Falso Negativo (FN): Corresponde aos valores classificados como pertencentes à classe negativa, mas que pertencem a classe positiva. Dessa forma, a predição foi incorreta.

Perceba que o total de instâncias analisadas é dado pela Equação (15)

$$n = VP + VN + FP + FN \tag{15}$$

Na extensão desse conceito, ao analisar uma classe específica, essa é adotada como sendo a classe positiva, enquanto que o conjunto das demais, como classe negativa. Por exemplo, ao analisar o problema que foi proposto neste trabalho, sobre a classificação do uso da máscara,

analisando a classe pessoas sem máscara, esta passa a ser a classe positiva. Os Verdadeiros Positivos seriam os casos de pessoas sem máscara que foram classificados corretamente como sendo pessoas sem máscara. Por outro lado, os casos de pessoas com máscara correta ou pessoas com máscara incorreta que forem classificados incorretamente como sendo pessoas sem máscara serão os Falso Positivos.

A partir do conceito de classe positiva e classe negativa, as seguintes medidas são obtidas (FACELI et al., 2021):

Taxa de erro na classe positiva - Equação (16): São as instâncias classificadas como negativa, mas que pertencem à classe positiva (FACELI et al., 2021). Leva-se em conta todas as instâncias que verdadeiramente pertencem à classe positiva (FN + VP).

$$erro_{+}(\hat{f}) = \frac{FN}{VP + FN} \tag{16}$$

Taxa de erro na classe negativa - Equação (17): São as instâncias classificadas como positivas, mas que pertencem à classe negativa (FACELI et al., 2021). Leva-se em conta todas as instâncias que verdadeiramente pertencem à classe negativa (FP + VN).

$$erro_{-}(\hat{f}) = \frac{FP}{FP + VN} \tag{17}$$

Taxa de acerto ou acurácia total - Equação (18): Corresponde à taxa de exemplos classificados corretamente em relação ao total (FACELI et al., 2021).

$$ac(\hat{f}) = \frac{VP + VN}{n} = \frac{VP + VN}{FN + FP + VN + VP} \tag{18}$$

Precisão - Equação (19): Corresponde à quantidade de instâncias classificadas corretamente como positivas em relação ao total de instâncias classificadas como positivas (FACELI et al., 2021).

$$prec(\hat{f}) = \frac{VP}{VP + FP} \tag{19}$$

Sensibilidade ou revocação - Equação (20): Corresponde à quantidade de instâncias classificadas corretamente como positivas em relação ao total de instâncias que verdadeiramente são positivas (FACELI et al., 2021).

$$sens(\hat{f}) = \frac{VP}{VP + FN} \tag{20}$$

F1 - Equação (21): A precisão (Equação (19)) e a sensibilidade (Equação (20)) são medidas que se complementam, sendo analisadas em conjunto. Isso ocorre pois, a primeira toma como relação o total de predições como sendo positivas, enquanto a segunda, o total que realmente é positiva. Da busca por combiná-las, surgiu a medida F1 que usa uma média harmônica ponderada com peso w. Para o caso de w = 1, tem-se Equação (22) (GéRON; CONTATORI, 2019).

$$F_w(\hat{f}) = \frac{(w+1)(prec * sens)}{(prec + sens)} \tag{21}$$

$$F_1(\hat{f}) = \frac{2 * (prec * sens)}{(prec + sens)} \tag{22}$$

2.4 Amostragem em problemas de classificação

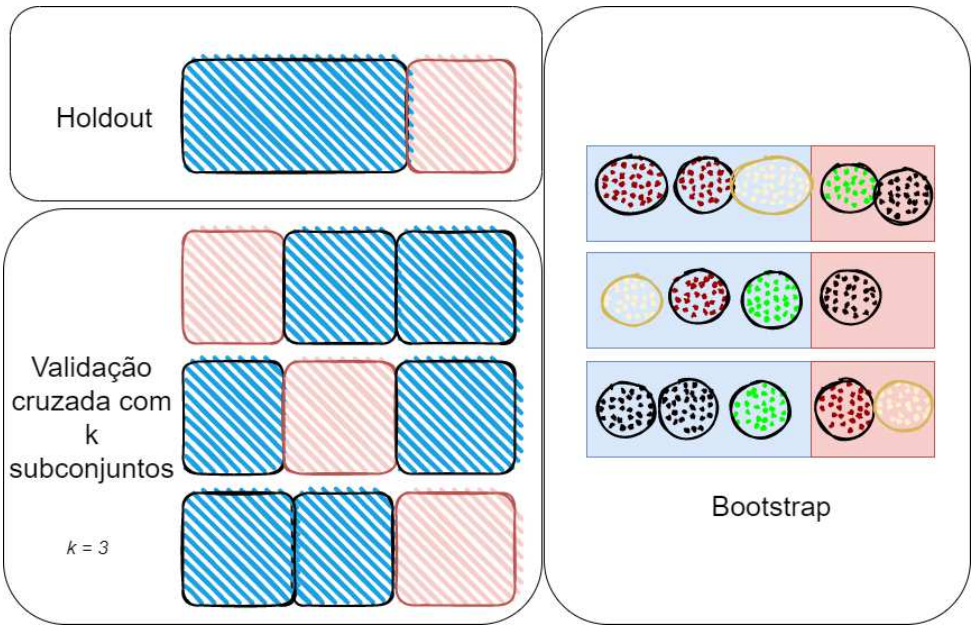
Ao separar o *dataset*, é comum o fazer em dois conjuntos disjuntos, o de treinamento e o de teste FACELI et al. (2021).

Segundo FACELI et al. (2021), a utilização de instâncias do treinamento no teste/avaliação é conhecida como ressubstituição e acaba sendo uma avaliação otimista. Isso por que, durante a fase de treinamento (fase indutiva), o modelo preditivo busca otimizar através de ajustes internos seu desempenho em relação a esse conjunto. Assim, ao submeter instâncias com as quais ele foi treinado, as chances de acerto são maiores.

Em trabalhos envolvendo imagens, esse problema pode ocorrer também de acordo com a similaridade das imagens. Por exemplo, se as imagens do conjunto de teste forem muito similares as do conjunto de treinamento, é mais fácil para o modelo classificar corretamente. A noção de similaridade pode ainda variar de acordo com o algoritmo de classificação utilizado.

Existem diversos métodos de amostragem, conforme ilustrados na Figura 16, em que a cor azul denota o subconjunto de treinamento e a cor vermelha o de teste. Esses métodos serão apresentados nas subseções seguintes.

Figura 16 – Técnicas de amostragem



Fonte: Autoria própria (2022)

2.4.1 Holdout

FACELI et al. (2021) explicam que esse método consiste em dividir o conjunto de dados em duas partes, normalmente na proporção $2/3$ para treinamento e $1/3$ para teste. Essas duas partes são disjuntas, isto é, um exemplar que aparece no conjunto de treinamento não aparecerá no conjunto de teste. Por essa razão, para algumas situações, em que na prática um exemplar poderia aparecer em ambos conjuntos, ela tenderá a ser pessimista. FACELI et al. (2021) acrescentam que uma crítica a esse método é que objetos considerados mais fáceis (de classificar) podem acabar sendo selecionados para o subconjunto de teste, aumentando o desempenho do modelo.

2.4.2 Validação cruzada com *k* subconjuntos

Neste método, também conhecido como *k-fold*, o conjunto de *n* elementos é dividido de forma aleatória em *K* subconjuntos disjuntos, cada um com aproximadamente n/k elementos. Cada um dos *k* subconjuntos é empregado como subconjunto de teste enquanto que os outros *k - 1* são usados como subconjunto de treinamento. O processo é repetido *K* vezes (GOLDSCHMIDT; BEZERRA; PASSOS, 2015).

Goldschmidt, Bezerra e Passos (2015, p.52) dividem a validação cruzada com k subconjuntos em dois tipos, a normal e a estratificada. Explicam que esse último é aplicável aos problemas de classificação, e nele a montagem dos subconjuntos respeita a proporção entre as classes.

FACELI et al. (2021) acrescentam que o desempenho final é calculado como a média dos desempenhos observados em cada uma das rodadas. Pontua ainda, que uma crítica ao método é que os subconjuntos de treinamento das rodadas compartilham objetos em comum, não havendo uma total independência entre as rodadas.

Neste trabalho, a amostragem foi feita da seguinte forma: foram extraídos cinco subconjuntos do *dataset* inicial. Depois disso, cada subconjunto foi dividido em dois grupos disjuntos, o de treino (treino + validação) e o de teste. Os cinco subconjuntos foram utilizados de forma independente. Maiores detalhes são apresentados no Capítulo 4, através da Subseção 4.2.1. A técnica empregada pode ser considerada uma variação do *k-fold*, em termos de diferença, deve-se observar que os cinco subconjuntos montados não são mutuamente exclusivos, e que a soma deles não resulta no conjunto inicial (*dataset* completo).

2.4.3 Bootstrap

Já nesse método, *k* subconjuntos de treinamento são montados através de uma seleção aleatória com reposição de n objetos. Os conjuntos de teste ficam com os elementos não sorteados de cada rodada (FACELI et al., 2021).

3 REVISÃO DE LITERATURA

Neste capítulo foi feita uma revisão dos trabalhos anteriores relacionados com a temática de classificação de imagens referente ao uso de máscara de proteção no contexto da COVID-19. Foi observado como diferentes autores escolheram técnicas de aprendizagem de máquina, quais métricas de desempenho utilizaram, bem como realizaram o treinamento do modelo. Depois de apresentar os trabalhos e suas limitações, foi apresentado o que o presente trabalho tem de diferencial e contribuição em relação aos já existentes.

3.1 Trabalhos anteriores

A visão computacional combinada com a Inteligência Artificial é assunto de diversos trabalhos. Em Abreu, Delfino e Voltan (2018), foi desenvolvido um aplicativo para a retirada de faltas de alunos empregando o reconhecimento facial. Cabe destacar que enquanto a detecção facial visa identificar faces em uma imagem, o reconhecimento facial busca dizer a quem aquela face pertence.

Em situações do dia-a-dia, como a captura de imagens em ambientes abertos ou fechados, é importante primeiro detectar as faces (detecção facial), para depois verificar se as pessoas estão utilizando a máscara corretamente, incorretamente ou se estão sem. Nesse tipo de aplicação, a qualidade da imagem é considerada importante, pois diversos fatores como resolução, iluminação e brilho, por exemplo, podem afetar o desempenho do modelo. No trabalho, Abreu, Delfino e Voltan (2018) utilizaram o algoritmo *haarcascade frontal face* através da biblioteca de código aberto *OpenCV* como forma de detectar a face. Após isso, cada face era classificada como pertencente a um dos alunos da turma.

Com o contexto da pandemia de COVID-19 e seu respectivo impacto global, diversos autores estudaram e propuseram modelos para identificar se as pessoas estariam com máscara. Dos trabalhos disponíveis na literatura, é possível destacar dois grupos que se diferenciam quanto à quantidade de classes. O primeiro considera apenas se a pessoa está com máscara ou sem máscara, havendo assim duas classes possíveis. Já o segundo separa em três classes, isto é, sem máscara, com máscara correta ou com máscara incorreta.

Junior, Teixeira e Homem (2020) desenvolveram um sistema web voltado para o monitoramento da utilização de máscara em locais públicos. Eles investigaram o emprego das Redes Neurais Convolucionais em Cascata Multitarefa (MTCNN) e do método *Haar Cascade* para a classificação de indivíduos com e sem máscara de proteção. A validação contou com uma base de apenas 194 imagens, escolhidas dentre as 1.376 imagens que formam o *dataset* (DATAFLAIR, 2022). Essas imagens foram distribuídas de forma igualitária nas duas classes. Da análise das imagens do *dataset* (DATAFLAIR, 2022) é possível observar que uma máscara branca foi inserida de forma artificial na maioria das imagens, essa máscara não foi adaptada

ao rosto. Além disso, inicialmente, as imagens não se encontram com uma mesma dimensão e resolução. Junior, Teixeira e Homem (2020) utilizaram o *framework* OpenCV para tratamento e codificação das imagens.

Como medida de desempenho, Junior, Teixeira e Homem (2020) utilizaram a precisão, a revocação e $f_\beta - score$. Esses resultados são apresentados através da Tabela 1. Como resultado, o modelo MTCNN obteve uma acurácia de 87,1%, enquanto o modelo Haar Cascade 68,6%. A Tabela 2 e Tabela 3 mostram a matriz de confusão de cada um deles, em que a classe positiva corresponde às pessoas sem máscara.

Tabela 1 – Resultado obtidos.

	Precisão	Revocação	F1-score
Haar Cascade	0,63	0,93	0,75
MTCNN	0,83	0,93	0,88

Fonte: Junior, Teixeira e Homem (2020)

Tabela 2 – Matriz de Confusão - Haar Cascade

		Valor Verdadeiro	
		Classe Positiva	Classe Negativa
Valor previsto	Classe Positiva	90	54
	Classe Negativa	7	43

Fonte: Junior, Teixeira e Homem (2020)

Tabela 3 – Matriz de Confusão - MTCNN

		Valor Verdadeiro	
		Classe Positiva	Classe Negativa
Valor previsto	Classe Positiva	90	18
	Classe Negativa	7	79

Fonte: Junior, Teixeira e Homem (2020)

Sobre os resultados, Junior, Teixeira e Homem (2020) explicam que o emprego das Redes Neurais Convolucionais em Cascata Multitarefa (MTCNN) apresentou melhor resultado que o classificador Haar Cascade em relação à classe negativa, isto é, conseguiu classificar de forma melhor as pessoas com máscara.

Junior, Teixeira e Homem (2020) acreditam que, em relação ao método de MTCNN, o treinamento com outros *datasets* poderia melhorar a precisão do método. Sobre o classificador *Haar Cascade*, apontam a escolha dos hiperparâmetros como uma oportunidade de melhoria.

Ao contrário de Junior, Teixeira e Homem (2020), este trabalho optou pela utilização de três classes, ao invés de duas. Essa escolha se justifica pois, no dia-a-dia, é possível observar

pessoas com a máscara incorretamente posicionada, o que leva a uma baixa eficácia do equipamento de proteção. Além disso, como o modelo não seria treinado com exemplos de máscara incorretamente posicionada, ao se deparar com um exemplo dessa situação, ele poderia classificar imagens entre as classes com ou sem máscara. Outro ponto que este trabalho explorou foi a busca por modelos que apresentem melhores taxas de precisão e acurácia. Do ponto de vista da aplicação, um modelo que não identifique corretamente a situação da máscara, pode trazer uma série de problemas, como impedir o acesso de quem está com a máscara corretamente, ou permitir o acesso de quem está sem máscara.

Outro trabalho relevante foi o de Noreña et al. (2022), em que foi analisado o rendimento de um sistema de Inteligência Artificial das Coisas (do inglês, *Artificial intelligence of things - AIoT*) para detecção, em tempo real, do uso correto, incorreto ou sem máscara baseado em modelos computacionais de nuvem (*cloud*) e de computação de borda (*Edge*). Foram analisados fatores como o local (interior e exterior), desempenho dos algoritmos, tempo de resposta e uso de recursos computacionais.

Noreña et al. (2022) identificaram como sendo o estado da arte, em relação aos algoritmos de detecção de objetos em tempo real voltados para AIoT, os algoritmos baseados em aprendizagem profunda que utilizam a arquitetura CNN. Assim, optaram pelo algoritmo YOLOv3, que realiza a detecção em uma única etapa, sendo caracterizado pela maior velocidade quando comparado com algoritmos como o R-CNN (do inglês, *Region-Based Convolutional Neural Networks*) e o Fast-R-CNN (do inglês, *Fast Region-Based Convolutional Neural Networks*) (NOREñA et al., 2022).

Para o treinamento e teste, Noreña et al. (2022) utilizaram um *dataset* composto por imagens aleatórias de outros 2 *datasets* públicos, o *Kaggle Medical Mask Dataset* (VICH, 2020) e o *MAFA* (GE et al., 2018). Os autores utilizaram a técnica *hold-out* para a divisão do conjunto em treinamento e teste, na proporção 80% (716 imagens) para o primeiro e 20% (179 imagens) para o segundo. Vale destacar que a distribuição das imagens entre as classes não estava balanceada, o que poderia levar o classificador a favorecer a classe majoritária. As imagens possuem uma ou mais faces/pessoas. Assim, o conjunto de treinamento foi composto por 3.070 faces com máscara correta, 675 sem máscara e 113 com máscara incorreta. Durante o treinamento, foi utilizada a transferência de aprendizagem a partir de um modelo YOLOv3-tiny de detecção de oitenta classes de objetos. Também foram apresentados alguns parâmetros da rede convolucional, como tamanho máximo de *batch* ($max\_batches = 30000$), dimensão da imagem (416x416), número de filtros na camada convolucional (24 filtros) e tamanho do *batch* ($batch = 64$ e 2 subdivisões). Sendo que o processo de treinamento durou 16 horas e foi realizado com a ferramenta *Google Colab*. O trabalho documentou de forma primorosa as características do treinamento e arquitetura da rede proposta.

Durante a validação, que contou com 179 imagens, os autores obtiveram as métricas apresentadas na Tabela 4, o que corresponde a uma precisão média de 75,95%.

Outro ponto interessante do trabalho, foi que, após a validação, Noreña et al. (2022)

Tabela 4 – Resultado obtidos na validação por Noreña et al. (2022).

Classe	Precisão
Com máscara correta	85,97%
Com máscara incorreta	73,15%
Sem máscara	68,72%

Fonte: Noreña et al. (2022)

buscaram testar o projeto em um cenário real, separando em dois ambiente, um *outdoor* e outro *indoor*. Esses cenários se diferenciam pelo segundo possuir um fluxo controlado de pessoas e de condições ambientais, como iluminação. Justamente por esse maior controle em um ambiente *indoor*, a precisão foi maior. Os autores também testaram a captação das imagens em duas resoluções, uma alta (1920×1080) e a outra, baixa (640×360). Desse modo, concluíram que a maior resolução permitiu captar objetos pequenos e distantes. Também, concluíram que o modelo *Edge* superou o de *Cloud*.

O modelo *YOLOv3-tiny* adaptado por Noreña et al. (2022) obteve uma precisão menor do que outros modelos. Isso, em parte, pode ser explicado pela utilização de 3 classes que guarda maior dificuldade do que 2 classes. A diferença entre uma pessoa com máscara corretamente posicionada e uma incorretamente por vezes estará em pequenos detalhes, como parte do nariz ou queixo exposto, fato que aumenta a complexidade nesse tipo de classificação com 3 classes.

Assim como feito em Noreña et al. (2022), este trabalho também utilizou três classes, porém ao invés de realizar o controle de ambiente, focou-se no controle de acesso. Dessa forma, a pessoa ficaria com a face frontal a câmera e em condições mais controlada. Outro ponto que diferencia este trabalho foi a busca em se obter resultados de classificação correta melhores.

Costa et al. (2021) destacam que a pesquisa envolvendo a detecção de máscara facial é algo que ganhou importância nos últimos anos. Eles pontuam a relação dessa tarefa de classificação com a área de detecção e de reconhecimento facial. O trabalho desenvolvido por eles tratou da classificação em duas classes, *com máscara* e *sem máscara*. Quanto à implementação, utilizaram a biblioteca *TensorFlow* (ABADI et al., 2015) e a API *Keras* (CHOLLET et al., 2015).

Costa et al. (2021, p. 344) utilizaram parte do *dataset* Wang et al. (2020), extraindo 3.063 imagens, sendo 1.707 de indivíduos sem máscara e 1.356 com máscara. A separação entre o conjunto de treinamento e o de teste/validação foi de 75% (2.297) para o primeiro e 25% (766) para o segundo. Costa et al. (2021) destacam ainda que as imagens possuem diferentes modelos e cores de máscaras. Eles utilizaram um modelo *MobileNetV2* pré-treinado com o *dataset ImageNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Para evitar um *overfitting*, Costa et al. (2021) usaram no treinamento um *Early Stopping*. Durante o treinamento, foram projetadas a realização de 30 épocas, porém ele foi

interrompido com 10 épocas, devido a diminuição de desempenho do conjunto de validação. Ao final do treinamento do modelo, os autores apresentam uma matriz de confusão, que considerou 613 imagens, afirmando ter sido gerada para mostrar a acurácia verdadeira. A Tabela 5 consiste em uma adaptação dela apresentando como classe positiva as pessoas sem máscara. Assim, considerando a classe positiva, a precisão do modelo foi de 99,12% e a acurácia total de 99,02%. Entretanto, Costa et al. (2021) não deixam claro no texto se essas 613 imagens, seriam parte do conjunto de validação/teste, aquele mesmo conjunto que interrompeu a execução de outras épocas (*Early Stopping*).

Tabela 5 – Matriz de Confusão

		Valor Verdadeiro	
		Classe Positiva	Classe Negativa
Valor previsto	Classe Positiva	339	3
	Classe Negativa	3	268

Fonte: Costa et al. (2021)

Já Siradjuddin, Reynaldi e Muntasa (2021) propuseram uma abordagem de detecção a dois estágios. O primeiro objetiva identificar as regiões candidatas (faces) e usa uma *Region Proposal Network* (RPN). Depois disso, essas regiões são classificadas em uma Faster R-CNN. Siradjuddin, Reynaldi e Muntasa (2021) trabalharam com a classificação com 3 classes. Os autores utilizaram dois datasets, o *Mafa dataset* (MANGALAMPALLI, 2020) e o *Annotated Facial Landmarks in the Wild (AFLW dataset)*[1]. Do primeiro, utilizaram 25.876 imagens e do segundo, 14.587. Algumas das imagens possuem mais do que uma face, isto é, um conjunto de pessoas. Assim, obtiveram 17.019 sem máscara, 3.842 com máscara incorreta e 25.610 com máscara correta. O modelo foi testado considerando-se 1.000 faces de cada uma das classes (2.713 imagens). A Tabela 6 apresenta a precisão média em cada uma das classes e compara os métodos Faster R-CNN e Fast R-CNN.

Tabela 6 – Precisão Média por classe - Faster R-CNN x Fast R-CNN

Classe	Faster R-CNN	Fast R-CNN
Sem máscara	0.86	0.52
Com máscara correta	0.78	0.47
Com máscara incorreta	0.57	0.17
mAP	0.73	0.39

Fonte: Siradjuddin, Reynaldi e Muntasa (2021)

Ao analisar os resultados, Siradjuddin, Reynaldi e Muntasa (2021) identificaram que a classe de pessoas com máscara incorreta apresentou no que se refere ao *Precision-Recall*

[1] A URL informada por Siradjuddin, Reynaldi e Muntasa (2021) não se encontra acessível "AFLWData." https://www.tugraz.at/institute/icg/research/teambischof/ lrs/downloads/aflw/.

uma menor curva decrescente, assim como uma menor precisão. Eles consideraram que esse pior desempenho se deve ao *dataset* de treino não estar balanceado, tendo apenas 8% de imagens dessa classe. Assim, sugeriram o aumento e a re-amostragem de dados como possíveis melhorias em trabalhos futuros.

Siradjuddin, Reynaldi e Muntasa (2021, p. 285) apontaram ainda que o melhor desempenho da Faster R-CNN frente à Fast R-CNN se deve ao fato da *regional proposal stage*. Pois no Faster R-CNN, são utilizadas redes profundas, enquanto o Fast R-CNN usa uma busca seletiva com segmentação de imagens. Dessa forma, a camada de classificação no Faster R-CNN recebe apenas a localização do objeto (informação específica), enquanto que no Fast R-CNN, recebe informação geral.

Algo que chama atenção no conjunto de imagens escolhido por Siradjuddin, Reynaldi e Muntasa (2021) é que algumas imagens possuem pessoas em 2º plano ou mesmo com o rosto inclinado ou parcialmente encoberto, situações corriqueiras no dia-a-dia com câmeras. Isso justifica a utilização da detecção facial, para, em um primeiro momento, extrair as faces detectadas na imagem, e depois, tratar o problema da classificação. Essa metodologia se assemelha com a utilizada no problema do reconhecimento facial apresentado no início desse capítulo. Algumas imagens são difíceis de serem classificadas até por humanos, como o caso de uma menina segurando um urso de pelúcia em que o urso encobre parte de sua máscara, ela poderia estar com a máscara corretamente colocada, ou ainda incorretamente. Essa dificuldade também já havia sido apontada por Noreña et al. (2022) ao analisar os ambientes *Indoor* e *Outdoor*. Ambientes *Outdoor* são mais descontrolados, com o trânsito de pessoas em todas direções, ao passo que ambiente *Indoor* podem direcionar o fluxo das pessoas, por exemplo uma entrada e uma saída.

Das, Ansari e Basak (2020) destacam o atual cenário causado pela pandemia do COVID-19 e como isso afetou a vida cotidiana. Além disso, afirmam que o uso de máscaras tornou-se o "novo normal". Eles se propuseram a apresentar uma abordagem simples com a finalidade da detecção de faces e classificação (com ou sem máscara). Além disso, alinhado com a simplicidade, utilizaram bibliotecas básicas de Machine Learning, tais como TensorFlow (ABADI et al., 2015), Keras (KERAS, 2022a), OpenCV e Scikit-Learn.

O trabalho utilizou dois *datasets*, mas sem uni-los. O primeiro, Bhandary (2020), com 1.376 imagens, sendo 690 de pessoas com máscara e 686 sem máscara (DAS; ANSARI; BASAK, 2020). Nele, as imagens de pessoas com máscara foram geradas de forma artificial (sobreposição de uma máscara em uma imagem real) e só há um tipo de máscara. Cada imagem apresenta uma única face. Já o segundo, Maranhão (2020), apesar de possuir as 3 classes, foi adaptado para 2 classes[2] e contou com 853 imagens, os autores não mencionaram a distribuição entre as classes. Em Maranhão (2020) há imagens em que aparecem 1 ou mais pessoas, e com diferentes tipos de máscaras. Os autores explicam que separaram os conjuntos

[2] Não fica claro em Das, Ansari e Basak (2020) como isso foi feito, se a classe incorreta foi descartada ou se foi considerada como sendo uma das outras duas.

em 90% para treinamento e 10% para teste, sendo que o conjunto de treinamento ainda foi separado em treinamento (80%) e validação (20%).

Das, Ansari e Basak (2020) propuseram a aplicação de um classificador em cascata combinado com uma CNN pré-treinada [3], que consistia em duas camadas de convolução 2D conectadas a camadas de neurônios densos. No algoritmo apresentado, inicialmente é realizado um pré-processamento nas imagens utilizando a biblioteca OpenCV. Primeiramente, a imagem de entrada é convertida de RGB para a escala cinza, depois é redimensionada para 100 x 100. Os autores justificam a mudança de cores com a explicação de que muitos sistemas modernos de reconhecimento de imagem que são baseados em descritores funcionam regularmente em imagens em tons de cinza, e não com RGB e que isso ocorre para eliminar informação não essencial. Quanto ao redimensionamento das imagens, explicam que CNN profundas (*Deep CNNs*) requerem um tamanho fixo nas imagens de entrada. Depois disso, a imagem ainda tem seus pixels normalizados e convertidos para um vetor de 4 dimensões (DAS; ANSARI; BASAK, 2020).

Já para a CNN, o algoritmo inicia com a criação de uma camada convolucional 2D com 200 filtros, com seu *kernel* configurado para 3 x 3, seguida pela aplicação de uma função de ativação *ReLU* e uma camada *MaxPooling* com *Pool size* de 3 x 3. Depois outra camada convolucional com 100 filtros, e *kernel* 3 x 3, também seguida por uma *ReLU* e uma *MaxPooling*. Em seguida, há ainda uma camada do tipo *flatten*. Na sequência, foi adicionada uma camada densa com 64 neurônios usando a função de ativação *ReLU* e por fim outra camada densa com 2 saídas para 2 categorias usando a função de ativação *Softmax*. Os autores também utilizaram uma camada de *dropout* com a finalidade de evitar reduzir o *overfitting* (DAS; ANSARI; BASAK, 2020).

Durante o treinamento, Das, Ansari e Basak (2020) utilizaram o método de otimização *Adam* e a função de perda *categorical crossentropy*. A métrica utilizada foi a acurácia e foram executadas 20 épocas.

Quanto aos resultados obtidos, Das, Ansari e Basak (2020) explicam que repetiram os processos de treinamento, validação e teste nos dois conjuntos de imagens em separado. Para o primeiro (BHANDARY, 2020), obtiveram uma acurácia de 95.77% no conjunto de validação. Já para o segundo (MARANHãO, 2020), foi obtida uma acurácia de 94,58% , também no conjunto de validação (DAS; ANSARI; BASAK, 2020). Não foi apresentada a acurácia com base no conjunto de teste.

Ao final do trabalho, Das, Ansari e Basak (2020) apontam que a pesquisa poderia ser expandida para identificar se a máscara está sendo utilizada corretamente ou não, ou ainda o tipo de máscara utilizado e se ela oferece proteção contra o vírus ou não.

A Tabela 7 sintetiza os trabalhos que estudaram o problema da identificação de máscaras em pessoas e que foram expostos nessa seção. As colunas *treinamento* e *validação e teste* apresentam as quantidades de imagens utilizadas. No trabalho de Siradjuddin, Reynaldi

[3] Das, Ansari e Basak (2020) não menciona qual seria esse pré-treinamento.

e Muntasa (2021), foram utilizados dois *datasets*, totalizando 40.463 imagens, sendo que algumas imagens possuíam mais do que uma face [4]. Eles explicam que para o teste utilizaram 3.000 faces, o que não necessariamente corresponde a 3.000 imagens e nada foi falado sobre o tamanho do conjunto de validação. A questão de mais de uma face por imagem também aconteceu com Noreña et al. (2022) e com Das, Ansari e Basak (2020). Para este último no que se refere ao *dataset* Maranhão (2020). Sobre esse trabalho, os autores propuseram o uso de um modelo próprio [5]. Além disso, fizeram uso de dois *datasets* distintos, mas sem uni-los [6]. Também se destaca que o trabalho não apresentou a acurácia com base no conjunto de teste, por isso a separação usando um '-' (o valor a esquerda do - corresponde ao conjunto de validação e o da direita o de teste).

Os conjuntos de dados utilizados nos trabalhos relacionados foram sintetizados na Tabela 8. Cabe destacar que a maioria seguiu uma distribuição equilibrada entre as classes, exceto Wang et al. (2020) que das 95.000 imagens, apenas 5.000 são com máscara.

[4] Por essa razão foi utilizado um * na Tabela 7 .
[5] Por essa razão foi utilizado um ** na Tabela 7 .
[6] Por isso os valores foram colocados separados com uma '/' na Tabela 7 .

Tabela 7 – Trabalhos relacionados.

Referência	Classes	Modelo	Treinamento	Validação e Teste
Junior, Teixeira e Homem (2020)	2	MTCNN	-	194
Noreña et al. (2022)	3	YOLOv3	716*	179*
Costa et al. (2021)	2	MobileNetV2	2.297	766
Siradjuddin, Reynaldi e Muntasa (2021)	3	Faster/Fast R-CNN	40.463*	3.000*
Das, Ansari e Basak (2020)	2	CNN**	1238/77*	248-138/15-85*

Fonte: Autoria própria (2022)

Tabela 8 – *Datasets* utilizados nos trabalhos relacionados.

Nr	Referência	Classes	Total imagens	Trabalho utilizou
1	DataFlair (2022)	com x sem	1.376	Junior, Teixeira e Homem (2020)
2	Vich (2020)	correta x incorreta x sem	1.148	Noreña et al. (2022)
3	Ge et al. (2018)	correta x incorreta x sem	30.811	Noreña et al. (2022)
4	Wang et al. (2020)	com x sem	95.000	Costa et al. (2021)
5	Mangalampalli (2020)	correta x incorreta x sem	30.811	Siradjuddin, Reynaldi e Muntasa (2021)
6	Bhandary (2020)	com x sem	1.376	Das, Ansari e Basak (2020)
7	Maranhão (2020)	correta x incorreta x sem	853	Das, Ansari e Basak (2020)

Fonte: Autoria própria (2022)

Ao final, ficou evidente que, apesar de muitos trabalhos proporem implementações para o problema, em nenhum deles, de acordo com o nosso conhecimento, foram comparados os desempenhos de modelos no que se refere à classificação correta utilizando as três classes (com ou sem máscara, máscara incorreta). Quanto às métricas de desempenho, por vezes, a escolha das imagens interfere nos resultados. Por exemplo, se uma imagem está no conjunto de treinamento, e sua variante (uma rotação ou espelhamento), no conjunto de teste, seria mais fácil para o modelo classificá-la. A própria escolha da arquitetura da rede e pequenos detalhes podem trazer grande impacto para a acurácia do modelo. Além disso, a escolha do *dataset* de treinamento é importante, pois as características são extraídas desse conjunto para ajuste de parâmetros no modelo. Outra questão complexa é a utilização do conjunto de validação durante o processo de treinamento para ajuste de parâmetros, e posteriormente, como forma de se aferir a acurácia do modelo. Não é uma boa prática utilizá-la como métrica de avaliação ao comparar modelos, uma vez que essas imagens já foram apresentadas ao modelo durante o treinamento.

A falta de informações, como a arquitetura da rede, quantidade de camadas, funções de ativação, taxa de aprendizagem, número de épocas, critérios de parada dentre outros dificultam a replicação exata de alguns dos trabalhos.

4 MATERIAIS E MÉTODOS

Neste capítulo foram abordados os materiais e métodos utilizados no trabalho. Inicialmente, na Seção 4.1, o *dataset* escolhido foi explorado, também foi explicado sua montagem a partir de outros *datasets*. Foram abordadas ainda as bibliotecas utilizadas ao longo da pesquisa. Já na Seção 4.2, foram tratados os procedimentos para montagem e separação do conjunto de dados, separação de treinamento e teste, como as arquiteturas foram construídas, os modelos treinados e todas as etapas desde a importação das imagens até a avaliação dos modelos.

4.1 Materiais

4.1.1 Dataset

Até o início da pandemia do novo Coronavírus, não era comum encontrar um *dataset* com pessoas com máscara em contraste com pessoas incorretamente a usando. Durante a pesquisa, não foi encontrado um *dataset* com imagens de pessoas brasileiras, utilizando máscara de forma correta e incorreta e que contivesse um grande volume de imagens. Essas características se justificam devido as características intrínsecas ao treinamento das Redes Neurais Convolucionais e ajuste de parâmetros.

Um *dataset* interessante foi o gerado de forma sintética por Cabani et al. (2021), denominado de *MaskedFace-Net*. O conjunto compreende 137.016 imagens, Cabani et al. (2021) utilizaram o site *Github* [1] para disponibilizar o link de download.

Sobre a forma como o *dataset* foi gerado, Cabani et al. (2021) destacaram que utilizaram o conjunto de imagens de rosto desenvolvido por NVIDIA (2019), denominado *Flickr-Faces-HQ3* (FFHQ). Esse conjunto continha inicialmente 70.000 imagens de alta resolução de rostos humanos em formato de arquivo PNG. Dessas, 177 imagens não foram utilizadas por problemas de oclusão da face. Cabani et al. (2021) destacam ainda algumas características importantes desse conjunto original: variedade de idade das pessoas, etnia, ponto de vista, iluminação e imagem de fundo.

Com base nessas imagens, Cabani et al. (2021) criaram outras de forma sintética, utilizando um modelo deformável máscara para face. Em síntese, se tratou de inserir uma máscara descartável azul nas imagens de face do FFHQ.

Os autores do *dataset* dividiram o conjunto de imagens em duas classes: pessoas utilizando corretamente a máscara e pessoas utilizando de forma incorreta. A máscara é considerada correta quando cobre o nariz, boca e queixo. Incorreta quando: máscara cobrindo apenas o nariz e boca; máscara cobrindo apenas a boca e o queixo e máscara sob a boca.

Um problema do *dataset* MaskedFace-Net é que ele não possui imagens de pessoas sem máscara. Desta forma, neste trabalho, a base de dados foi adaptada por meio da inserção

[1] https://github.com/cabani/MaskedFace-Net

de imagens oriundas do *dataset* Flickr-Faces-HQ3 (FFHQ). Como resultado da fusão dos dois *datasets*, o conjunto resultante contém imagens rotuladas nas seguintes classes:

- Pessoas sem máscara (classe 1),
- Pessoas com máscara incorreta (classe 2),
- Pessoas com máscara correta (classe 3).

Além disso, algumas imagens do *dataset* MaskedFace-Net estavam corrompidas. Assim, essas foram descartadas.

A Figura 17 sintetiza como o *dataset* utilizado no presente trabalho foi montado contendo imagens dos *datasets* de Cabani et al. (2021) e de NVIDIA (2019).

Figura 17 – Como o dataset utilizado foi montado

Fonte: Autoria própria (2022)

Quanto a licença de uso, é importante ressaltar que o *dataset* Flickr-Faces-HQ3 (FFHQ) (NVIDIA, 2019) apresenta imagens em 5 categorias diferentes: Creative Commons BY 2.0, Creative Commons BY-NC 2.0, Public Domain Mark 1.0, Public Domain CC0 1.0 ou U.S. Government Works license. Todas elas permitem o uso gratuito, redistribuição e adaptação para os casos de fins não comerciais. Por sua vez, o *dataset* MaskedFace-Net (uma adaptação do FFHQ) apresenta licença Creative Commons BY-NC-SA 4.0 licenciado pela NVIDIA Corporation que permite o uso e redistribuição para fins não comerciais, desde que sejam citados os trabalhos Cabani et al. (2021) e Hammoudi et al. (2020), além da indicação das modificações realizadas, e que a distribuição dos trabalhos derivados ocorra sob a mesma licença.

A Figura 18 exibe algumas imagens do *dataset* resultante. É possível observar a grande diversidade das pessoas e as 3 classes de imagens existentes. Além disso, nas imagens com máscara, por serem geradas de forma sintética, algumas vezes, a máscara apresenta distorções.

A Tabela 9 demonstra que o *dataset* está bem balanceado quanto às três classes de imagens apresentadas. O *dataset* ocupa um espaço total de 127 GB e contém 203.782 imagens.

Figura 18 – Estrutura do dataset utilizado

Fonte: Autoria própria (2022)

Tabela 9 – Distribuição da imagens do dataset.

	Quantidade	Percentual
Pessoas sem máscara	70.000	34,35%
Pessoas com máscara correta	67.048	32,90%
Pessoas com máscara incorreta	66.734	32,75%

Fonte: Autoria própria (2022)

4.1.2 TensorFlow

Depois de construída a base de imagens, foi necessário encontrar quais ferramentas seriam utilizadas para a construção, treinamento e teste da CNN. Isso foi feito utilizando a ferramenta TensorFlow e a API Keras.

TensorFlow é uma interface voltada para algoritmos de aprendizado de máquina que foi desenvolvida pela empresa Google através da Google Brain, tendo sido lançada em novembro de 2015. Além disso, essa ferramenta é um pacote de código aberto, que utiliza a licença Apache 2.0 (ABADI et al., 2015).

Abadi et al. (2016) explicam que o TensorFlow é capaz de operar em ambientes heterogêneos e de larga escala. Além disso, ele oferece grande suporte a aplicações, com foco tanto em treinamento como inferência em redes neurais profundas. Essa ferramenta é popular em pesquisas de aprendizado de máquina porque possibilita o uso em várias máquinas em um cluster, ou ainda a utilização de CPUs multicore ou GPU, trazendo flexibilidade e otimizando o uso (ABADI et al., 2016).

Abadi et al. (2016) destacam também o importante papel das redes neurais profundas em tarefas de reconhecimento de objetos e a importância que foi dada a elas no TensorFlow.

O TensorFlow possui APIs disponíveis em diversas linguagens de programação, sendo a API voltada para Python a mais completa e fácil de usar. Além disso, a documentação oficial informa que o TensorFlow 2 é compatível com o Python 3.6-3.9 (TENSORFLOW, 2022a).

Géron e Contatori (2019) destacam o poder dessa biblioteca, e explicam que o funcionamento se baseia em grafos de cálculos, para execução em Python e que a biblioteca realiza o processamento deles através de um código em C++ otimizado. Géron e Contatori (2019) explicam ainda que o grafo pode ser dividido permitindo a execução paralela em várias CPUs e GPUs, além de permitir o uso da computação distribuída. Outras vantagens incluem o design limpo, escalabilidade, flexibilidade (funciona em Windows, Linux, MacOS e mobile), além da documentação bem elaborada.

4.1.3 Keras

Keras é uma API de alto nível que utiliza a plataforma TensorFlow 2 e que é compatível com Python 3.6–3.9. Ela foi escrita em Python e é voltada para aprendizagem profunda. Seu foco é ser simples, flexível e poderosa (KERAS, 2022a). Por essas razões foi escolhida para ser utilizada em conjunto com o TensorFlow neste trabalho.

A API Keras permite através de um modelo sequencial a adição de camadas dos tipos convolucional, *max pooling*, ativação, *dropout*, e também normalização em lote (*batch normalization*). A criação de um modelo usando a Keras pode ser dividida em quatro partes principais (MANASWI, 2018):

1. Definir o modelo: um modelo sequencial é criado e as camadas mencionadas anteriormente são adicionadas;
2. Compilar o modelo: é aplicada a função de perda e de otimização;
3. Treinamento: o modelo deve ser treinado;
4. Predições: o modelo gerado é utilizado para realizar predições.

Neste trabalho, a API Keras foi a principal ferramenta utilizada para a criação e o treinamento da rede.

Durante o treinamento da rede, a API Keras utiliza alguns conceitos importantes, os quais são abordados a seguir (KERAS, 2022d):

1. *Batch* (lote): é um conjunto de amostras. Essas amostras são processadas de forma paralela e independente. Para o treinamento, um lote resulta em uma única atualização. Lotes muito grandes tem o inconveniente de levarem mais tempo para serem processados e necessitarem de mais memória.
2. *Epoch* (época): consiste em um corte onde, geralmente, ocorre uma passagem por todo conjunto de treinamento. Os dados de validação são usados para uma avaliação ao final de cada época.
3. *Loss*: é um valor que a rede tenta minimizar através dos ajustes de pesos. Ela é a distância entre a classe real e a classe predita. Em problemas de classificação, como foi o caso deste trabalho, a escolha da classe predita é baseada na probabilidade. Dessa forma, a rede tenta diminuir a probabilidade de atribuir uma baixa probabilidade a classe real. Durante o treinamento, o valor exibido como *loss* se refere ao conjunto de treinamento e o *val_loss* ao conjunto de validação.

4. *Acc*: corresponde a acurácia. Durante o treinamento, o valor exibido como *acc* se refere ao conjunto de treinamento e o *val_acc* ao conjunto de validação.

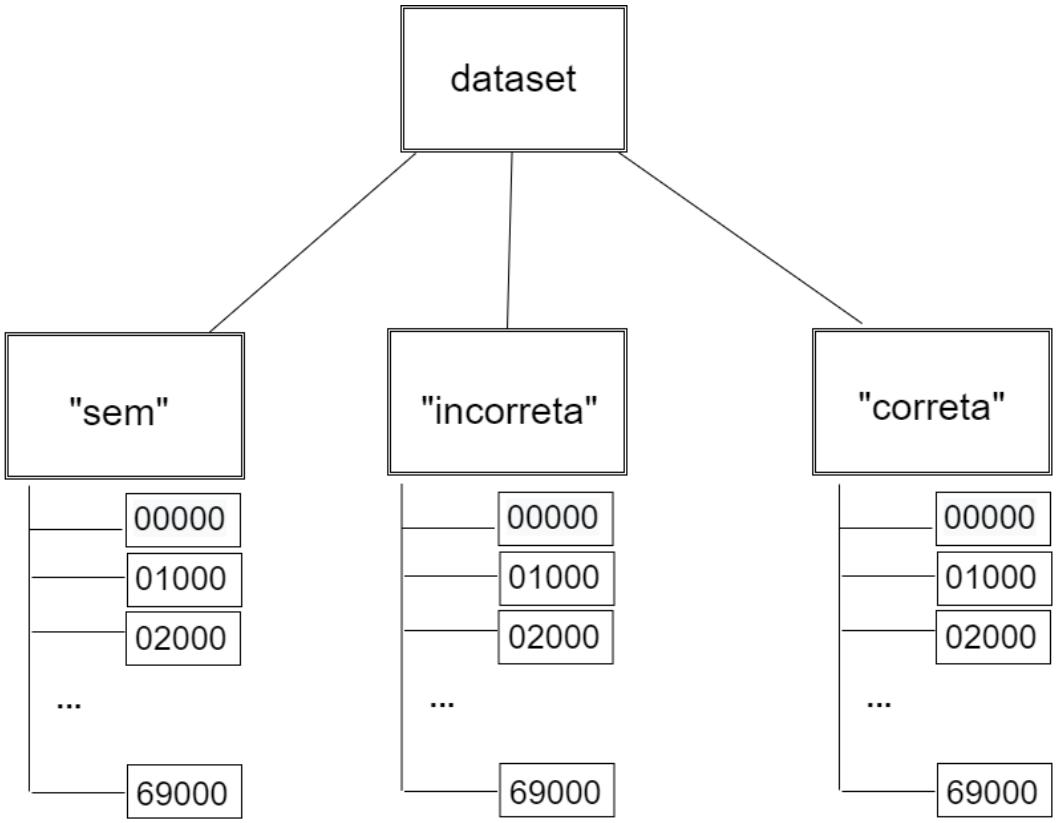
4.2 Métodos

Depois de analisados os materiais utilizados no trabalho, pode-se passar aos procedimentos realizados.

4.2.1 Amostragem do dataset

Como explicado brevemente no Capítulo 2, através da Seção 2.4, para realização dos experimentos, foi escolhida uma técnica de amostragem aleatória (*random subsampling*) similar ao *k-fold*, porém com pequenas diferenças. Foi utilizada a própria estrutura de diretórios em que o *dataset* se encontrava. O *dataset* montado a partir dos outros dois foi separado em 3 pastas, uma para cada classe. Posteriormente, a classe (*label*) das imagens pôde ser extraída através de inferência do nome da pasta. Cada pasta apresentava ainda 70 subpastas (conforme adotado pelo *dataset* original), sendo cada uma com aproximadamente mil imagens. A Figura 19 busca trazer de forma visual a organização descrita.

Figura 19 – Estrutura de pastas da organização das imagens



Fonte: Autoria própria (2022)

Aproveitando a estrutura de pastas, foi possível realizar subamostragens do *dataset* montado. Assim foram montados 5 subconjuntos aleatórios [2], cada um com aproximadamente trinta mil imagens. Deve-se observar que os cinco subconjuntos montados (partições) não são mutuamente exclusivos, e que a soma deles não resulta no conjunto inicial (*dataset* completo). Cada uma dessas partições contou com 10 pastas de cada uma das classes, havendo assim um balanceamento das classes (cada classe com aproximadamente 10.000 imagens). A escolha das pastas foi feita de forma pseudo aleatória, utilizando a biblioteca *random* (FOUNDATION, 2022b). A Figura 20 exemplifica como isso foi feito para a classe 1 e por analogia para as outras classes. O processo foi repetido para cada uma das cinco partições. Ao realizar o sorteio de uma das partições, foram selecionadas 10 pastas por classe. As oito primeiras foram usadas no treinamento e validação, o que corresponde a 80%, e as duas últimas no teste, o que corresponde a 20%. Dessa forma, se garantiu que uma imagem usada no conjunto de testes não era conhecida pelo modelo. Cada partição aleatória compreendeu aproximadamente 14,3% do *dataset* original.

A Tabela 29, disponível no Apêndice A, detalha como foi montada a estrutura de cada subamostragem. Ela permite identificar o nome das pastas sorteadas em cada uma das 3 classes, para cada uma das 5 partições. A utilização de 5 partições buscou diminuir a influência da escolha das imagens e da separação em treino, validação e teste nos resultados apresentados pelos modelos. Por fim, a Tabela 30, também disponível no Apêndice A, apresenta a quantidade de imagens em cada uma das 3 classes, tanto para o treinamento e validação como para o teste. Também é apresentado o percentual de imagens em relação a cada uma das classe. É interessante observar que as imagens estão distribuídas de forma balanceada entre as classes.

As pastas selecionas e suas imagens foram separadas em duas subpastas, uma para o treinamento e outra para o teste. Depois disso, essas imagens foram extraídas para as pastas raízes (treinamento e teste). Dessa forma, dentro das pastas treinamento e teste, havia as pastas 'sem', 'incorreta' e 'correta', as quais continham as imagens. Adicionalmente, houve a necessidade de converter as imagens da pasta "sem", que eram provenientes do *dataset* Flickr-Faces-HQ3 (FFHQ) (NVIDIA, 2019) para o formato jpg (mesmo formato das outras imagens). Para isso, as bibliotecas PIL (LUNDH; CLARK, 2022) e OS (FOUNDATION, 2022a) foram utilizadas.

4.2.2 Etapas do treinamento do modelo

A Figura 21 ilustra de maneira sintética as etapas seguidas desde o apontamento de diretório até a avaliação do modelo. Algumas etapas foram suprimidas com a finalidade de facilitar o entendimento.

Inicialmente, após carregar as bibliotecas, foi feito o apontamento do diretório do conjunto de treinamento da partição que seria utilizada. A quantidade de imagens por classe

[2] Esses subconjuntos foram chamados neste trabalho de partições, por serem partições do *dataset* inicial.

Figura 20 – Montagem das partições aleatórias

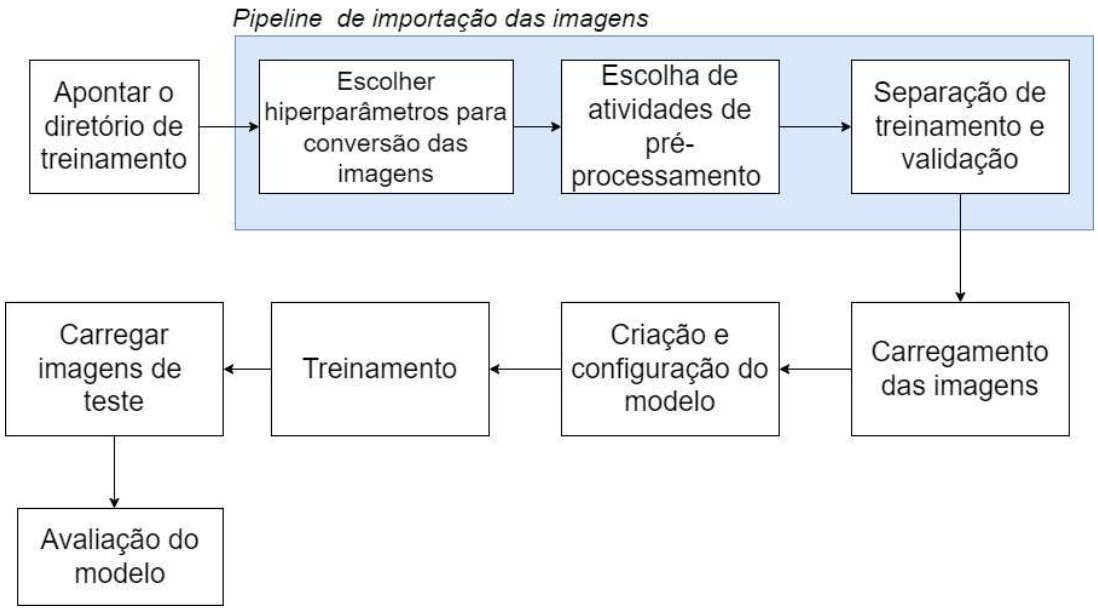
Fonte: Autoria própria (2022)

foi verificada novamente. Foram escolhidos os hiperparâmetros referentes às imagens, como dimensões, canais, tamanho do lote (*batch* e faixa de valor do pixel, conforme a seguir:

1. image_width = 150,
2. image_height = 150,
3. image_color_channel = 3,
4. image_color_channel_size = 255,
5. batch_size = 32.

Ao invés de empregar o utilitário de pré-processamento padrão presente no Keras, isto é, o *tf.keras.utils.image_dataset_from_directory( )*, optou-se por escrever um pipeline de entrada próprio utilizando o *tf.data*. Essa escolha permitiu personalizar e customizar a fase de carregamento do conjunto de treinamento (TENSORFLOW, 2022b).

Figura 21 – Fluxograma da montagem do modelo



Fonte: Autoria própria (2022)

Também, foi escolhida a proporção da divisão das imagens de treinamento em treino (80%) e validação (20%). Durante o carregamento das imagens, ocorreu a extração das classes de cada uma delas, através do nome do diretório em que elas se encontravam, depois, o nome da classe foi convertido em um número inteiro. Também ocorreu o redimensionamento conforme os hiperparâmetros apresentados acima.

Na etapa de configuração do modelo, alguns hiperparâmetros foram passados :

1. taxa de aprendizagem = 0.0001,
2. épocas = 50.

O valor da taxa de aprendizagem e o de épocas são hiperparâmetros. Para o primeiro, um valor muito pequeno faria com que o algoritmo de otimização (*e.g.* gradiente descendente) passasse por muitas iterações até convergir. Por outro lado um valor muito grande, poderia fazer o algoritmo perder o ponto ótimo (mínimo global) (GéRON; CONTATORI, 2019). Quanto a quantidade de épocas, uma quantidade excessiva poderia levar a um *overfitting*, uma pequena a um *underfitting*. Keras (2022b) explica que a taxa de aprendizagem padrão é de 0,001. Neste trabalho, optou-se por testar um valor correspondente a 10% dessa sugestão. Durante o treinamento, os valores de *val_acc, val_loss, loss* e *acc* foram monitorados com a finalidade de verificar se ocorreria um *overfitting*, o que não ocorreu.

Como visto na Subseção 2.1.4, as redes neurais convolucionais possuem inúmeros hiperparâmetros, tais como quantidade de neurônios, tamanho do *kernel*, tamanho da camada convolucional, quantidade de épocas, taxa de aprendizado dentre inúmeros outros. Os hiperparâmetros são parâmetros do algoritmo, e não do modelo (GéRON; CONTATORI, 2019). Justamente por isso, depois de iniciado o treinamento, eles não são ajustados, isto é, não ocorre

ajuste nos hiperparâmetros durante a etapa de treinamento. Uma opção é fixar determinados hiperparâmetros, treinar o modelo, avaliá-lo e compará-lo com outro modelo que utilizou pequenas variações nos hiperparâmetros.

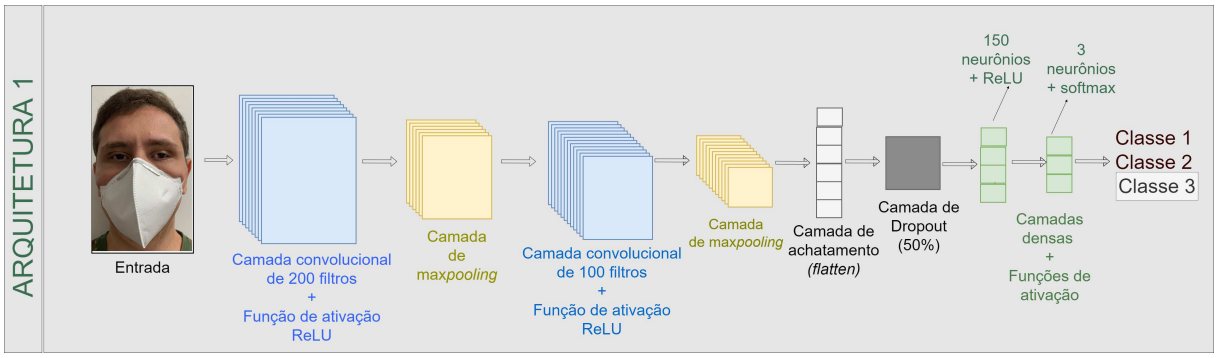
A grande quantidade de hiperparâmetros nas CNN torna essa tarefa muito complicada, além do tempo de treinamento que chega na ordem das dezenas de horas. Este trabalho buscou utilizar valores de hiperparâmetros que são populares na literatura, e variar outros, de forma a comparar os modelos.

Nas subseções seguintes, a construção de cada uma das arquiteturas foi abordada.

4.2.2.1 Arquitetura 1

Para a primeira arquitetura proposta, a rede contou com uma camada convolucional 2D com 200 filtros, *kernel_size*, isto é, a altura e a largura da janela de convolução 2D, de dimensão (3,3) e função de ativação do tipo ReLU. O *padding* utilizado foi o padrão, ou seja, *valid*, o que acarreta em não utilizá-lo. Maiores informações sobre esse tipo de camada podem ser encontrados na documentação oficial da API Keras, através de Keras (2022e). O estudo de Keras (2022e) e Géron e Contatori (2019) auxiliou na escolha dos parâmetros e comentários a seguir. A Figura 22 mostra a arquitetura 1, permitindo sua visualização.

Figura 22 – Arquitetura 1



Fonte: Autoria própria (2022)

A escolha da camada convolucional 2D foi devido ao fato de se estar trabalhando com imagens, a quantidade de filtros é um hiperparâmetro. Por um lado uma quantidade pequena pode não ser suficiente para a extração de características relevantes. Por outro, um valor alto, poderia gerar um excesso de características para as outras camadas, além de demandar mais tempo para ajuste dos parâmetros da rede durante o treinamento. Na arquitetura 2, essa camada foi projetada com 25% dessa quantidade de filtros. O valor do *kernel_size* seguiu uma sugestão de Géron e Contatori (2019, p.382) que mencionam ser um erro a escolha de *kernels* de convolução muito grandes, pois isso implica em um aumento de parâmetros. Quanto a função de ativação ReLU, ela é uma das mais populares (GéRON; CONTATORI, 2019). Quanto ao *padding*, ele se aplica a borda da imagem, entretanto a face de uma pessoa dificilmente estará nessa parte. Assim, não faria sentido seu uso.

Depois dessa camada, foi utilizada uma *MaxPooling2D* (agregação), ela tem o objetivo de diminuir a amostra de entrada por meio da obtenção do valor máximo em uma janela *pool_size*, com dimensão (2, 2). Esse tamanho da janela *pool_size* faz com que ocorra uma redução de *4x* o tamanho inicial (*2x* em cada direção), valores muito altos são muito destrutivos, uma vez que ocorre perda de informação (GéRON; CONTATORI, 2019).

A utilização de uma camada de convolução seguida de uma camada de *Pooling* é uma recomendação presente na literatura, como em Géron e Contatori (2019, p. 381). Após isso, foi utilizada outra camada convolucional similar a primeira, mas com 100 filtros. Essa camada faz uma combinação das características extraídas pela camada convolucional anterior. Em seguida, foi inserida uma camada de *MaxPooling2D*.

Depois, foi utilizada uma camada de achatamento (*Flatten*). Em seguida, para evitar um *overfitting*, foi utilizada uma camada de *Dropout*. Ela é utilizada durante o treinamento, mas não para predições (método *fit*) (KERAS, 2022e). Essa camada foi responsável por selecionar de forma aleatória unidades de entrada a serem descartadas. Utilizou-se uma taxa de *Dropout* de 0.5, dessa forma metade das unidades de entrada dessa camada eram desligadas de forma aleatória.

Em seguida, foram utilizadas duas camadas densas. A primeira com 150 neurônios e a função de ativação ReLU. Essa quantidade de neurônios é um hiperparâmetro, já a escolha da função de ativação se baseou na literatura (GéRON; CONTATORI, 2019). A segunda camada densa, foi projetada com 3 neurônios e com a função de ativação 'softmax'. Géron e Contatori (2019, p. 279) recomenda essa função de ativação na camada de saída para os problemas de classificação, em que as classes são mutuamente exclusivas, como foi o caso deste trabalho. A escolha dos 3 neurônios se deve a quantidade de classes utilizadas no problema.

Durante a compilação, foi passado o otimizador *Adam* e a função de perda (*loss*) *SparseCategoricalCrossentropy*. Essa última calcula a métrica de entropia cruzada (*crossentropy*) entre a classificação real e a prevista, e é utilizada quando se tem duas ou mais classes (KERAS, 2022f). Já aquele otimizador, implementa o algoritmo Adam que é um método estocástico de gradiente descendente, que se baseia na estimativa adaptativa de momentos de primeira e segunda ordem (KERAS, 2022b). A métrica que foi avaliada pelo modelo foi a acurácia. A Figura 23 traz as camadas e quantidades de parâmetros que foram utilizados na arquitetura 1, ela foi obtida com o método *summary()*.

4.2.2.2 Arquitetura 2

Consistiu em uma simplificação da arquitetura anterior. Nesse caso, optou-se por diminuir a quantidade de filtros nas camadas convolucionais de neurônios na penúltima camada densa.

1. Camada convolucional: Conv2D, com 50 filtros, *kernel size* de 3x3 e função de ativação ReLU,
2. Camada de agregação: MaxPooling2D com janela *pool size* de 2x2,

Figura 23 – Arquitetura 1 - summary( )

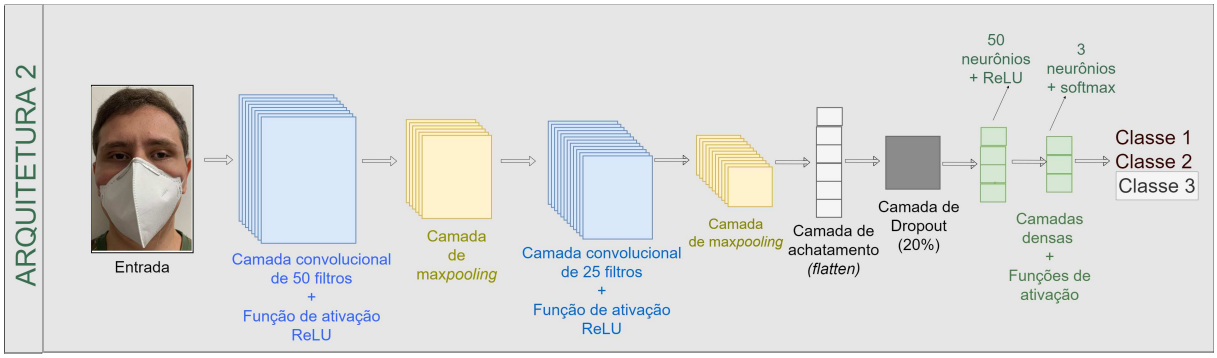
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 200)     5600
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 200)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 100)       180100
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 100)       0
_________________________________________________________________
flatten (Flatten)            (None, 129600)            0
_________________________________________________________________
dropout (Dropout)            (None, 129600)            0
_________________________________________________________________
dense (Dense)                (None, 150)               19440150
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 453
=================================================================
Total params: 19,626,303
Trainable params: 19,626,303
```

Fonte: Autoria própria (2022)

3. Camada convolucional: Conv2D, com 25 filtros, *kernel size* de 3x3 e função de ativação ReLU,
4. Camada de agregação: MaxPooling2D com janela *pool size* de 2x2,
5. Camada de achatamento (Flatten),
6. Camada de *Dropout* com taxa de 0,2.
7. Camada densa com 50 neurônios, e função de ativação ReLU.
8. Camada densa com 3 neurônios e função de ativação *softmax*.

A Figura 24 permite verificar de forma visual a estrutura dessa arquitetura.

Figura 24 – Arquitetura 2



Fonte: Autoria própria (2022)

A combinação de uma camada convolucional seguida por uma de agregação foi mantida nesta arquitetura. Assim como o uso das funções de ativação ReLU e *softmax*. A 1ª camada

convolucional contou com 50 filtros (na arquitetura anterior eram 200), a segunda com 25 (na arquitetura anterior eram 100 filtros). A taxa de *Dropout* também foi diminuída para 0,2 (antes era 0,50) o que equivale a se desligar 20% das entradas dessa camada. Por fim, a primeira camada densa utilizou 50 neurônios, contra 150 da arquitetura anterior. Esse valor também é um hiperparâmetro, cuja escolha se baseia em testes no problema em análise.

Durante a compilação, foi passado o otimizador Adam e a função de perda (loss) *SparseCategoricalCrossentropy*. Além disso, se utilizou a função *EarlyStopping* como forma de interromper a execução de épocas, quando o *val_loss* deixasse de diminuir. Ela foi configurada usando o parâmetro *patience=10*, no qual o treinamento seria interrompido depois de 10 épocas sem diminuição no *val_loss* (KERAS, 2022c). A escolha do valor 10 foi para evitar uma interrupção prematura.

A Figura 25 mostra essa configuração através do comando summary() e exibe o nome da camada, sua dimensão de saída e quantidade de parâmetros.

Figura 25 – Arquitetura 2 - summary( )

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 50)      1400
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 50)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 25)        11275
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 25)        0
_________________________________________________________________
flatten (Flatten)            (None, 32400)             0
_________________________________________________________________
dropout (Dropout)            (None, 32400)             0
_________________________________________________________________
dense (Dense)                (None, 50)                1620050
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 153
=================================================================
Total params: 1,632,878
Trainable params: 1,632,878
```

Fonte: Autoria própria (2022)

Com base na Figura 25, é possível observar ainda que a maior simplicidade da rede acarretou em menos parâmetros para serem ajustados durante o treinamento. Enquanto a arquitetura 1 possuía 19.626.303 parâmetros, a arquitetura 2 tinha 1.632.878, cerca de 8,30% da quantidade da arquitetura 1.

4.2.2.3 Conjunto de teste

Depois de realizar o treinamento dos modelos, as imagens de teste foram carregadas de forma similar ao que foi feito com o conjunto de treinamento. Para comparar os resultados preditos e reais, foi utilizado um *dataframe*, através da biblioteca Pandas (TEAM, 2020). Nele,

cada instância analisada era representada por uma linha. Em seguida foi construída uma função para gerar outro *dataframe* montando uma matriz de confusão. Por fim, foram geradas as seguintes métricas para cada um dos modelos de cada uma das duas arquiteturas: acurácia total do modelo, precisão, revocação e F1-*score*. A forma de cálculo dessas métricas foi apresentada no Capítulo 2, através da Subseção 2.3.3.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos por este trabalho. Inicialmente se aborda o *dataset*, e depois a análise se concentra nas duas arquiteturas propostas, sendo abordados os resultados obtidos por elas através de seus modelos. Depois, a complexidade entre elas é comparada, assim como as métricas de desempenho em classificar as imagens dentre as três classes. Também foi feita uma análise comparativa com os trabalhos relacionados no Capítulo 3.

5.1 Dataset

Durante a Capítulo 3 foram analisados os *datasets* utilizados na literatura, e as limitações existentes. Assim surgiu a necessidade de construir algo novo. Posteriormente, ao longo da Subseção 4.1.1 foi explicada como se deu a criação desse novo conjunto de imagens separado em três classes, isto é, indivíduos sem máscara, com máscara corretamente posicionada e com máscara incorretamente posicionada. Essa criação utilizou a junção de dois outros conjuntos, o MaskedFace-Net (CABANI et al., 2021) e o Flickr-Faces-HQ3 (FFHQ) (NVIDIA, 2019). No primeiro, só havia imagens de pessoas com máscara, podendo estar correta ou incorretamente posicionada no rosto. Já no segundo, pessoas sem máscara. Assim o conjunto resultante apresentou as distribuições de classes apresentadas na Tabela 10 e ilustrados através da Figura 26.

Tabela 10 – Resultados - *dataset* construído.

Classe	Quantidade	Percentual
Sem máscara	70.000	34,35%
Com máscara correta	67.048	32,90%
Com máscara incorreta	66.734	32,75%
Total	203.782	100 %

Fonte: Autoria própria (2022)

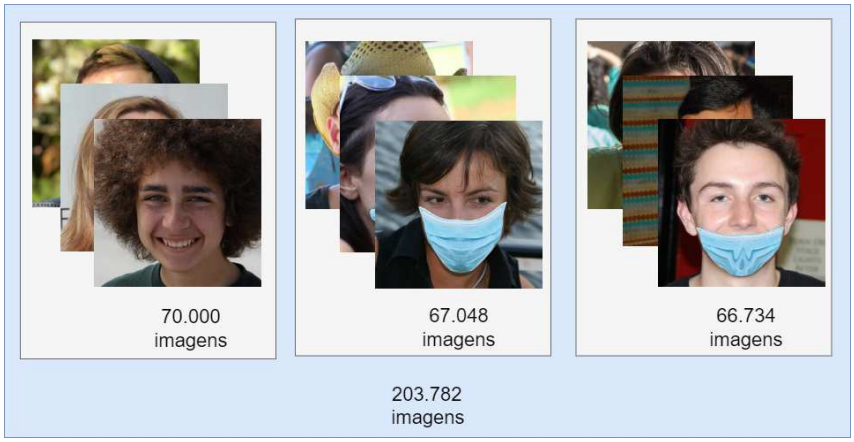
5.2 Modelo 1

Cada um dos cinco modelos foi treinado em 50 épocas. A seguir, a matriz de confusão de cada um deles é apresentada [1].

Essas matrizes mostram a consistência da arquitetura proposta, que apresentou ótimos índices de acerto, mesmo tendo seu conjunto de treinamento e teste alterado. Dessa forma, não

[1] Observar que o nome das classes foi reduzido, onde está a palavra *sem* se refere à classe sem máscara, *incorreta* à classe de pessoas com máscara colocada de forma incorreta e *correta* à classe das pessoas que estavam com a máscara corretamente posicionada.

Figura 26 – *Dataset* construído



Fonte: Autoria própria (2022)

Tabela 11 – Matriz de confusão do modelo 1 - arquitetura 1.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1998	1	1
incorreta	0	1885	17
correta	0	0	1904

Fonte: Autoria própria (2022)

Tabela 12 – Matriz de confusão do modelo 2 - arquitetura 1.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1998	2	0
incorreta	0	1894	4
correta	0	1	1918

Fonte: Autoria própria (2022)

Tabela 13 – Matriz de confusão do modelo 3 - arquitetura 1.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1994	6	0
incorreta	0	1904	10
correta	0	0	1904

Fonte: Autoria própria (2022)

caberia a associação dos resultados ao processo de amostragem do *dataset*. Para as instância da classe *sem*, que foram erroneamente classificadas, na classe *incorreta*, somou-se 13 casos, o

Tabela 14 – Matriz de confusão do modelo 4 - arquitetura 1.

Classe real	Classe predita		
	sem	incorreta	correta
sem	2000	0	0
incorreta	0	1879	3
correta	0	4	1892

Fonte: Autoria própria (2022)

Tabela 15 – Matriz de confusão do modelo 5 - arquitetura 1.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1996	4	0
incorreta	0	1884	21
correta	0	1	1919

Fonte: Autoria própria (2022)

que representa 0,04% em relação ao total de instâncias do conjunto de treinamento de todas as partições. Em similar análise, as instância da classe *sem*, que foram erroneamente classificadas, na classe *correta* não representam 0,004%. Para as da classe *incorreta*, que foram erroneamente classificadas, na classe *correta*, esse percentual não chegou a 0,19%. É interessante notar que nenhuma imagem das classes incorreta ou correta foi atribuída, durante a predição, à classe sem máscara. Dessa forma, é possível concluir que os modelos tenham conseguido gerar filtros robustos em suas camadas convolucionais para conseguir essa separação. Os modelos que mais se destacaram foram os da partição 2 e 4, que apresentaram uma maior acurácia (99,88%).

A Tabela 16 apresenta a acurácia total, precisão, revocação e F1 de cada um dos modelos da arquitetura 1. A penúltima coluna apresenta os valores médios (média aritmética) de cada uma dessas métricas. É possível observar mais uma vez, o bom desempenho dos modelos. Analisando o Desvio Padrão (DP) de cada uma das métricas dos modelos, pode se observar que são valores baixos, em geral, com o primeiro algarismo diferente de zero em sua $2^{\underline{a}}$ ou $3^{\underline{a}}$ casa decimal. Isso significa que os valores obtidos foram muito próximos a média, dessa forma, é possível afirmar que as métricas obtidas foram consistentes.

5.3 Modelo 2

A matriz de confusão de cada um dos modelos é apresentada a seguir.

Mais uma vez, as matrizes de confusão demonstram a consistência dos modelos, mesmo a arquitetura sendo treinada com diferentes conjunto de imagens (diferentes partições do *dataset*), os modelos obtidos tiveram resultados similares. Nesses modelos, também se repetiu algo que aconteceu nos modelos da arquitetura 1, nenhuma imagem da classe incorreta

Tabela 16 – Modelos da arquitetura 1 - Desempenho no conjunto de teste.

	1	2	3	4	5	Média	DP
Acurácia total	0,9967	**0,9988**	0,9972	**0,9988**	0,9955	0,9974	0,0014
Precisão sem	1,0	1,0	1,0	1,0	1,0	1,0	0,0
Precisão incorreta	**0,9995**	0,9984	0,9969	0,9979	0,9974	0,9980	0,0010
Precisão correta	0,9906	0,9979	0,9948	**0,9984**	0,9892	0,9942	0,0042
Revocação sem	0,9990	0,9990	0,9970	**1,0**	0,9980	0,9986	0,0011
Revocação incorreta	0,9911	0,9978	0,9947	**0,9984**	0,9890	0,9942	0,0041
Revocação correta	**1,0**	0,9994	**1,0**	0,9978	0,9995	0,9993	0,0009
F1 sem	0,9995	0,9994	0,9985	**1,0**	0,9990	0,9993	0,0006
F1 incorreta	0,9952	**0,9981**	0,9958	**0,9981**	0,9931	0,9961	0,0021
F1 correta	0,9952	**0,9986**	0,9973	0,9981	0,9943	0,9967	0,0019

Fonte: Autoria própria (2022)

Tabela 17 – Matriz de confusão do modelo 1 - arquitetura 2.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1997	2	1
incorreta	0	1886	8
correta	0	3	1901

Fonte: Autoria própria (2022)

Tabela 18 – Matriz de confusão do modelo 2 - arquitetura 2.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1998	1	1
incorreta	0	1886	11
correta	0	3	1916

Fonte: Autoria própria (2022)

Tabela 19 – Matriz de confusão do modelo 3 - arquitetura 2.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1991	9	0
incorreta	0	1909	8
correta	0	2	1902

Fonte: Autoria própria (2022)

Tabela 20 – Matriz de confusão do modelo 4 - arquitetura 2.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1998	2	0
incorreta	0	1872	7
correta	0	3	1893

Fonte: Autoria própria (2022)

Tabela 21 – Matriz de confusão do modelo 5 - arquitetura 2.

Classe real	Classe predita		
	sem	incorreta	correta
sem	1998	2	0
incorreta	0	1893	9
correta	0	3	1917

Fonte: Autoria própria (2022)

ou correta foi classificada erroneamente como sendo sem máscara. Ocorreram 43 imagens (valor da soma das ocorrências de todas as rodadas) que eram incorretas e que foram classificadas como sendo da classe correta. Esse valor é pequeno quando comparado com a quantidade total de imagens do conjunto de teste de todas as partições (29.052), representa 0,15%. Poderiam ser casos em que a máscara se encontrava levemente mal posicionada, como quando permite aparecer uma pequena parte do nariz ou do queixo. As imagens que pertenciam a classe sem e que foram classificadas como corretas somam apenas 2 casos, o que representa menos de 0,01%. Por fim, as imagens pertencentes a classe sem que foram classificadas como incorretas, somaram 16 casos, o que corresponde a 0,06%.

De forma similar ao que foi feito no modelo 1, a Tabela 22 é apresentada com as métricas em cada um dos modelos e suas médias. A precisão da classe sem máscara foi máxima, o que indica que o modelo 2 consegue diferenciá-la das outras classes. A melhor revocação média ocorreu para a classe correta, o que indica a baixa taxa de FN para essa classe. Os valores de desvio padrão foram baixos, em geral os valores tiveram seu primeiro algarismo não nulo em sua quarta casa decimal. Assim, as diferentes métricas dos diferentes modelos da arquitetura 2 foram consistentes, e bem próximas, tomadas dentro da mesma métrica.

5.4 Comparação dos resultados entre as arquiteturas 1 e 2

Os modelos da arquitetura 2 executaram 21, 24, 29, 21 e 17 épocas respectivamente durante o treinamento, o que equivale a uma média de 44,8% a quantidade de épocas executadas pelo treinamento dos modelos (da arquitetura) 1 (50 épocas). Assumindo que o tempo médio por treinamento de cada época é aproximadamente o mesmo, uma vez que a quantidade de

Tabela 22 – Modelos da arquitetura 2 - Desempenho no conjunto de teste.

	1	2	3	4	5	Média	DP
Acurácia total	0,9976	0,9973	0,9967	**0,9979**	0,9976	0,9974	0,0005
Precisão sem	1,0	1,0	1,0	1,0	1,0	1,0	0,0
Precisão incorreta	0,9974	**0,9979**	0,9943	0,9973	0,9974	0,9969	0,0015
Precisão correta	0,9953	0,9938	**0,9958**	0,9963	0,9953	0,9953	0,0009
Revocação sem	0,9985	**0,9990**	0,9955	**0,9990**	**0,9990**	0,9982	0,0015
Revocação incorreta	0,9957	0,9942	0,9958	**0,9963**	0,9953	0,9955	0,0008
Revocação correta	0,9984	0,9984	**0,9989**	0,9984	0,9984	0,9985	0,0002
F1 sem	0,9990	**0,9995**	0,9977	**0,9995**	**0,9995**	0,9990	0,0008
F1 incorreta	0,9966	0,9960	0,9950	**0,9968**	0,9963	0,9961	0,0007
F1 correta	0,9969	0,9961	**0,9974**	**0,9974**	0,9969	0,9969	0,0005

Fonte: Autoria própria (2022)

instâncias de treinamento e validação são as mesmas, assim como o tamanho do *batch*, é possível concluir que os treinamentos dos modelos 2 foram mais rápidos que dos modelos 1. Também auxilia nessa conclusão, a arquitetura 1 ter 19.626.303 parâmetros treináveis, enquanto que a arquitetura 2 possui apenas 1.632.878, ou seja, 8,3% a quantidade de parâmetros.

A Tabela 23 apresenta os valores médios de cada uma das duas arquiteturas. É possível observar que mesmo sendo uma arquitetura mais simples, conforme abordado no parágrafo anterior, os modelos da arquitetura 2 apresentaram, em média, métricas de desempenho muito próximas as da arquitetura 1, ora um sendo melhor para uma classe, ora sendo o outro. Tanto a precisão média, como a revocação média de ambos conjuntos de modelos foram a mesma, 99,74%. O F1-score médio dos modelos 1 foi melhor em apenas 0,01%. Quanto a constância dos modelos, a arquitetura 2 mostrou menores valores de desvio padrão, o que mostra que seus modelos obtiveram métricas mais próximas.

Tabela 23 – Comparação da média dos modelos.

	modelos da arquitetura 1	modelos da arquitetura 2
Acurácia total	0,9974	0,9974
Precisão sem	1,0	1,0
Precisão incorreta	**0,9980**	0,9969
Precisão correta	0,9942	**0,9953**
Revocação sem	**0,9986**	0,9982
Revocação incorreta	0,9942	**0,9955**
Revocação correta	**0,9993**	0,9985
F1 sem	**0,9993**	0,9990
F1 incorreta	0,9961	0,9961
F1 correta	0,9967	**0,9969**

Fonte: Autoria própria (2022)

Além da comparação utilizando a média das métricas, é possível comparar o melhor modelo da arquitetura 1 com o melhor da arquitetura 2, sendo a acurácia total o critério para escolha do melhor modelo. A Tabela 24 mostra essa comparação. Como existem dois modelos da arquitetura 1 com o mesmo valor de acurácia, ambos foram selecionados. O modelo 4 da arquitetura 1 tem quase todas métricas melhores que o modelo 4 da arquitetura 2, isso só não ocorre com a revocação da classe correta. De forma similar, o modelo 2 da arquitetura 1 também possui melhores métricas que o modelo 4 da arquitetura 1, exceto para o F1-score da classe sem máscara.

Tabela 24 – Comparação entre os melhores modelos.

	modelo 2 arquitetura 1	modelo 4 arquitetura 1	modelo 4 arquitetura 2
Acurácia total	**0,9988**	**0,9988**	0,9979
Precisão sem	1,0	1,0	1,0
Precisão incorreta	**0,9984**	0,9979	0,9973
Precisão correta	0,9979	**0,9984**	0,9963
Revocação sem	0,9990	**1,0**	0,9990
Revocação incorreta	0,9978	**0,9984**	0,9963
Revocação correta	**0,9994**	0,9978	0,9984
F1 sem	0,9994	**1,0**	0,9995
F1 incorreta	**0,9981**	**0,9981**	0,9968
F1 correta	**0,9986**	0,9981	0,9974

Fonte: Autoria própria (2022)

5.5 Comparação dos modelos das arquiteturas 1 e 2 e dos modelos da literatura

A comparação utilizou a média das métricas dos modelos da arquitetura 1 e da arquitetura 2. Tendo em vista que o desvio padrão entre os modelos é baixo, conforme visto em Tabela 16 e Tabela 22, isso não é um problema. A ideia nesse tópico foi considerar a arquitetura, por isso a utilização da média. A partir desse momento, a palavra modelo médio 1 se refere a média dos modelos da arquitetura 1, raciocínio análogo vale para o modelo médio 2.

Ao comparar os resultados obtidos pelos modelos das arquiteturas 1 e 2 com os obtidos por Junior, Teixeira e Homem (2020), é possível notar que as arquiteturas e técnicas ora proposta neste trabalho foram mais eficiente. Os modelos de Junior, Teixeira e Homem (2020) trabalharam com duas classes, o que deveria beneficiar a acurácia e demais métricas. Entretanto, isso não se observou. O modelo de detecção de faces do algoritmo *Haar Cascade* empregado por Junior, Teixeira e Homem (2020) não é voltado para a detecção de objetos, como é o caso de uma máscara. Também não há indicativo de que os modelos propostos por Junior, Teixeira e Homem (2020) tenham sido treinados com imagens de pessoas com e sem máscara. Além disso, a quantidade de imagens selecionadas do *dataset* DataFlair (2022) é

pequena, apenas 194 imagens, o que poderia influenciar nos resultados obtidos. A Tabela 25 mostra a comparação entre a média dos modelos da arquitetura 1 e 2 com os modelos Haar Cascade e MTCNN. As acurácias dos nossos modelos médios 1 e 2 são 31% a mais do que a do Haar Cascade e 13% do que o MTCNN.

Tabela 25 – Comparação dos modelos 1, 2 e Junior, Teixeira e Homem (2020).

	Modelo 1	Modelo 2	Haar Cascade	MTCNN
Acurácia total	99,74%	99,74%	68,6%	87,1%
Precisão média	99,74%	99,74%	63%	83%
Revocação média	99,74%	99,74%	93%	93%
F1 média	99,74%	99,73%	75%	88%

Fonte: Autoria própria (2022)

Sobre o desempenho dos modelos propostos por Junior, Teixeira e Homem (2020), os próprios autores fizeram uma análise crítica em seu trabalho e já haviam informado que em relação ao método de MTCNN, o treinamento com outros *datasets* poderia melhorar a precisão do método. Sobre o classificador *Haar Cascade*, apontaram a escolha dos hiperparâmetros como uma oportunidade de melhoria.

Este trabalho utilizou um conjunto de imagens consideravelmente maior do que Junior, Teixeira e Homem (2020), somando a quantidade das 5 partições, foram 116.563 imagens para treinamento e 29.052 para teste, contra apenas 194 imagens utilizadas para validação em Junior, Teixeira e Homem (2020). A maior quantidade de imagens, além de ser uma sugestão de Junior, Teixeira e Homem (2020), também é apontada na literatura, como Lenz et al. (2020, p. 18), ao abordar as etapas do processo de aprendizagem de máquinas e afirmar que a qualidade e quantidade dos dados é muito importante para a extração do conhecimento, e que quanto maior o *dataset* melhor o aprendizado. Nesse sentido, FACELI et al. (2021, p. 30) também afirmam que o tamanho da amostra é muito importante para representar o problema e que uma amostra pequena pode não o fazer adequadamente. Uma quantidade pequena de imagens no conjunto de avaliação, pode acarretar distorções.

Noreña et al. (2022) desenvolveram um modelo voltado para AIoT, baseado no algoritmo YOLOv3. Foi utilizada por eles, a técnica *hold-out* e as classes não estavam balanceadas. Eles também trabalharam com a classificação em 3 classes. A Tabela 26 compara a precisão de cada uma das três classes entre os modelos. É possível observar que o modelo proposto por Noreña et al. (2022) teve um desempenho inferior em todas elas. Essa diferença de desempenho foi maior na classe sem máscara, um motivo poderia ser o *dataset* desbalanceado utilizado por Noreña et al. (2022). Nele, considerando as faces (e não a quantidade de imagens), 79,6% eram de pessoas com máscara correta, 17,5% sem e 2,9% incorreta.

Outro ponto, que pode explicar tamanha diferença é a pequena base de imagens utilizada por Noreña et al. (2022), 716 imagens. Embora Noreña et al. (2022) mencionem que o modelo utilizou transferência de aprendizagem e era pré-treinado para reconhecer oitenta classes,

isso pode não ter sido suficiente para os ajustes de pesos nos filtros das camadas convolucionais, sobretudo se o modelo original não possuísse ligação com a tarefa de classificação de faces e máscara de proteção. Também deve-se observar que a tarefa da forma como modelada por Noreña et al. (2022) é mais desafiadora, uma vez que admite em uma imagem diversas faces, em algumas vezes a face está pequena, e em posição diferente da frontal. Essa maior quantidade de possibilidades em relação a posição do rosto exige um aumento na quantidade de imagens do conjunto de treinamento para que durante o treinamento seja possível realizar a extração de características desses diversos padrões. A classe que teve uma menor diferença na comparação entre as precisões, foi a classe correta, com cerca de 13,5%. Essa é justamente a classe majoritária no conjunto de imagens utilizado por Noreña et al. (2022).

Tabela 26 – Comparação dos modelos 1,2 e Noreña et al. (2022).

	Modelo 1	Modelo 2	YOLOv3
Precisão classe *correta*	99,42%	99,53%	85,97%
Precisão classe *incorreta*	99,80%	99,69%	73,15%
Precisão classe *sem*	100%	100%	68,72%

Fonte: Autoria própria (2022)

Costa et al. (2021) trataram do problema com duas classes, utilizaram a separação dos conjuntos de treinamento e de teste na porcentagem 75% e 25%, trabalhando com 2.297 imagens para o treino e 766 para o teste/validação. Quanto ao algoritmo, utilizaram o *MobileNetV2* pré-treinado com o *dataset ImageNet*. Durante o treinamento executaram apenas 10 épocas. A Tabela 27 traz a comparação do modelo proposto por Costa et al. (2021) com os modelos médios das duas arquiteturas propostas neste trabalho. O modelo *MobileNetV2* obteve acurácia, precisão, revocação e *F1-score* inferiores em 0,72%. Esse valor é pequeno e pode estar associado à dificuldade do *dataset* ou mesmo a própria arquitetura da *MobileNetV2*, que busca ser mais simples e possuir um custo computacional baixo, mais apropriado para dispositivos móveis ou de baixo poder computacional (COSTA et al., 2021).

Tabela 27 – Comparação dos modelos 1, 2 e Costa et al. (2021).

	Modelo 1	Modelo 2	MobileNetV2
Acurácia total	99,74%	99,74%	99,02%
Precisão média	99,74%	99,74%	99,01%
Revocação média	99,74%	99,74%	99,01%
F1 média	99,74%	99,73%	99,01%

Fonte: Autoria própria (2022)

Siradjuddin, Reynaldi e Muntasa (2021) compararam o desempenho de duas arquiteturas, uma a Faster R-CNN, que emprega dois estágios, e a outra a Fast R-CNN. Também

trabalharam com três classes e utilizaram conjuntos de imagens de tamanhos próximos aos utilizados neste trabalho. A Tabela 28 apresenta a comparação da precisão dos modelos. A diferença é significativa sobretudo para a classe com máscara incorreta. Siradjuddin, Reynaldi e Muntasa (2021) já haviam feito uma crítica em seu próprio trabalho nesse sentido, apontando um desbalanceamento entre as classes no *dataset* que utilizaram para treinamento, apenas 8% das imagens eram da classe com máscara incorreta. Outro ponto é a dificuldade a mais das imagens contidas no *dataset* escolhido por Siradjuddin, Reynaldi e Muntasa (2021), que contava com faces não apenas frontais e também mais do que uma face por imagem. Algumas imagens eram difíceis de serem classificadas até mesmo por humanos, devido a oclusão da máscara por outro objeto.

Tabela 28 – Precisão Média por classe - Siradjuddin, Reynaldi e Muntasa (2021) e arquitetura 1 e 2.

Classe	Faster R-CNN	Fast R-CNN	Modelo 1	Modelo 2
Sem máscara	86%	52%	100%	100%
Com máscara correta	78%	47%	99,42%	99,53%
Com máscara incorreta	57%	17%	99,80%	99,69%
Precisão média	73%	39%	99,65%	99,83%

Fonte: Siradjuddin, Reynaldi e Muntasa (2021) e Autoria Própria

Por fim, Das, Ansari e Basak (2020) também propuseram uma arquitetura simples, porém pré-treinada, voltada para a classificação em duas classes. Eles empregaram a conversão das imagens coloridas para imagens em tons cinzas. Utilizaram ainda dois *datasets* de forma separada. Um com 1.376 imagens, e outro com 853 imagens. Dessa quantidade, 72% foi utilizada para treinamento, 18% para validação e 10% para teste. A arquitetura proposta por eles obteve uma acurácia de 95,77% com o primeiro *dataset* e 94,58% com o segundo. Os valores foram próximos, porém inferiores aos obtidos pelas arquiteturas 1 e 2 neste trabalho. Uma das razões pode ser a pequena quantidade de imagens utilizadas por Das, Ansari e Basak (2020), ou ainda a conversão de cores, em que ocorre perda de informação.

Assim, ao final desse capítulo, foi possível verificar o melhor desempenho das arquiteturas propostas neste trabalho em relação a outros. Dentre as possíveis razões está a utilização de um conjunto de imagens maior combinada com técnicas de aprendizagem profunda eficientes.

6 CONCLUSÃO

A pandemia do novo coronavírus atingiu o mundo, alterando comportamentos e paralisando atividades. Dentre as inúmeras medidas de controle da pandemia, a utilização de máscaras individuais de proteção se destacou como uma medida não farmacológica, barata e de fácil adoção. Mesmo assim, muitas pessoas não a utilizavam corretamente, ou esqueciam de usá-la.

Nesse contexto, o trabalho tratou da classificação de imagens, identificando pessoas sem máscara, com máscara correta e com máscara incorreta. Para isso foram comparadas duas arquiteturas de Redes Neurais Convolucionais (CNN), cada uma gerando cinco modelos. Ao longo da pesquisa, pode-se observar inúmeros trabalhos que trataram do problema e suas sugestões para o uso de um *dataset* com uma maior quantidade de imagens ou a adoção de três classes ao invés de duas. Ambas sugestões foram adotadas neste trabalho.

Esse trabalho buscou empregar um *dataset* grande, com mais de 23 mil imagens por partição, realizando cinco treinamentos e testes com partições aleatórias do *dataset* montado. Teve-se o objetivo de com isso diminuir qualquer viés de uma amostragem única.

Ao final, os modelos propostos obtiveram uma acurácia superior a 99%. Esse resultado é igual ou superior aos trabalhos encontrados na literatura. Os modelos treinados para cada uma das duas arquiteturas apresentaram métricas de desempenho muito próximas em seus grupos, mostrando a consistência da arquitetura ora proposta.

Dessa forma, o trabalho cumpriu o objetivo geral de implementar um modelo de classificação a três classes. Além disso, também foi comparado o desempenho com outros métodos apresentados na literatura. Por fim, outra contribuição desse trabalho foi a construção de um *dataset* com imagens das três classes, totalizando 203.782 imagens. Ele foi composto pela junção dos *datasets* Cabani et al. (2021) e NVIDIA (2019) e poderá ser utilizado em trabalhos futuros. Outra contribuição foi a criteriosa análise do desempenho dos modelos propostos, o que poderá servir de material comparativo para novas propostas. A extensa fundamentação teórica e revisão da literatura também poderão auxiliar pessoas interessadas no estudo das redes neurais convolucionais e no problema da identificação das máscaras.

6.1 Limitações

O *dataset* construído foi gerado de maneira artificial e contou com um único modelo de máscara. Além disso, ele é próprio para situações em que a imagem capturada apresente a face frontal da pessoa, como por exemplo no acesso a um elevador, passagem por uma roleta ou porta, ou ainda início de uma viagem por aplicativo de transporte privado.

6.2 Trabalhos Futuros

Trabalhos futuros incluem: utilização de outros modelos de máscaras nas imagens do conjunto de treinamento, com a finalidade de aumentar a diversidade; coleta de imagens de ambiente aberto (*e.g.* rua), onde as faces não são necessariamente frontais; e construção de modelos para verificar se as máscaras oferecem a proteção adequada.

Referências

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado 3 vezes nas páginas 40, 42 e 50.

ABADI, M. et al. **TensorFlow: A System for Large-Scale Machine Learning**. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 2016. Disponível em: <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>. Acesso em: 30 de abril de 2022. Citado 2 vezes nas páginas 22 e 50.

ABREU, D. dos R.; DELFINO, G. M.; VOLTAN, J. L. N. Graduação em Engenharia da Computação, **Tchau papeleta de faltas**. Rio de Janeiro: [s.n.], 2018. Citado na página 37.

BANERJEE, K. et al. **Exploring Alternatives to Softmax Function**. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2011.11538>. Citado na página 19.

BHANDARY, P. **prajnasb/observations**. 2020. Disponível em: <https://github.com/prajnasb/observations/tree/master/experiements/data>. Acesso em: 5 de abril de 2022. Citado 3 vezes nas páginas 42, 43 e 46.

BRASIL. Lei nº 14.019, de 2 de julho de 2020. **Diário Oficial da União**, Brasília, 2020. Disponível em: <http://https://www.in.gov.br/en/web/dou/-/lei-n-14.019-de-2-de-julho-de-2020-264918074>. Acesso em: 12 de dezembro de 2021. Citado na página 10.

BRASIL. MINISTÉRIO DA SAÚDE. Portaria nº 454, de 20 de março de 2020. **Diário Oficial da União**, Brasília, DF, 2020. ISSN 1677-7042. Disponível em: <https://www.in.gov.br/en/web/dou/-/portaria-n-454-de-20-de-marco-de-2020-249091587>. Acesso em: 12 de dezembro de 2021. Citado na página 10.

CABANI, A. et al. Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19. **Smart Health**, v. 19, p. 100144, 2021. ISSN 2352-6483. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352648320300362>. Citado 4 vezes nas páginas 48, 49, 61 e 71.

CHOLLET, F. et al. **Keras**. GitHub, 2015. Disponível em: <https://github.com/fchollet/keras>. Citado na página 40.

COSTA, V. L. et al. On the performance analysis of a tensorflow based neural network for face mask detection. In: **2021 International Wireless Communications and Mobile Computing (IWCMC)**. [S.l.: s.n.], 2021. p. 342–346. Citado 6 vezes nas páginas 7, 40, 41, 45, 46 e 69.

DAS, A.; ANSARI, M. W.; BASAK, R. Covid-19 face mask detection using tensorflow, keras and opencv. In: **2020 IEEE 17th India Council International Conference (INDICON)**. [S.l.: s.n.], 2020. p. 1–5. Citado 6 vezes nas páginas 42, 43, 44, 45, 46 e 70.

DATAFLAIR. **Download Face Mask Data - DataFlair**. 2022. Disponível em: <https://data-flair.training/blogs/download-face-mask-data/>. Acesso em: 2 de abril de 2022. Citado 3 vezes nas páginas 37, 46 e 67.

DAWSON-HOWE, K. **A Practical Introduction to Computer Vision with OpenCV**. 1. ed. [S.l.]: Wiley, 2014. Citado na página 28.

FACELI, K. et al. **Inteligência artificial : uma abordagem de aprendizado de máquina**. 2. ed. Rio de Janeiro: LTC, 2021. Citado 13 vezes nas páginas 14, 17, 18, 19, 20, 21, 22, 31, 32, 34, 35, 36 e 68.

FILHO, W. de P. P. **Multimídia: Conceitos e aplicações**. 2. ed. Rio de Janeiro: LTC, 2011. Citado 3 vezes nas páginas 29, 30 e 31.

FOUNDATION, P. S. **OS — Diversas interfaces de sistema operacional**. 2022. Disponível em: <https://docs.python.org/pt-br/3/library/os.html>. Acesso em: 30 de abril de 2022. Citado na página 53.

FOUNDATION, P. S. **Random — Generate pseudo-random numbers**. 2022. Disponível em: <https://docs.python.org/3/library/random.html>. Acesso em: 30 de abril de 2022. Citado na página 53.

FREITAS, C. M. de et al. **Covid-19 no Brasil: cenários epidemiológicos e vigilância em saúde**. Rio de Janeiro: Fundação Oswaldo Cruz, 2021. Disponível em: <https://doi.org/10.7476/9786557081211>. Acesso em: 10 de dezembro de 2021. Citado na página 10.

FUKUSHIMA, K. Cognitron: a self-organizing multilayered neural network. 1975. Citado na página 22.

GE, S. et al. **MAFA**. 2018. Disponível em: <https://bit.ly/3FBC52o>. Acesso em: 2 de abril de 2022. Citado 2 vezes nas páginas 39 e 46.

GOLDSCHMIDT, R.; BEZERRA, E.; PASSOS, E. **Data mining : conceitos, técnicas, algoritmos, orientações e aplicações**. 2. ed. Rio de Janeiro: Elsevier, 2015. Citado na página 36.

GéRON, A.; CONTATORI, R. **Mãos à Obra Aprendizado de Máquina com Scikit-Learn e TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes**. 1. ed. Rio de Janeiro: Alta Books, 2019. Citado 13 vezes nas páginas 15, 16, 19, 23, 24, 25, 26, 32, 34, 51, 55, 56 e 57.

HAMMOUDI, K. et al. Validating the correct wearing of protection mask by taking a selfie: Design of a mobile application "checkyourmask" to limit the spread of covid-19. **Computer Modeling in Engineering & Sciences**, v. 124, n. 3, p. 1049–1059, 2020. ISSN 1526-1506. Disponível em: <http://www.techscience.com/CMES/v124n3/39927>. Citado na página 49.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. Citado na página 24.

JUNIOR, A. P.; TEIXEIRA, F.; HOMEM, T. **Aplicação de visão computacional para o monitoramento do uso de máscaras de proteção**. 2020. Disponível em: <http://ocs.ifsp.edu.br/index.php/conict/xiconict/paper/view/7007>. Citado 7 vezes nas páginas 7, 37, 38, 45, 46, 67 e 68.

KERAS. **About Keras**. 2022. Disponível em: <https://keras.io/about/>. Acesso em: 30 de abril de 2022. Citado 3 vezes nas páginas 24, 42 e 51.

KERAS. **Adam**. 2022. Disponível em: <https://keras.io/api/optimizers/adam/>. Acesso em: 30 de abril de 2022. Citado 2 vezes nas páginas 55 e 57.

KERAS. **EarlyStopping**. 2022. Disponível em: <https://keras.io/api/callbacks/early_stopping/>. Acesso em: 30 de abril de 2022. Citado na página 59.

KERAS. **Keras FAQ**. 2022. Disponível em: <https://keras.io/getting_started/faq/>. Acesso em: 30 de abril de 2022. Citado na página 51.

KERAS. **Keras layers API**. 2022. Disponível em: <https://keras.io/api/layers/>. Acesso em: 30 de abril de 2022. Citado 3 vezes nas páginas 27, 56 e 57.

KERAS. **optimizers**. 2022. Disponível em: <https://keras.io/api/optimizers/>. Acesso em: 30 de abril de 2022. Citado na página 57.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural network. **Advances in neural information processing systems**, v. 25, 2012. Citado na página 40.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. p. 436–444, 2015. Citado na página 22.

LECUN, Y. et al. Gradient-based learning applied to document recognition. v. 86, n. 11, p. 2278–2324, 1998. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665>. Citado na página 23.

LENZ, M. L. et al. **Fundamentos de aprendizagem de máquina**. 1. ed. Porto Alegre: SAGAH, 2020. Citado 7 vezes nas páginas 13, 14, 15, 16, 17, 20 e 68.

LUGER, G. F.; VIEIRA, D.; TAVARES, A. I. **Inteligência artificial**. 6. ed. São Paulo: Pearson Education, 2013. Citado 2 vezes nas páginas 13 e 17.

LUNDH, F.; CLARK, A. **About Pillow (PIL Fork) 9.1.0 documentation**. 2022. Disponível em: <https://pillow.readthedocs.io/en/stable/about.html#what-about-pil>. Acesso em: 30 de abril de 2022. Citado na página 53.

MANASWI, N. K. Understanding and working with keras. In: _____. **Deep Learning with Applications Using Python**. Berkeley, CA: Apress, 2018. p. 31–43. ISBN 978-1-4842-3516-4. Disponível em: <https://doi.org/10.1007/978-1-4842-3516-4_2>. Citado na página 51.

MANGALAMPALLI, R. **MAFA data**. 2020. Disponível em: <https://www.kaggle.com/datasets/rahulmangalampalli/mafa-data>. Acesso em: 5 de abril de 2022. Citado 2 vezes nas páginas 41 e 46.

MARANHãO, A. **Face Mask Detection**. 2020. Disponível em: <https://www.kaggle.com/andrewmvd/face-mask-detection>. Acesso em: 5 de abril de 2022. Citado 4 vezes nas páginas 42, 43, 44 e 46.

NEVES, L. A. P.; NETO, H. V.; GONZAGA, A. **Avanços em Visão Computal**. 22. ed. Curitiba: Omnipax, 2012. Citado na página 28.

NOREñA, F. A. S. et al. Evaluación de aiot en modelos computacionales en la nube y en el borde aplicado a la detección de mascarillas. **INGENIUS**, n. 27, p. 32–48, 2022. ISSN 1390-650X. Citado 9 vezes nas páginas 7, 39, 40, 42, 44, 45, 46, 68 e 69.

NVIDIA. **FFHQ dataset**. 2019. Disponível em: <https://github.com/NVlabs/ffhq-dataset>. Acesso em: 08 de janeiro de 2022. Citado 5 vezes nas páginas 48, 49, 53, 61 e 71.

ORGANIZAÇÃO PANAMERICANA DA SAÚDE. **Orientação sobre o uso de máscaras no contexto da COVID-19**: Orientação provisória, 6 de abril de 2020. Brasília - DF, 2020. 5 p. Disponível em: <https://iris.paho.org/handle/10665.2/51994>. Acesso em: 10 de dezembro de 2021. Citado na página 10.

PAGLIARI, C. L. Notas de aula - slide - material de apoio da disciplina Multimídia, curso de engenharia da computação, **Multimídia**. 2018. Citado na página 29.

RUSSELL, S.; NORVIG, P.; MACEDO, R. C. S. de. **Inteligência artificial**. 3. ed. Rio de Janeiro: Elsevier, 2013. Citado na página 13.

SANTOS, R. R. dos et al. **Fundamentos de big data**. Porto Alegre: SAGAH, 2021. Citado na página 29.

SILVERTHORN, D. U. et al. **Fisiologia humana : uma abordagem integrada**. 7. ed. Porto Alegre: artmed, 2017. Citado 3 vezes nas páginas 17, 28 e 29.

SIRADJUDDIN, I. A.; REYNALDI; MUNTASA, A. Faster region-based convolutional neural network for mask face detection. In: **2021 5th International Conference on Informatics and Computational Sciences (ICICoS)**. [S.l.: s.n.], 2021. p. 282–286. Citado 8 vezes nas páginas 7, 41, 42, 44, 45, 46, 69 e 70.

TEAM, T. pandas development. **pandas-dev/pandas: Pandas**. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Citado na página 59.

TENSORFLOW. **API Documentation - TensorFlow Core v2.8.0**. 2022. Disponível em: <https://www.tensorflow.org/api_docs>. Acesso em: 30 de abril de 2022. Citado na página 50.

TENSORFLOW. **Carregar e pré-processar imagens**. 2022. Disponível em: <https://www.tensorflow.org/tutorials/load_data/images>. Acesso em: 02 de maio de 2022. Citado na página 54.

VICH, I. **medical masks dataset images tfrecords**. 2020. Disponível em: <https://bit.ly/3er0tb8>. Acesso em: 2 de abril de 2022. Citado 2 vezes nas páginas 39 e 46.

WANG, Z. et al. **Masked Face Recognition Dataset and Application**. 2020. Disponível em: <https://arxiv.org/pdf/2003.09093.pdf>. Acesso em: 2 de abril de 2022. Citado 3 vezes nas páginas 40, 44 e 46.

APÊNDICE A – APÊNDICE A - MONTAGEM DAS PARTIÇÕES

Tabela 29 – Organização das partições.

	Classe "sem"	Classe "incorreta"	Classe "correta"
Partição 1	[01000, 09000, 15000, 21000, 28000, 29000, 43000, 48000, 59000, 67000]	[00000, 01000, 02000, 05000, 14000, 26000, 28000, 42000, 47000, 60000]	[01000, 03000, 04000, 08000, 53000, 57000, 60000, 62000, 66000, 67000]
Partição 2	[04000, 07000, 37000, 38000, 48000, 54000, 56000, 63000, 65000, 69000]	[08000, 18000, 19000, 29000, 35000, 40000, 42000, 43000, 49000, 55000]	[04000, 15000, 29000, 38000, 41000, 45000, 56000, 58000, 60000, 63000]
Partição 3	[05000, 09000, 14000, 15000, 28000, 38000, 44000, 52000, 55000, 69000]	[01000, 03000, 11000, 18000, 21000, 51000, 52000, 56000, 61000, 68000]	[04000, 06000, 09000, 20000, 30000, 32000, 33000, 34000, 43000, 63000]
Partição 4	[07000, 22000, 23000, 24000, 28000, 30000, 39000, 42000, 47000, 57000]	[03000, 10000, 21000, 27000, 33000, 38000, 41000, 42000, 46000, 59000]	[06000, 22000, 24000, 27000, 36000, 46000, 49000, 51000, 54000, 68000]
Partição 5	[04000, 09000, 12000, 15000, 19000, 24000, 32000, 33000, 46000, 57000]	[03000, 13000, 23000, 26000, 33000, 35000, 42000, 49000, 53000, 63000]	[00000, 17000, 18000, 19000, 21000, 23000, 25000, 32000, 33000, 61000]

Fonte: Autoria própria (2022)

Tabela 30 – Quantidade de imagens por classe em cada partição, separada por treino e teste.

	Classe "sem"	Classe "incorreta"	Classe "correta"	Total
Partição 1	8.000/2.000	7.624/1.902	7.683/1.904	23.307/5.806
Partição 2	8.000/2.000	7.631/1.899	7.682/1.919	23.313/5.818
Partição 3	8.000/2.000	7.614/1.917	7.679/1.904	23.293/5.821
Partição 4	8.000/2.000	7.646/1.884	7.662/1.896	23.308/5.780
Partição 5	8.000/2.000	7.652/1.907	7.690/1.920	23.342/5.827
Total	40.000/10.000	38.167/9.509	38.396/9.543	116.563/29.052
Percentual	34,4%	32,7%	32,9%	100.0/100.0

Fonte: Autoria própria (2022)