Reconhecimento de Pessoas com e sem Máscara Usando Redes Neurais Convolucionais

Lucas Z. Costa, Matheus Fernando S. da Silva, Carlos Andres Ferrero

[1] Instituto Federal de Santa Catarina (IFSC) - Campus Lages
Lages – SC - Brasil
R. Heitor Villa Lobos, 225 - São Francisco, Lages - SC, 88506-400

lucas.z05@aluno.ifsc.edu.br, matheus.fss@aluno.ifsc.edu.br

andres.ferrero@ifsc.edu.br

Abstract. *In 2020, the COVID-19 pandemic was officially declared, which brought several demands in different areas of knowledge. The use of a face mask has become an indispensable item in closed and open spaces. In this context, the development of technologies for automatic identification of face mask use is of fundamental importance. In this work we propose the use of convolutional neural networks to build a computational model that classifies images of faces with and without mask and the development of a mobile application that prototypes this model. The experimental results showed that, among the trained and evaluated models, the Squeezenet model reached 98.04% of accuracy. The developed application was evaluated by members of the academic community indicating that, although there are aspects to be improved, the solution is promising.*

Resumo. *Em 2020 foi oficialmente declarada a pandemia da COVID-19, a qual trouxe diversas demandas em diferentes áreas de conhecimento. O uso de máscara facial tornou-se um item indispensável em espaços fechados e abertos. Nesse contexto, o desenvolvimento de tecnologias para identificação automática do uso de máscara facial é de fundamental importância. Neste trabalho é proposto o uso de redes neurais convolucionais para construir um modelo computacional que classifica imagens de faces com e sem máscara e o desenvolvimento de um aplicativo mobile que prototipa esse modelo. Os resultados experimentais mostraram que, dentre os modelos treinados e avaliados, o modelo Squeezenet alcançou 98,04% de acurácia. O aplicativo desenvolvido foi avaliado por integrantes da comunidade acadêmica indicando que, embora existam aspectos a serem melhorados, a solução é promissora.*

1. Introdução

Com o início da pandemia da COVID-19 em 2020 foi necessária uma mudança nos hábitos de toda a população mundial, no que diz respeito ao distanciamento físico entre pessoas e ao uso de máscaras faciais de proteção, tanto para entrar em espaços fechados, como para caminhar em locais abertos. A Organização Mundial da Saúde (OMS) aconselha, desde 2020, o uso de máscara como uma das medidas de prevenção e controle de propagação do vírus que causa a COVID-19. No Brasil, o uso de máscaras em espaços

públicos e privados tornou-se obrigatório com a Rec. N°072 de 21 de dezembro de 2020 (Brasil, 2020).

Contudo a restrição para o uso de máscaras para entrar em locais se tornou algo custoso, no qual o procedimento que envolve um funcionário para fazer esta verificação torna o processo vulnerável a erros por conta do fator humano e outras dificuldades em função de fatores como falta de atenção, fadiga, estresse, e imprevistos do local de trabalho.

A utilização de Inteligência Artificial e Visão Computacional são áreas de estudo que podem auxiliar no processo de identificação automática de pessoas sem máscaras de proteção por meio de algoritmos que analisam os *pixels* da imagem e são capazes de identificar essas características. Desde o início da pandemia diversos modelos de redes neurais existentes foram avaliados para esse fim como o desenvolvido por Redmon e Farhadi (2018) e Howard et al. (2019). Esses algoritmos podem ser usados em sistemas computacionais de captura e análise de imagens em barreiras sanitárias para identificação de pessoas sem máscara, emissão de alertas e relatórios.

Este trabalho tem o objetivo de desenvolver uma solução computacional de automatização da identificação de pessoas com e sem máscara por meio de técnicas de Visão Computacional. São objetivos específicos desse trabalho:

- Desenvolver e avaliar modelos de classificação de imagens para identificar pessoas com e sem máscara;
- Desenvolver um aplicativo para dispositivos móveis multi plataforma para auxiliar no reconhecimento de pessoas com e sem máscara;
- Integrar o modelo de identificação de pessoas com e sem máscara ao aplicativo.

O processo de fiscalização é realizado através de um *Smartphone* que usa uma câmera para verificar se a face de uma pessoa está utilizando máscara ou não, auxiliando a garantir o atendimento às normas sanitárias nos ambientes de convívio social ou profissional. Embora, a idealização do presente trabalho seja em função de pandemia da COVID-19, os resultados deste trabalho são úteis para quaisquer outros domínios, fora do contexto da pandemia, que tenham como requisito o uso de máscaras para evitar a exposição de pessoas a agentes físicos, químicos ou biológicos prejudiciais à saúde.

Em relação à metodologia, este trabalho foi desenvolvido em cinco etapas macro. A primeira etapa consiste em pesquisas realizadas através do *Google Scholar*, onde foi definido uma expressão de busca de artigos científicos que se assemelham com o tema tratado neste trabalho. A segunda etapa consiste em entender todo o processo de identificação de máscara, desde a forma de se posicionar em frente ao dispositivo até retornar uma resposta do mesmo, procurando um método mais eficiente de automatização. A terceira etapa é realizada com auxílio de um dispositivo móvel, que serviu tanto como entrada, que captura uma imagem do rosto de uma pessoa como também para saída de dados, informando na tela do dispositivo o resultado do reconhecimento. A quarta etapa foi desenvolvida todo o esquema de teste e avaliação de todo o processo de identificação e reconhecimento facial. A etapa cinco consiste na publicação do artigo científico.

Sob o ponto de vista da natureza, este trabalho se classifica como pesquisa aplicada, pois envolve aplicação prática e construção de um projeto de aplicação de reconhecimento facial. Sob o ponto de vista da forma de abordagem do problema, trata-se de uma

pesquisa quantitativa, pois utiliza-se de métodos estatísticos para geração de relatórios. Sob o ponto de vista de seus objetivos, é uma pesquisa descritiva, pois ela envolve uma etapa de coleta de dados. Sob o ponto de vista dos procedimentos técnicos é uma pesquisa bibliográfica, pois se utiliza de materiais, artigos encontrados através do site *Google Scholar* como base.

As próximas seções do documento estão dispostas da seguinte forma. Na Seção 2 é apresentado o referencial teórico, necessário para o entendimento do trabalho. Na Seção 3 é detalhado o desenvolvimento do projeto, na Seção 4 são apresentados e discutidos os resultados do desenvolvimento, na Seção 5 são apresentadas as conclusões e os trabalhos futuros.

2. Referencial Teórico

Nesta seção foram abordados estudos relacionados que tiveram importância para o desenvolvimento do trabalho. Foi feito tambem uma revisao sistematica, em que foram agrupados os artigos, e que fossem filtrados por ordem de relevância. Para esta revisão foi utilizado o *Google Scholar* para a procura desses artigos.

2.1. COVID-19

Inicialmente dada como uma epidemia na China, a COVID-19 em seus meses iniciais já somava mais de 2 milhões de casos e 120 mil mortes e logo se tornou uma pandemia com um alto nível de risco. Assim como outras crises sanitárias, a COVID-19 também contou com algumas medidas para evitar a contaminação, como o distanciamento social, uso de máscara entre outros métodos (Werneck e Carvalho, 2020).

Por tratar-se de uma pandemia, um dos principais objetivos atuais é a gerência de fatores que evitam o contágio e a proliferação do vírus. Sendo assim, alguns hábitos higiênicos foram adotados, como o distanciamento social, que evita aglomerações, o uso da máscara facial, que reduz a transmissão do vírus, higiene das mãos e roupas além de outros métodos para combater o vírus (Farias, 2020).

Como mencionado, o uso de máscara de proteção facial é um fator fundamental, junto a outras formas de proteção, para conter a pandemia da COVID-19. E a utilização de métodos computacionais para identificar pessoas com e sem máscara ao entrar em estabelecimento e/ou espaços fechados, pode auxiliar nesse sentido. Essa identificação, a partir da fotografia da face de um indivíduo, pode ser realizada por meio de um processo chamado de classificação de imagens.

2.2. Classificação de Imagens utilizando Aprendizado de Maquina

Aprendizado de Máquina é uma subárea da Inteligência Artificial, que tem como principal objetivo desenvolver técnicas computacionais capazes de adquirir conhecimento automaticamente (Han et al., 2011). Esse conhecimento pode ser representado por meio de funções matemáticas, chamadas de modelo, que podem auxiliar em processos de tomada de decisão. Uma das principais formas de aprendizado é o aprendizado supervisionado, em que um modelo é induzido a partir de um conjunto de dados rotulado, onde cada instância do problema possui o objetivo preditivo anotado por um especialista ou por algum processo, em que o algoritmo de indução de modelos pode confiar para aprender. Caso o objetivo preditivo seja um valor numérico a tarefa de aprendizado é chamada de

regressão e caso seja uma categoria a tarefa é denominada de classificação (Tan et al., 2005). Independentemente da tarefa, o resultado desse aprendizado é uma função matemática (modelo) que pode ser utilizada para predizer o rótulo de novos dados.

Os modelos de classificação de imagens devem ser construídos a partir de um conjunto de imagens previamente rotuladas, denominado conjunto de treinamento. Como exemplo, no contexto deste trabalho, um conjunto de imagens consiste em uma coleção de fotografias de faces de pessoas e o rótulo consiste na anotação que indica a informação verdadeira do uso ou não de máscara. Assim, um conjunto de imagens de treinamento é definido como um conjunto de pares $D_{train} = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$, onde o i-ésimo par contém a imagem X_i e a anotação y_i o rótulo ou classe dessa imagem (por exemplo, com ou sem máscara). Cada imagem X do conjunto, de altura h e largura w é representada por uma matriz de h linhas e w colunas, onde cada elemento da matriz corresponde a um ponto ou pixel da imagem e possui três valores numéricos entre 0 e 255, referentes às componentes R, G e B, do sistema de cores RGB (*red*, *green*, *blue*) (Gonzalez e Woods, 2009). Nesse sentido, um classificador é uma função matemática $f(X) \rightarrow y$ que mapeia os dados da imagem (*pixels*) para uma das classes predefinidas do problema.

Realizar o treinamento de um classificador requer a utilização de um algoritmo de aprendizado. Atualmente, existem diversos métodos para classificação de dados em geral, como Árvores de Decisão, Máquinas Vetores de Suporte, e Redes Neurais. Especificamente para construir construir modelos de classificação de imagens, um tipo de redes neurais tem tido destaque nos últimos anos, as Redes Neurais Convolucionais. A seguir são apresentados conceitos básicos de Redes Neurais e de Redes Neurais Convolucionais.

2.3. Redes Neurais Artificiais

As Redes Neurais são funções matemáticas inspiradas em sistemas biológicos. Uma rede neural Artificial pode ser descrita como um mapeamento de um espaço de entrada a um espaço de saída. O propósito de uma rede neural é mapear a entrada para uma saída desejada. Na Figura 1 são apresentados a representação biológica de um neurônio (à esquerda) e a representação computacional do neurônio (à direita), onde um conjunto de valores de entrada (que representam os dendritos) são combinados linearmente com pesos, que descrevem a importância de cada valor de entrada no problema, e uma função de ativação $f(.)$ é utilizada para decidir se envia ou não um sinal para um próximo neurônio.

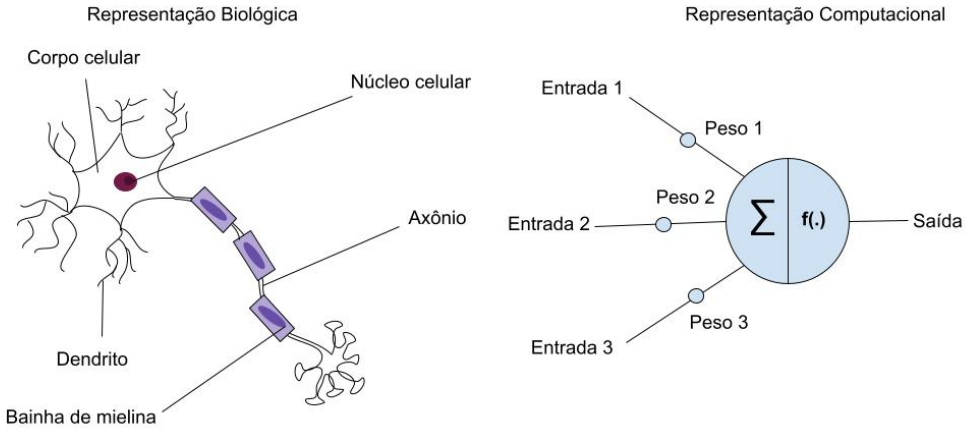

Figura 1. Representação de um neurônio biológico e um computacional

A coleção de neurônios organizados na forma de camadas constituem as chamadas Redes Neurais, como apresentado na Figura 2. Os neurônios de uma camada enviam sinais como entrada para neurônios das camadas seguintes. Redes Neurais devem ter uma camada de entrada (com número de nerônios de acordo com o número de dados de entrada), uma ou mais camadas intermediárias, de acordo com a complexidade do problema, e uma camada de saída, que em problemas de classificação possui a quantidade de neurônios de acordo com o número de classes do problema. No contexto do problema deste artigo, as entradas da primeira camada da rede neural são os *pixels* da imagem, e a comunicação entre neurônios ocorre somente da esquerda (camada de entrada) para a direita (camada de saída).

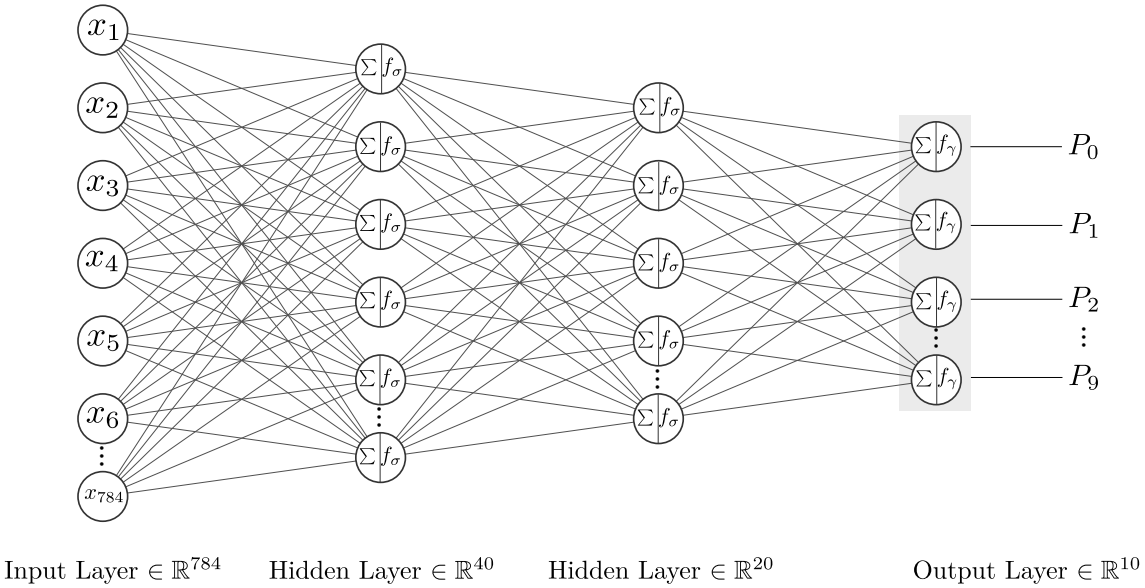

Figura 2. Exemplo de uma rede neural, adaptada de (Ferrero, 2021)

Na Figura 2 é apresentado um exemplo de rede neural para classificação de dez classes. O modelo contém: uma camada de entrada (*input layer*) de 784 dados, onde cada dado corresponde a um *pixel* de uma imagem de dimensões $h = 28$ e $w = 28$, i.e., $28 \times 28 = 784$; duas camadas intermediárias (*hidden layer*) uma com 40 neurônios e outra

com 20 neurônios; e uma camada de saída (*output layer*) que contém uma camada com dez neurônios, um para cada classe do problema e a saída corresponde à probabilidade de uma imagem ser de cada uma das classes. A rede neural apresentada faz parte de um tipo de rede neural denominada de Rede Neural Perceptron Multicamadas (*Multilayer Perceptron* – MLP) e tem como principal característica que suas camadas são completamente conectadas (*Fully Connected Layers*), i.e., todas as saídas de uma camada se conectam a todas as entradas da próxima camada. Essa tipo de camadas que faz com que o número de parâmetros ou arestas (os pesos mencionados na Figura 1) cresça rapidamente com a complexidade do problema e consequentemente seja mais difícil e demorado realizar o treinamento do modelo.

2.3.1. Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNN) ganharam relevância nos últimos tempos em aplicações de aprendizado de máquina envolvendo o reconhecimento visual de objetos (LeCun et al., 1989). A quantidade de parâmetros que uma Rede Neural MLP utiliza pode ser reduzida utilizando as Redes Neurais Convolucionais. Diferente das MLP, as CNNs exploram padrões locais, o que faz sentido em análise de imagens, já que cada *pixel* de imagem tem mais chances de ter relação com os *pixels* em sua vizinhança do que com *pixels* mais distantes dentro da imagem.

As CNNs para análise de imagens possuem duas principais partes em sua arquitetura, como mostra a Figura 3: extração de características (*feature extraction*) e classificação. A extração de características tem como principal objetivo transformar, ou representar, a imagem por um conjunto de valores numéricos que representam as características mais importantes da imagem para o problema de classificação em questão. E a parte de classificação corresponde basicamente a uma rede neural MLP, como a apresentada na seção anterior.

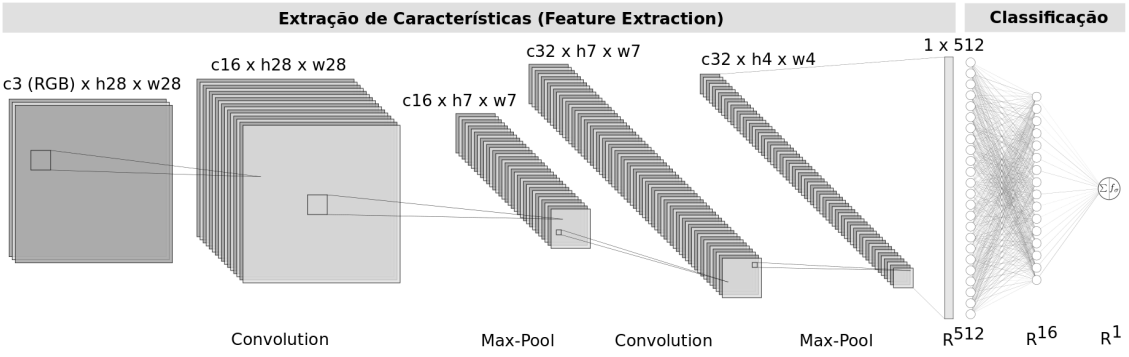

Figura 3. Exemplo de uma rede neural convolucional, adaptada de (Ferrero, 2021)

A Figura 3 apresenta um exemplo de rede neural convolucional que possui como entrada uma imagem de dimensões $h = 28$ e $w = 28$ e três componentes $c = 3$ do sistema RGB. A primeira camada, de convolução, permite criar 16 novos componentes. Para isso são usados 16 filtros, que consistem em matrizes de pesos (por ex. de 3×3) que deslizam por toda a imagem aplicando uma combinação linear dos *pixels* com os pesos da matriz resultando em uma nova componente (Albawi et al., 2017). Esses filtros são

semelhantes aos filtros usados em técnicas convencionais de processamento de imagens, como filtros Passa-alta e Sobel, para detecção de bordas (Gonzalez e Woods, 2009), com a diferença que nos filtros das camadas convolucionais os valores dos coeficientes não são predefinidos, eles são ajustados durante o processo de treinamento da rede neural.

Após a aplicação da convolução é utilizada uma camada para reduzir o tamanho de cada componente, chamada de *Max-pool*. Essa camada transforma cada componente de dimensões $h = 28$ e $w = 28$ em outra componente de menor tamanho, por ex. quatro vezes menor, com dimensões $h' = 7$ e $w' = 7$, procurando não diminuir a qualidade de informação contida em cada componente. Após isso, um novo par de camadas (convolucional e Max-pool) é utilizado para aplicar 32 novos filtros e reduzir as 32 componentes para dimensões $h'' = 4$ e $w'' = 4$. Após isso, os valores contidos nessas componentes são rearranjados em uma única camada contendo todos os 512 ($= 32 \times 4 \times 4$) valores. Esse processo de transformar uma imagem de dimensões $3 \times 28 \times 8$, em soma 2352 valores, para somente 512 valores é denominado de redução de dimensionalidade. Por fim, esses valores são a entrada para uma rede neural MLP na parte de classificação.

Desde 2012 as CNNs ganharam a atenção tanto de pesquisadores acadêmicos quanto de centros de pesquisa na indústria, em função de sua aplicabilidade para a resolução de problemas do mundo real. Diferente do modelo apresentado na Figura 3, as CNNs aplicadas permitem imagens de entrada de tamanho maior (comumente de dimensões $h = 224$ e $w = 224$) e um grande número de camadas convolucionais, *Max-pool*, e de outros recursos. Alguns exemplos de arquiteturas propostas e amplamente utilizadas em aplicações acadêmicas e comerciais são VGG (Krizhevsky et al., 2012), Alexnet (Simonyan e Zisserman, 2014), ResNet (He et al., 2016), Squeezenet (Iandola et al., 2016) e Mobilenet (Howard et al., 2019). No entanto, o treinamento dessas redes é custoso, tanto em tempo de processamento quanto em recursos de memória. Por esse fato, existem bibliotecas computacionais que permitem carregar essas redes pré-treinadas para problemas de classificação complexos (Chollet et al., 2015; Keras, 2021), geralmente envolvendo um grande número de classes, e realizar o treinamento de apenas parte da rede neural, usualmente apenas das últimas camadas do classificador. Dessa forma, os pesos das camadas referentes à extração de características ficam intactas. Esse processo é denominado de ajuste fino (*fine tuning*) sendo o processo utilizado para o desenvolvimento deste trabalho.

2.4. Trabalhos Relacionados

A revisão da literatura foi realizada por meio de uma revisão sistemática, utilizando a plataforma *Google Scholar* para a procura de artigos e materiais sobre o tema. A *string* de pesquisa utilizada foi *"Neural Networks Facemask Recognition"*. A maior parte dos resultados corresponde a trabalhos realizados desde 2020, pois é um assunto que tem recebido maior atenção pela pandemia da COVID-19. Foi construída uma planilha para organizar os trabalhos de acordo com a ordem de relevância. De forma a ordenar os trabalhos, foi utilizada uma escala de 1 a 5, onde 1 representa um trabalho pouco relevante e 5 representa um trabalho muito relevante. A partir disso, foram selecionados apenas os trabalhos com relevância 3 ou maior para leitura completa, são apresentados a seguir.

Adusumalli et al. (2021) desenvolveram um modelo de classificação pessoas com e sem máscara baseado na arquitetura *MobileNetV2* da biblioteca *Keras* de (Chollet et al.,

2015). Esse modelo foi gerado e treinado utilizando 75% de um conjunto de imagens rotuladas e 25% para testar e avaliar a qualidade do modelo. O funcionamento se dá por uma entrada de uma imagem que é enviada para o módulo de reconhecimento facial. Após o redimensionamento da imagem ela é enviada para o detector facial, na qual é recortado o rosto sem o fundo da imagem e como saída é apresentada se a pessoa na imagem está ou não utilizando máscara.

Em um trabalho desenvolvido por Militante e Dionisio (2020a), para o reconhecimento de máscara facial, foi utilizado um conjunto de dados contendo 25.000 (vinte e cinco mil) imagens com dimensões entre 800 e 1.200 de altura e largura, as quais são transformadas para o tamanho de 224×224 e divididas em imagens de pessoas com máscara facial e sem a máscara facial.

Um trabalho feito por Sharma (2020) tem início na coleta de imagens rotuladas como se estivessem ou não de máscara. Para que o conjunto possa ser treinado, logo após é utilizada a biblioteca *TensorFlow* para o processamento de imagens criando um modelo *deep learning* em *Python*, logo após é utilizado o YOLO para a detecção de objetos em tempo real, no caso a máscara. O próximo passo é treinar o modelo, após isso é feito o teste baseado no *dataset* utilizando o *OpenCV* para carregar as imagens para o processo de teste, após isso o modelo classifica com precisão as pessoas que usam ou não máscara.

Segundo Militante e Dionisio (2020b), o processo de reconhecimento de máscara facial, se inicia na coleta da imagem para a identificação, o que resulta em um *dataset* com imagens de pessoas com e sem máscara. O processo de segmentação divide o conjunto de imagem em segmentos e são utilizados na extração de áreas cobertas pela máscara facial no rosto da pessoa. Após isso, os segmentos de imagem são utilizados para treinar modelos de redes neurais convolucionais para posterior classificação.

Em 2020 foi criado uma rede neural por Inamdar e Mehendale (2020) chamada *Facemasknet*, utilizada para reconhecimento de máscaras através de vídeos *streams*, a estrutura do *Facemasknet* em relações a outras redes neurais é muito menos complexa e fornece resultados instantâneos, segundo eles, foram os primeiros há utilizarem três classes para classificações: com máscara, sem máscara e utilização incorreta, com a acurácia de 98,7%.

Foi proposto em 2021 um detector de máscaras feito pela Nithya Sankari et al. (2021) que recebe como entrada vídeos, nos quais é possível detectar o uso das mesmas. O modelo proposto foi treinado utilizando um *dataset* de 1395 imagens com pessoas com/sem máscara. A partir dele foi alterado ângulos e filtros tendo como resultado um *dataset* de 2.751 imagens.

Foi desenvolvido em 2021 por Qin e Li (2020) uma nova rede neural para identificação combinando imagens de alta resolução em conjunto com classificações neurais, na qual quantifica um problema de classificação *three-category* (utilização correta da máscara, sem utilização da máscara e mal uso da máscara) baseada em uma irrestrita imagem facial 2D, conseguindo uma acurácia de 98,7%, passando a performance da tradicional classificação de imagens *end-to-end*, com os resultados indicando que o *SRCnet* pode alcançar uma alta acurácia em identificação de máscaras.

Em 2020, por Oumina et al. (2020) foi usado diferentes tipos de arquiteturas, como *MobileNetV2*, *VGG19* e *Xception* para extrair detalhes de imagens de rostos, e

também foram utilizados classificadores de *Machine Learning* como *Support Vector Machine* (SVM) e *K-Nearest Neighbours* (kNN) na parte de classificação da arquitetura, ao invés de utilizar uma rede neural MLP. A melhor acurácia alcançada foi de 95,1% combinando *MobileNet* com SVM o segundo melhor foi de 91,3% com VGG19 e kNN.

Fasfous et al. (2021) apresentaram uma rede neural binária com baixo consumo de energia, que consegue classificar imagens de pessoas com o uso correto das máscaras. A tarefa de classificação foi implementada num acelerador FPGA embutido, realizando operações binárias de alta performance, consumindo 2W de energia utilizando diversas câmeras e 1,6W de energia com apenas uma câmera, alcançando uma acurácia de 98%.

Nagrath et al. (2021) propuseram um método chamado SSDMNV2 com o uso de *Deep Learning*, *Tensorflow*, *Keras* e *OpenCV* para detectar rostos com máscaras, no qual afirmam que esse método criado é seguro pois utiliza poucos recursos para ser eficiente. O SSDMNV2 utiliza como classificador *Single Shot Multibox Detector* e MobileNetV2, que garante que é bastante leve para realizar esta atividade e ainda é compatível com os dispositivos da NVIDIA.

Lin et al. (2021) introduziu uma nova Rede Neural chamada *OpenposeCNN* de *new realtime* para automatização de reconhecimento de rostos com máscara com a combinação de reconhecimento de posturas. O processo foi feito treinando o modelo para identificar se a imagem capturada do sujeito está com máscara, reduzindo a área para ser processada pelo *framework* e também adotaram aprendizado supervisionado para detectar se a máscara está presente, com acurácia de 95,8% para fotos de pessoas com máscara de dia e 94,6% de noite.

Ayyappa et al. (2021) propôs um sistema automatizado de detector de rostos com máscara com a utilização de *Back Propagation Neural Network* (BPNN) em conjunto de *Median Filtering*, obtendo 91% de acurácia e afirma que é bastante apta para classificar pessoas utilizando máscara ou não e ainda confirma que a acurácia poderá aumentar utilizando *datasets* maiores.

Loey et al. (2021) apresentaram e utilizaram em seu projeto a técnica de *Hybrid Transfer Learning* consistindo de duas partes. A primeira parte foi a extração de características utilizando uma arquitetura ResNet50 He et al. (2016), enquanto a segunda parte foi para classificar o uso de pessoas com máscara utilizando algoritmos clássicos de *Machine Learning*, tais como SVM, *Decision Tree*, chegando com a acurácia máxima e pouco processamento com de 99,6%.

Na Tabela 1 são apresentados os algoritmos e a acurácia em cada modelo utilizado nos trabalhos mencionados anteriormente, dado que em alguns dos trabalhos há dados faltantes e estes estão marcados com um "-". De acordo com a tabela, a primeira coluna trata do autor do trabalho, a segunda coluna do tamanho do conjunto de dados utilizado, a terceira a divisão do modelo de treino e teste, a quarta a rede neural utilizada e por fim na quinta coluna esta a acurácia obtida no trabalho.

Tabela 1. Comparação dos trabalhos relacionados.

Artigo	Tamanho do Dataset	Treino/Teste	Modelo de Rede Neural	Acurácia
Qin e Li (2020)	70.534	90/10	SRCNet	98,7%
Militante e Dionisio (2020b)	25.000	80/20	VGG-16	96%
Oumina et al. (2020)	1.376	80/20	MobileNetV2	95,1%
Inamdar e Mehendale (2020)	3835	70/30	Facemasknet	98,7%
Sharma (2020)	-	-	YoloV5	97,1%
Nagrath et al. (2021)	5.521	80/20	MobileNetV2	92%
Adusumalli et al. (2021)	3.835	75/25	YoloV4	98%
Lin et al. (2021)	2.868	42/58	OpenposeCNN	95,8%/94,6%
Loey et al. (2021)	600.000	80/20	YoloV3 (variação)	99,4%
Ayyappa et al. (2021)	92.671	-	BPNN	91%
Fasfous et al. (2021)	133.783	82/18	BinaryCoP	98%

De acordo com a Tabela 1 diversos modelos de redes neurais foram utilizados e entre os mais utilizados nos trabalhos relacionados estão o modelo *MobileNetV2* (Sandler et al., 2018) e o SRCNet (Qin e Li, 2020).

Os três modelos que apresentaram melhor acurácia nos trabalhos relacionados foram uma variação de YoloV3 (99,4%) proposta por (Loey et al., 2021), o modelo Facemasknet (98,7%) proposto por (Inamdar e Mehendale, 2020), e o modelo SRCNet (98,7%) proposto por (Qin e Li, 2020). É importante ressaltar que os modelos de variação de YoloV3 e SRCNet foram treinados e avaliados com os maiores tamanhos de *dataset*, 600 mil e 70 mil imagens, respectivamente, o que auxilia na precisão dos modelos.

3. Desenvolvimento

Esta seção aborda o desenvolvimento do trabalho e está apresentado em 5 etapas: seleção de dados, pré-processamento de imagens, indução de modelos, avaliação de modelos e desenvolvimento da aplicação mobile. Essas etapas são descritas a seguir.

3.1. Seleção dos Dados de Imagem

O conjunto de dados utilizado neste trabalho está constituído de 19.544 imagens, das quais 6.832 imagens correspondem a faces de pessoas com máscara e 12.712 imagens correspondem a faces de pessoas sem máscara. Essas imagens foram coletadas a partir de datasets disponíveis publicamente. Esses datasets são *Real-World-Masked-Face-Dataset* de X-zhangyang (2021) contendo 92.672 imagens mas que não foram todas utilizadas, coletando somente 3.009 imagens. *Face Mask Detection* de Kumar (2021) contendo 7.553 imagens, o terceiro deles é o *dataset Face Mask Detection Dataset* de 8.982 imagens Gurav (2020).

3.2. Pré-processamento de Imagens

Na etapa de pré-processamento as imagens são processadas para que estejam no formato adequado de entrada das redes neurais e para auxiliar na criação de modelos de classificação mais robustos.

Para as imagens utilizadas para o treinamento de modelos, primeiramente é aplicado o redimensionamento, que transforma o tamanho da imagem original para 224 x 224 pixels, que é o tamanho de imagem adequado para que possa ser utilizada pelas redes neurais utilizadas neste trabalho. Além disso, às imagens de treinamento são aplicados: o espelhamento horizontal, onde cada imagem tem uma probabilidade de 50% de ser espelhada horizontalmente ou não; e alteração de cor, onde o brilho, o contrataste, a saturação e a tonalidade, podem variar com fator 0,1, indicando que os valores desses atributos da imagem podem variar de 0,9 a 1,1 do valor original. Esses pré-processamentos são importantes durante o treinamento para que a rede neural aprenda a classificar as imagens originais mas também variações delas, aumentando a sua robustez.

Para as imagens utilizadas durante a avaliação do modelo, seja para validação ou teste, é aplicado apenas o redimensionamento da imagem para 224 x 224 pixels.

3.3. Indução de Modelos

A partir do levantamento bibliográfico utilizado neste trabalho e a consulta às redes neurais pré-treinadas disponíveis na biblioteca *Torchvision* de PyTorch (Keras, 2021), foram selecionados 6 modelos para realizar este trabalho: Alexnet, VGG16, Squeezenet, Mobilenet V2, Mobilenet V3 Large e Small. Esses modelos citados foram utilizados para realizar transferência de aprendizado (*Transfer Learning*), que permite que esses modelos que tiveram um ótimo desempenho para classificar conjuntos de imagens com um grande número de classes sejam utilizados para outros problemas de classificação, como o tratado neste trabalho, apenas realizando o ajuste de um subconjunto de parâmetros do modelo (*fine-tunning*), já que treinar o modelo por inteiro pode ser muito custoso e resultar menos robusto. Com isso, o *fine-tunning* do modelo foi realizado a partir da versão pré-treinada desses modelos.

3.4. Avaliação de Modelos

Os modelos foram avaliados utilizando uma estratégia *holdout*. Do total de 19.544 imagens do conjunto foram selecionadas aleatoriamente 5.000 imagens para treinamento, 5.000 imagens para validação (avaliação durante o treinamento do modelo) e o restante, 9.544 imagens, para teste. As imagens de conjunto de treinamento são utilizadas para ajustar os pesos da rede, as do conjunto de validação são utilizadas para monitorar o erro da rede neural e evitar *overfitting*, e as de teste são utilizadas para calcular as medidas de avaliação do modelo. Todos os modelos foram treinados por 10 épocas utilizando a plataforma *Google Colab* com suporte a GPU.

As medidas de avaliação utilizadas para comparar os modelos foram: acurácia, *precision*, *recall*, F1-*score*. Na Tabela 2 são apresentados os resultados alcançados por esses modelos no conjunto de imagens teste, bem como o tamanho, em memória, de cada modelo treinado.

Tabela 2. Comparação de redes neurais.

Modelo de Rede Neural	Acurácia	Precisão	Recall	F1 Score	Memória (MB)
Alexnet	97,63%	0,9794	0,9845	0,9820	226,43
VGG16	98,76%	0,9977	0,9834	0,9905	731,54
Squeezenet	98,04%	0,9911	0,9789	0,9850	92,60
Mobilenet V2	96,87%	0,9703	0,9823	0,9763	161,93
Mobilenet V3 Large	96,43%	0,9902	0,9549	0,9723	126,35
Mobilenet V3 Small	95,75%	0,9808	0,9538	0,9671	42,04

A rede neural que apresentou a melhor acurácia foi a VGG16, pois possui uma acurácia de 98,76%, já o seu tamanho em memória é de 731,53 Megabytes (MB). O segundo melhor resultado foi a rede neural Squeezenet com 98,04% de acurácia com o tamanho de 92,60 MB e a terceira melhor foi a Alexnet com 97,63% de acurácia com tamanho de 226,43 MB. A rede neural Squeezenet apresentou um melhor relação custo benefício, pois ela possui um desempenho próximo ao da VGG16, sendo que ocupa menos espaço em memória, o que permite que seja carregada mais rapidamente em um serviço na nuvem e que o tempo de resposta seja menor, que são características desejáveis na hora de implementar o modelo em produção, por meio de uma API, e ser utilizado em aplicações *web* ou *mobile*. Com isso, o modelo de rede neural Squeezenet treinado foi escolhido para a desenvolver um serviço em nuvem que permite classificar imagens e que a aplicação Mobile desenvolvida neste trabalho utilizou.

3.5. Desenvolvimento da Aplicação Mobile

O desenvolvimento do aplicativo que utiliza o modelo construído na etapa anterior consistiu nos seguintes passos: levantamento de requisitos, projeto da solução, implementação e avaliação. Começando pelo levantamento de requisitos, que enumera os requisitos funcionais, e menciona os requisitos não funcionais, o próximo passo sobre o projeto da solução que diz respeito a base da construção do aplicativo com diagrama de casos de uso, protótipo visual construído para o aplicativo ser baseado e o diagrama de atividades, após esse passo a implementação que contém parte do desenvolvimento prático do aplicativo, e por fim é descrita a avaliação que diz como o software foi avaliado para servir de base de qualidade.

3.5.1. Levantamento de Requisitos

Com base em estudos feitos foram definidos as prioridades dos requisitos. Os valores das prioridades foram baseadas em seu nível de importância dentro do sistema, sendo o valor 1 de menor prioridade e valor 5 com maior prioridade.

Requisitos funcionais são declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas, e como o sistema deve se comportar em situações particulares, em casos particulares, o requisito funcional também pode declarar explicitamente o que o sistema não deve fazer (Sommerville, 2011). Na Tabela 3 são apresentados os requisitos funcionais da aplicação que interage com o usuário (*frontend*).

Na Tabela 4 são apresentados os requisitos funcionais da API (*backend*), que realiza a análise das imagens e a predição de pessoas com e sem máscara. Essa API é utilizada pela aplicação mobile (*frontend*).

Tabela 3. Requisitos funcionais da interação do usuário com o aplicativo.

Requisitos Funcionais		
Id	**Descrição**	**Prioridade**
RF. A001	Todas as salas serão carregadas pelo sistema de acordo com o banco de dados.	5
RF. A002	O usuário pode selecionar uma das salas.	3
RF. A003	O usuário pode fotografar-se.	4
RF. A004	A foto será processada pelo modelo para amostragem do resultado.	5
RF. A005	Após ser mostrado o resultado no aplicativo, será registrado no banco de dados.	5

Tabela 4. Requisitos funcionais de interações entre o *Backend*, *Frontend* e o banco de dados.

Requisitos Funcionais		
Id	**Descrição**	**Prioridade**
RF. B001	O *frontend* faz uma requisição através de um método *JavaScript Object Notation* (JSON) para o *backend* passando como parâmetro a imagem (Base64).	5
RF. B002	O *backend* faz a decodificação da imagem (Base64).	5
RF. B003	A imagem decodificada passa pelo modelo e gerando assim os resultados da imagem.	5
RF. B004	Como resposta o *backend* retorna os resultados da imagem em conjunto com outros dados.	5
RF. B005	Com esses dados recebidos o *frontend* apresenta no aplicativo as informações.	5
RF. B006	Após receber as informações do *backend*, esses dados são reenviados para o banco de dados pelo *frontend*.	5

Requisitos não funcionais são restrições nos serviços ou funções oferecidas pelo sistema. Eles incluem restrições de tempo, restrições no processo de implementação e restrições impostas pelos padrões. Requisitos não funcionais muitas vezes se aplicam ao sistema como um todo, ao invés de sistema individual de recursos ou serviços (Sommerville, 2011). Por exemplo, o tempo de processamento da imagem que o *backend* faz e retorna para o *frontend* é rapido e ainda gera bons resultados.

O desenvolvimento deste protótipo tem como principal objetivo utilizar o modelo de reconhecimento de uso de máscara desenvolvido, por meio de uma aplicação mobile. A utilização deste aplicativo foi da seguinte maneira: um funcionário da instituição (usuário) realiza o posicionamento do dispositivo móvel (tablet ou celular) na entrada de um espaço físico e escolhe qual o nome da sala; cada usuário ao entrar na sala deverá se posicionar em frente à câmera frontal do dispositivo móvel e pressionar o botão de captura de fotografia; o usuário deverá aguardar o resultado e, se for identificado com máscara, será permitida a entrada na sala. É importante ressaltar que nenhuma imagem capturada pelo aplicativo é armazenada em banco de dados. No entanto, são realizados registros do resultado das identificações de cada sala, para posterior produção de relatórios.

3.5.2. Projeto da Solução

A *Unified Modeling Language* (UML) é uma família de notações gráficas, apoiado por um único meta-modelo, que ajuda a descrever e projetar software sistema (Fowler, 2018). Neste trabalho foram usados o diagramas de Caso de Uso e Diagramas de Atividades.

Na Figura 4 é apresentado o protótipo desenvolvido utilizando a ferramenta Figma[1].

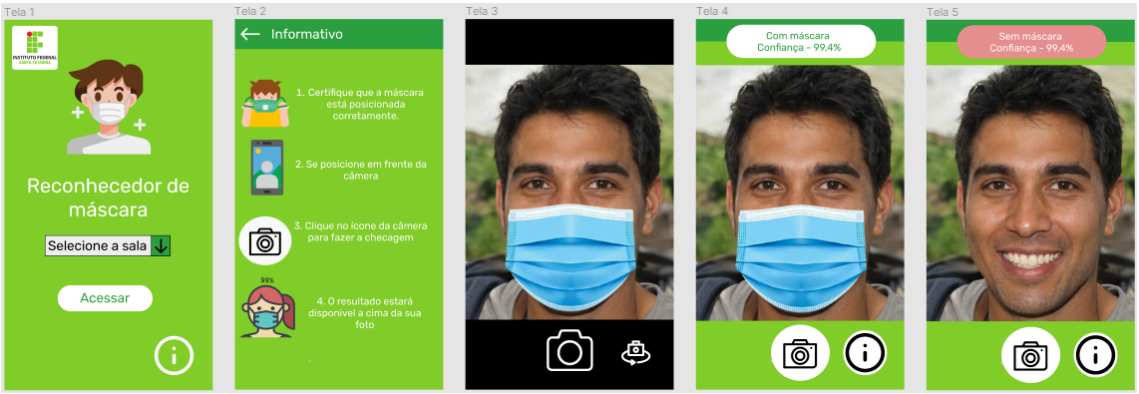

Figura 4. Representação do protótipo visual do aplicativo.

Por meio desse protótipo é possível descrever o conjunto de telas que o aplicativo possui, sendo a Tela 1 uma tela inicial na qual o usuário seleciona em qual Ambiente o aplicativo está localizado. Já na tela 2 que serve como um informativo para que o usuário consiga compreender como utilizar o aplicativo, e então na tela inicial novamente o usuário pode conceder o acesso a câmera para que possa ser tirado uma foto *selfie* (Tela

[1]A imagem utilizada no protótipo do Figma foi gerada artificialmente por meio do site `http://thispersondoesnotexists.com`

3) e assim que o aplicativo recebe como resposta a verificação de máscara (resposta que parte do modelo) é enviado para uma nova tela em que nela é mostrado a foto como foi tirada e sua resposta. Na Tela 4 ou Tela 5, dependendo da resposta se o usuário está com máscara ou não, é possível acessar a tela informativo para que ,se necessário, o próximo usuário possa entender como utilizar o aplicativo. Nessa tela também é disponibilizado um botão em ícone de câmera para que seja possível tirar uma nova foto para que retorne as Telas 4 e 5 novamente, completando assim o ciclo deste aplicativo.

Diagrama de Caso de Uso

A tarefa de um diagrama de caso de uso UML é mostrar as maneiras de como os Atores podem interagir em um sistema. As ações de cada ator são demonstradas na Figura 5 em conjunto com as funcionalidades do sistema.

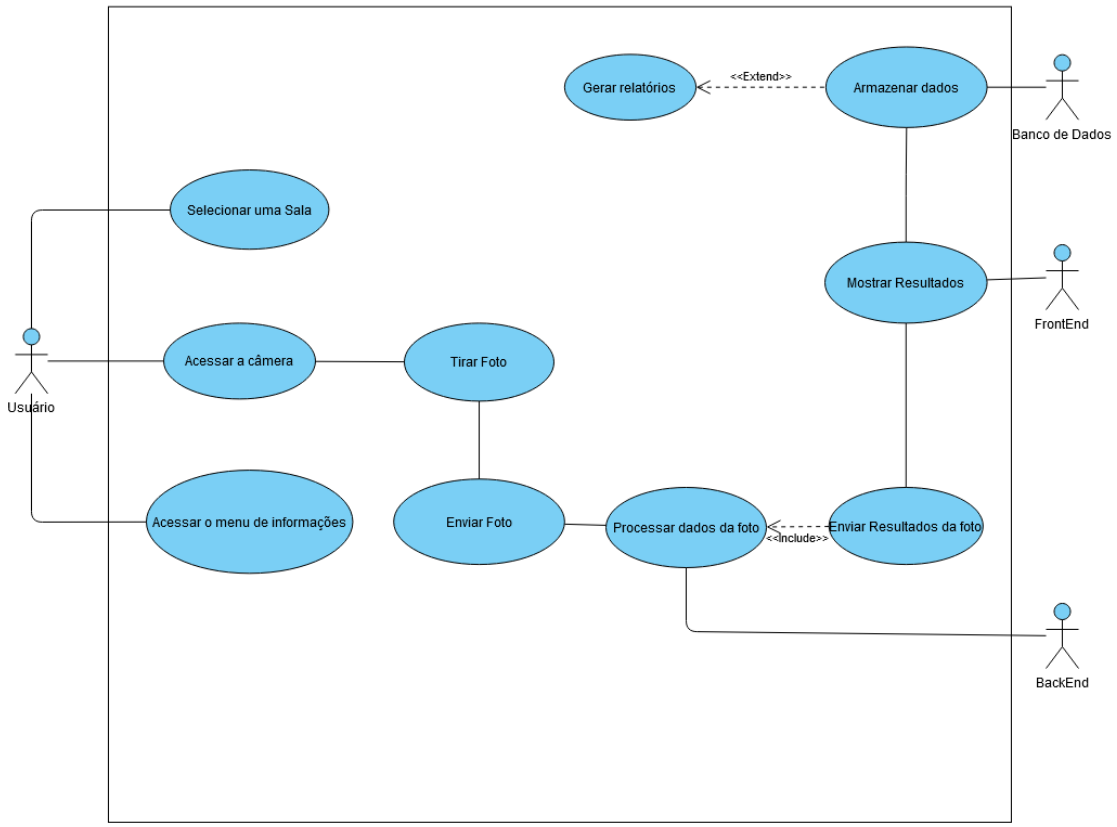

Figura 5. Representação do diagrama de casos de uso UML

O diagrama da Figura 5 apresenta as funções de cada ator, no lado esquerdo foi colocado o usuário e no lado direito foi colocado como atores: banco de dados, *frontend* e *backend*.

O usuário é o responsável pelo uso do *frontend* para que seja possível selecionar uma sala e fotografar o SEU rosto em um dispositivo móvel, onde as salas são trazidas pelo Banco de dados e a imagem é processada pelo *backend* que retorna uma resposta.

O *backend* é responsável em receber a imagem enviada do *frontend* onde é realizada a tarefa de processar a imagem e enviar um *JavaScript Object Notation* (JSON) com os resultados obtidos para o *Frontend*.

O banco de dados é responsável pelo armazenamento dos dados após uma utilização do sistema, ou seja, após todo o processo envolvido com o usuário, posteriormente ao *backend*, o banco de dados trata do armazenamento dessas informações para que seja possível ser feito a geração de relatórios baseado nessas informações armazenadas.

O *frontend* é a interação entre o usuário e o servidor, é a etapa onde vai ser realizado a fotografia por parte do usuário que será enviado para o *backend* retornando uma resposta com os resultados formatados para que se possa ser visualizada.

Diagrama de atividades

O diagrama de atividades da Figura 6 demonstra o fluxo entre o *frontend*, *backend* e o banco de dados, o primeiro círculo preenchido sem contorno representa o início do fluxo e o círculo com contorno branco representa o fim do fluxo, demonstrando como cada etapa interage umas com as outras.

A Figura 4 representa o fluxo de atividades que o sistema tem de acordo com as interações do usuário no aplicativo. Para que tenha um fluxo de atividades, houve uma divisão das atividades relacionadas ao sistema em 3 fases, são elas, o *backend*, *frontend* e o banco de dados, na primeira fase, que é caracterizada pelo início do sistema no *frontend*, o usuário pode interagir com o sistema selecionando uma sala, que essa leva a uma outra fase. Após ser selecionada uma sala o próximo passo diz respeito ao banco de dados e além do sistema de seleção, o usuário pode também iniciar o procedimento de tirar a foto na qual faz parte da comunicação do *frontend* com o *backend* que em função dessa comunicação, ambos compartilham informações para que seja possível o processamento da imagem e retorno dos dados para o *frontend*, dados esses que são mostrados na aplicação e armazenados na parte do banco de dados para que posteriormente seja possível a geração de relatórios baseado nessas informações armazenadas.

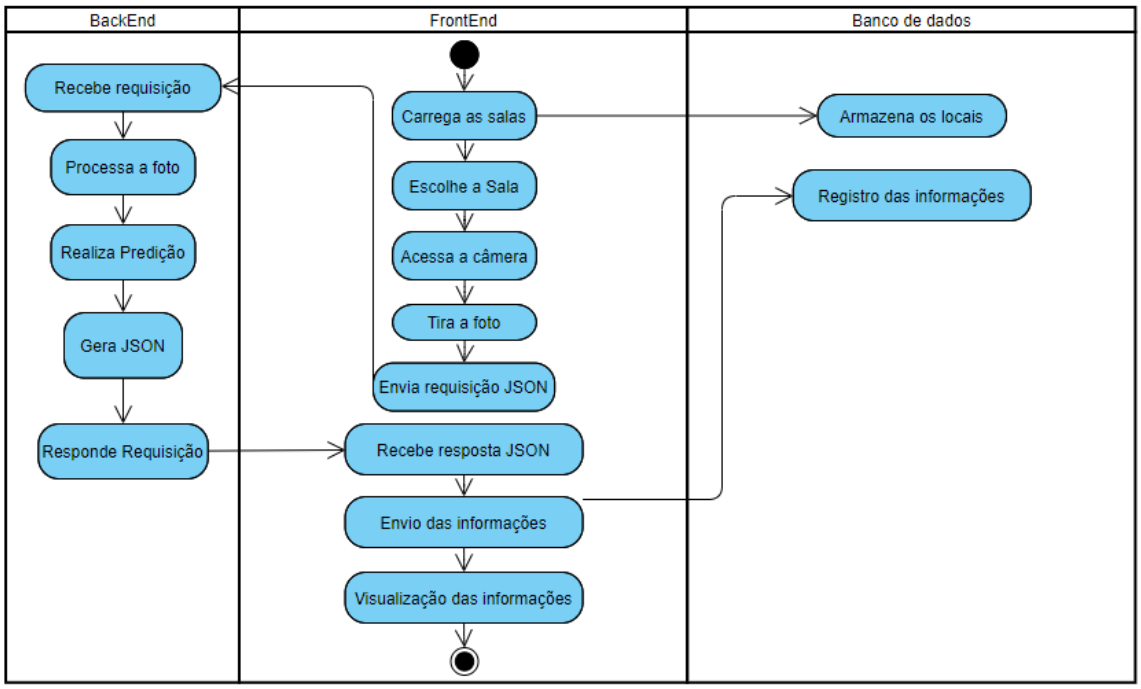

Figura 6: Representação do diagrama de atividade representando o fluxo de interação do usuário na ferramenta.

3.5.3. Implementação

Para a implementação desta aplicação foram utilizadas diferentes tecnologias para *frontend*, *backend* e banco de dados (*firebase*). A seguir são apresentadas cada uma delas.

Frontend

O *frontend* da aplicação *mobile* foi desenvolvido usando o *framework open source Flutter*, feito pelo *Google*. Foi utilizada uma *IDE* (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) *Android Studio* com o suporte para a linguagem *Dart* e extensões para *Flutter* que possibilitou o desenvolvimento do aplicativo.

Na aplicação *frontend* foi utilizado um *plugin* para a captura de imagens através da câmera do dispositivo mobile (sistema android) chamado *Image Picker*, com essa ferramenta é possível que a aplicação *flutter* consiga ter acesso a câmera do dispositivo para que o usuário possa então fotografar-se.

O *frontend* no qual é a parte que o aplicativo se encontra serve como base para que as informações que o usuário insira no sistema entre em contato com o modelo, no caso, como foi feito em *Flutter* o sistema envia uma requisição através da biblioteca *HTTP* que quando utilizada no projeto, possibilitou que o método *POST* fosse utilizado para fazer a requisição em formato JSON para a API (*backend*). Outra biblioteca utilizada em *Flutter* foi a *Firebase* para que possibilitasse o contato com o banco de dados da aplicação. A requisição feita em JSON possui a informação da imagem em *Base64* para que possa ser enviada via requisição para o modelo.

Backend

O modelo foi criado através da linguagem de programação *Python* (Van Ros-

sum e Drake Jr, 1995) que é uma linguagem de programação de alto nível (Van Rossum e Drake Jr, 1995), utilizando bibliotecas como pandas, para manipulação de arquivos,*numpy* para manipulação de matrizes e *sklearn* e *pytorch* para treinar e testar o modelo.

O servidor textitbackend foi desenvolvido em *Python* em conjunto com o *Framework Flask* (Ronacher et al., 2018) para a criação da API, onde são recebidas requisições *JSON* por método *POST*. Dentro desse JSON está contida a imagem em formato *Base64* para que possa ser feita a decodificação da imagem recebida. Posteriormente a imagem decodificada é avaliada pelo modelo e na última etapa é enviado como resposta para o *frontend* o resultado da predição do modelo.

Persistência de Dados e Autenticação

No armazenamento de dados, os usuários e suas entradas de dados são registrados no *Firebase*, nesse armazenamento está o horário e data do acontecimento, a resposta do modelo (com ou sem máscara), a confiança do modelo e a sala selecionada. Essas informações são importantes para uma posterior criação de relatórios.

3.5.4. Avaliação

Na etapa da avaliação foi elaborado um questionário para que fosse possível ter o *feedback* dos usuários sobre o aplicativo desenvolvido. O questionário foi criado a partir da plataforma *Google Forms*, foi formado por questões que caracterizam a amostra, como segmento de atuação no IFSC (aluno, técnico administrativo e docente), o tipo de dispositivo móvel utilizado e o modelo, que questões objetivas avaliam a experiência do usuário em termos de compatibilidade, usabilidade e *performance*. Além disso, o questionário contém duas questões abertas para relatar falhas no aplicativo e sugestão de melhorias pelos respondentes. As questões de caracterização da amostra e compatibilidade são perguntas de múltipla escolha e as questões de usabilidade e *performance* utilizam uma escala linear com notas no intervalo de 1 a 10 (quanto mais alta melhor a nota).

Junto ao questionário foi disponibilizado um vídeo explicativo demonstrativo sobre o aplicativo em questão para que fosse de fácil compreensão para os usuários que respondem questionário, desde a instalação. O aplicativo foi disponibilizado em formato APK (*Android Application Pack*) que pode ser instalado em dispositivos Android. Também foi recomendado que os respondentes realizassem pelo menos seis capturas de imagem, três com máscara e três sem máscara e, posteriormente, respondessem o questionário.

O total de respondentes da avaliação foi de 16 integrantes do IFSC. A amostra foi caracterizada da seguinte forma: em relação ao segmento de atuação no IFSC, 43,8% alunos, 12,5% técnicos administrativos e 43,8% professores; em relação ao dispositivo móvel utilizado, 100% celular e 0% tablet. Os modelos utilizados pelos respondentes são das marcas 50,00% (Samsung), 18,75% (Motorola) e 31,25% (Xiaomi), o aplicativo foi compatível em todos os casos.

O aplicativo apresentou falha em apenas 25% dos respondentes. A descrição das falhas relatadas pelos respondentes decorrem do não preenchimento da sala (local físico) antes de iniciar a captura de imagem, de falha de comunicação do aplicativo com o mo-

delo, e também do aplicativo não ter acertado o uso ou não de máscara em alguns dos testes. Em relação ao itens de usabilidade e *performance* os resultados são apresentados na Tabela 5.

Tabela 5. Resultado da avaliação do protótipo.

Item de avaliação	Mínimo	Máximo	Média	Desvio padrão
Usabilidade				
Facilidade de operar/usar	5	10	8,88	1,36
Facilidade de aprendizado	5	10	8,56	1,63
Interface amigável	7	10	9,19	1,11
Performance				
Tempo da resposta do aplicativo	1	10	8,50	2,22
Qualidade das predições	5	10	8,88	1,36
Tempo de resposta das predições	5	10	8,31	1,74

Com base nos resultados apresentados na Tabela 5 é possível observar que, quanto à usabilidade a menor nota foi 5 em facilidade de operar/usar e em aprendizado e a menor nota na interface foi 7 e maior nota foram 10 para todas as três, com a média de 8,88 (1,36), 8,56 (1,36), 9,19 (1,11), respectivamente. Quanto à *performance*, as menores notas forma 1 para tempo de resposta do aplicativo e 5 para qualidade de predições e tempo de resposta da predições, já a maior nota foi 10 para os três itens, com os desvios padrões de 8,50 (2,22), 8,88 (1,36) e 8,31 (1,74), respectivamente. Observa-se que a menor nota média foi para o item de tempo de resposta das predições, e a maior nota na interface amigável. É importante destacar que o tempo de resposta das predições pode ser melhorado alocando mais recursos de *hardware* na máquina virtual que hospeda a API que realiza as predições, como a possibilidade de uso GPU (*Graphics Processing Unit*).

Em relação ao relato de melhorias do aplicativo, os respondentes relataram os seguintes itens:

- Melhorar a estilização da cor do texto na resposta da predição e confiança;
- Abrir automaticamente a câmera frontal do dispositivo;
- Identificar a face para não haver necessidade de deixar o rosto muito próximo à câmera;
- Avaliar e aprimorar o modelo de reconhecimento em pessoas com características faciais específicas, como barba;
- Melhorar o tempo de resposta das predições.

Os itens relatados pelos respondentes representam a visão do usuário em relação ao uso do aplicativo e, em parte, da qualidade do modelo preditivo. Dessa forma, esses itens poderão ser transformados em trabalhos futuros em requisitos de software e problemas de pesquisa.

4. Conclusão e Trabalhos Futuros

Com o objetivo atingido foi feito um aplicativo para verificação de uso de máscara durante a pandemia causada pelo vírus da COVID-19, tal método propõe uma verificação

facial para determinar se uma pessoa está utilizando ou não alguma máscara de proteção. Considerando que primariamente foi feita uma pesquisa sobre tecnologias sobre redes convolucionais para que seja feito o reconhecimento através de treinamento de imagens de pessoas com e sem máscara para criação de um modelo utilizando redes neurais pré-treinadas.

Nesta pesquisa foram avaliados vários modelos, em que o modelo *Squeezenet* apresentou o melhor custo benefício, entre acurácia e espaço em memória, tendo alcançado 98,04% de acurácia, utilizando apenas 92MB de memória. Foi desenvolvido um aplicativo mobile usando o *framework Flutter*. O aplicativo desenvolvido permite ao usuário selecionar algum dos locais da instituição onde será utilizado, realizar a captura de fotos e a classificação de faces de pessoas com e sem máscara, informando o resultado da classificação ao usuário e registrando esse resultado em uma base de dados.

Durante o desenvolvimento deste trabalho houve diversos desafios, como o *framework Flutter* para o desenvolvimento de aplicativos, bem como a linguagem *Dart*, utilizada nesse ambiente. Flutter tem ganhado maior espaço no mercado desde 2020, após versão v2, onde foram realizadas importantes melhorias de performance, por essa razão ainda não existe tanta documentação, fóruns e bibliotecas de funcionalidades como em outras tecnologias, como *React Native*. Outro desafio foi a pesquisa a respeito de *Transfer learning* e *fine-tuning* em redes neurais, que permitiu treinar o próprio modelo para este trabalho. Por fim, realizar o recorte na imagem da face humana automaticamente para que o usuário não precisasse se aproximar tanto da câmera. Em relação a esse desafio foi possível realizar o recorte em pessoas sem máscara, mas em pessoa com máscara não se obteve sucesso, por esse fato esse recurso não foi incluído neste trabalho.

Trabalhos futuros incluem: a identificação automática de presença de face humana na câmera do aplicativo, pois atualmente a aplicação necessita da intervenção do usuário para tirar a foto; localização das múltiplas faces em uma imagem e classificação das mesmas; geração de relatórios de uso de máscaras por sala; realizar autenticação de usuário ao entrar na aplicação para melhorar a segurança; implementar a geração e gerenciamento de *tokens* para que a API de reconhecimento de máscara possa ser usada apenas pelo aplicativo desenvolvido.

Referências

Adusumalli, H., Kalyani, D., Sri, R. K., Pratapteja, M., e Rao, P. P. (2021). Face mask detection using opencv. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1304–1309. IEEE.

Albawi, S., Mohammed, T. A., e Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee.

Ayyappa, Y., Neelakanteswara, P., Bekkanti, A., Tondeti, Y., e Basha, C. Z. (2021). Automatic face mask recognition system with fcm and bpnn. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1134–1137. IEEE.

Brasil (2020). Recomendação nº 072, de 21 de dezembro de 2020. *Conselho Nacional de Saúde*.

Chollet, F. et al. (2015). Keras. github. Disponível em: `https://github.com/fchollet/keras` . Acesso em 04-02-2022.

Farias, H. S. d. (2020). O avanço da covid-19 e o isolamento social como estratégia para redução da vulnerabilidade. *Espaço e Economia. Revista brasileira de geografia econômica*, (17).

Fasfous, N., Vemparala, M.-R., Frickenstein, A., Frickenstein, L., Badawy, M., e Stechele, W. (2021). Binarycop: Binary neural network-based covid-19 face-mask wear and positioning predictor on edge devices. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 108–115. IEEE.

Ferrero, C. A. (2021). Material de aula da disciplina de tópicos especiais em inteligência artificial do ifsc. Disponível em: `https://anfer86.github.io/teaching-datascience-lessons`. Acesso em 04-02-2022.

Fowler, M. (2018). *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Object Technology Series. Pearson Education.

Gonzalez, R. C. e Woods, R. C. (2009). *Processamento digital de imagens*. Pearson Prentice Hall, 3 ed. edition.

Gurav, O. (2020). Face Mask Detection Dataset. Disponível em: `https://www.kaggle.com/omkargurav/face-mask-dataset`. Acesso em 04-02-2022.

Han, J., Kamber, M., e Pei, J. (2011). *Data Mining: Concepts and Techiniques*. Morgan Kaufmann, 3th edition.

He, K., Zhang, X., Ren, S., e Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., e Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Inamdar, M. e Mehendale, N. (2020). Real-time face mask identification using facemasknet deep learning network. *Available at SSRN 3663305*.

Keras (2021). Api keras reference: Available models. Disponível em: `https://keras.io/api/applications`. Acesso em 04-02-2022.

Krizhevsky, A., Sutskever, I., e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kumar, V. (2021). Face Mask Detection. Disponível em: `https://www.kaggle.com/vijaykumar1799/face-mask-detection`. Acesso em 04-02-2022.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lin, H., Tse, R., Tang, S.-K., Chen, Y., Ke, W., e Pau, G. (2021). Near-realtime face mask wearing recognition based on deep learning. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–7. IEEE.

Loey, M., Manogaran, G., Taha, M. H. N., e Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic. *Measurement*, 167:108288.

Militante, S. V. e Dionisio, N. V. (2020a). Deep learning implementation of facemask and physical distancing detection with alarm systems. In *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, pages 1–5. IEEE.

Militante, S. V. e Dionisio, N. V. (2020b). Real-time facemask recognition with alarm system using deep learning. In *2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, pages 106–110. IEEE.

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., e Hemanth, J. (2021). Ssdmnv2: A real time dnn-based face mask detection system using single shot multibox detector and mobilenetv2. *Sustainable cities and society*, 66:102692.

Nithya Sankari, A. et al. (2021). Face mask detection using convolutional neural network. *INFORMATION TECHNOLOGY IN INDUSTRY*, 9(3):626–628.

Oumina, A., El Makhfi, N., e Hamdi, M. (2020). Control the covid-19 pandemic: Face mask detection using transfer learning. In *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, pages 1–5. IEEE.

Qin, B. e Li, D. (2020). Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19. *Sensors*, 20(18):5236.

Redmon, J. e Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ronacher, A. et al. (2018). Flask (a python microframework). *Dosegljivo: http://flask. pocoo. org.[Dostopano: 20. 7. 2018]*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., e Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Sharma, V. (2020). *Face Mask Detection using YOLOv5 for COVID-19*. PhD thesis, California State University San Marcos.

Simonyan, K. e Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sommerville, I. (2011). *Software Engineering, 9/e*. Dorling Kindersley.

Tan, P.-N., Steinbach, M., e Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, USA, 1 edition.

Van Rossum, G. e Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Werneck, G. L. e Carvalho, M. S. (2020). A pandemia de covid-19 no brasil: crônica de uma crise sanitária anunciada.

X-zhangyang (2021). Real World Masked Face Dataset. Disponível em: `https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset`. Acesso em 04-02-2022.