
**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE TECNOLOGIA**

**DEPARTAMENTO DE TELECOMUNICAÇÕES**

**CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO**

**NEANDER DANUBIO MARINHO ANDRADE**

# DETECÇÃO FACIAL AUTOMÁTICA BASEADA EM REDES NEURAIS CONVOLUCIONAIS

**FORTALEZA**

**2023**

NEANDER DANUBIO MARINHO ANDRADE

# DETECÇÃO FACIAL AUTOMÁTICA BASEADA EM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. VICTOR HUGO COSTA DE ALBUQUERQUE .

FORTALEZA

2023

NEANDER DANUBIO MARINHO ANDRADE

# DETECÇÃO FACIAL AUTOMÁTICA BASEADA EM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

---

Prof. Dr. VICTOR HUGO COSTA DE ALBUQUERQUE (Orientador)
Universidade Federal do Ceará (UFC)

---

MSc Jefferson Silva Almeida
Universidade Federal do Ceará (UFC)

---

MSc Eugenio Peixoto Junior (UFC)
Universidade Federal do Ceará (UFC)

Ao Senhor JESUS CRISTO, que me sustentou todo o tempo, em especial durante o curso. À minha esposa, Fabiana soares, que suportou todas as adversidades do meu lado com paciência. À minha mãe, que Deus a tenha, por todo o esforço em prover as condições necessárias para que eu pudesse me desenvolver. À minha filha que me recebe todos os dias com um lindo sorriso.

## AGRADECIMENTOS

Ao Prof. Dr. Victor Hugo Costa de Albuquerque por me orientar no que concerne ao estágio e ao trabalho de conclusão de curso. Pelo conhecimento e tempo valioso dedicado a ajudar-me. Humildade e conhecimento o definem.

Ao MSc. Jefferson Silva Almeida que me auxiliou e repassou-me conhecimentos valiosos, por seu companherismo e humildade com que muitas vezes me ajudou.

Ao Prof. Dr. Edilson Rocha Porfírio Filho que proporcionou me a oportunidade de atuar como bolsista no programa de iniciação à docência, PID. Experiência extraordinária de aprendizado e superação.

Ao Prof. Dr. Carlos Eduardo Fisch de Brito pela humildade e dedicação em nos oferecer o melhor tratamento e ensino possíveis. Seu amor ao ensino é incrível.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Agradeço a todos os professores que dedicaram seu tempo e esforço para proporcionar conhecimentos valiosos para a minha formação profissional. E também pelo compartilhamento das experiências da vivência profissional. Aos servidores da UFC que mantém a universidade com as melhores condições possíveis e oferecem nos um ambiente excelente! Aos amigos, que são muitos, que nos momentos difíceis auxiliaram me.

“Por que você quer tanto isso? Porque você disse que eu não conseguiria.”
(Men of Honor)

## RESUMO

Dado o grande número de aplicações que utilizam reconhecimento facial, como autorizações de acesso em aplicações web, utilização de serviços públicos, segurança pública, entretenimento etc, este trabalho implementou a detecção e reconhecimento facial em *Python* das redes neurais *MobileNet*, *VGG19* e *VGG16* aplicadas ao conjunto de dados *labeled faces in the wild (LFW)*. Foram empregadas as técnicas de *data augmentation* e *Transfer Learning*. Aquela auxilia no desenvolvimento de um conjunto de treinamento robusto e esta na utilização de modelos com pesos pré-ajustados. Para obter os melhores resultados, o treinamento foi realizado com a técnica de validação cruzada, que visa obter modelos robustos. Quanto aos resultados, obtiveram-se acurácias de 61.0%, 87.0% e 98.0% para as redes *MobileNet*, *VGG19* e *VGG16*, respectivamente. Foram utilizadas redes neurais disponibilizadas nativamente em *Python*, que possuem seus pesos ajustados conforme treinamento prévio com o conjunto de imagens *imagenet*. A arquitetura dos modelos pode ser facilmente adaptada com a adição de novas camadas. A rede *VGG16* se mostrou eficaz para o reconhecimento facial com bom desempenho em termos de tempo de predição.



**Palavras-chave**: Redes Neurais Convolucionais. Reconhecimento Facial. Transferência de aprendizagem.

**ABSTRACT**

Given the large number of applications that use facial recognition, such as access authorizations in web applications, use of public services, public safety, entertainment, etc., this work implemented the detection and facial recognition in *Python* of the neural networks *MobileNet*, *VGG19* and *VGG16* applied to the dataset *labeled faces in the wild (LFW)*. The *data augmentation* and *Transfer Learning* techniques were employed. The former assists in the development of a robust training set and the latter in the use of models with pre-adjusted weights. To obtain the best results, the training was carried out with the cross-validation technique, which aims to obtain robust models. As for the results, accuracies of 61.0%, 87.0% and 98.0% were obtained for the networks *MobileNet*, *VGG19* and *VGG16*, respectively. Neural networks available natively in *Python* were used, which have their weights adjusted according to previous training with the set of images *imagenet*. The architecture of the models can be easily adapted by adding new layers. The *VGG16* network proved to be effective for facial recognition with good performance in terms of prediction time.

**Keywords**: Convolutional Neural Networks. Facial Recognition. Transfer Learning.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| *CNN* | convolutional neural networks/ redes neurais convolucionais |
| *LFW* | labeled faces in the wild/ rostos rotulados na natureza |
| *MLP* | multilayer perceptron/ multicamada perceptron |
| *ROC* | receiver operating characteristics |
| *ROI* | Region Of Interest/ região de interesse |
| *SVM* | Support Vector Machine/ Máquina de Vetores de Suporte |
| RNA | redes neurais artificiais |
| TJCE | Tribunal de Justiça do Estado do Ceará |

**SUMÁRIO**

# 1 INTRODUÇÃO

Nos tempos atuais, a tecnologia permeia a maioria das facetas da vida em sociedade. Grande parte das tarefas do setor produtivo envolve algum tipo de automação, processo computacional etc. O cidadão comum tem acesso a muitas aplicações web ou mobile que facilitam as tarefas cotidianas. Entre elas, pagar uma fatura de cartão de crédito, ter acesso a conta bancária através do *smartphone*, participar de uma reunião de trabalho via aplicação web, contactar parentes com aplicativos de celular etc. Esses avanços em tecnologia impactaram até mesmo nos serviços que o Estado oferece ao cidadão, como fazer a inscrição em um concurso público, pagar contas de água e energia elétrica por meio de celular, marcar uma consulta ou exame através da internet etc. Um serviço que o Estado deve fornecer ao cidadão que é essencial à manutenção da organização social é a segurança pública. Com vistas a prover esse serviço de modo mais eficiente e eficaz, as secretárias de segurança pública dos Estados estão utilizando o monitoramento de pessoas nas vias públicas. Isso poderá ajudar no reconhecimento de pessoas desaparecidas, de suspeitos de crimes etc. O sistema computacional já poderá automaticamente, dado que o reconhecimento fora realizado, verificar a existência, por exemplo, de mandados de prisão para o indivíduo em questão auxiliando assim as forças policiais (SHI *et al.*, 2020).

Com isso, uma grande quantidade de dados deve ser processada, impactando no desempenho dos algoritmos empregados. Logo, é de fundamental importância que os algoritmos aplicados tenham o melhor desempenho possível. Posto isto, justifica-se uma análise mais elaborada de desempenho dos algoritmos mais utilizados nesta tarefa. Em especial, este estudo tem por objetivo avaliar um tipo particular de técnica inteligência facial, que são as redes neurais convolucionais.

## 1.1 Justificativa do estudo

Com o avanço das pesquisas científicas no campo da inteligência artificial, vários métodos de classificação foram desenvolvidos, entre eles Support Vector Machine/ Máquina de Vetores de Suporte (*SVM*), Naive Bayes, redes neurais convolucionais etc. Como exemplo de aplicação de classificação, pode-se citar o reconhecimento facial. Que busca rotular uma dada amostra fornecida ao classificador. O reconhecimento facial já é uma realidade, com aplicação em diferentes aspectos da vida social. No que diz respeito à segurança pública, o reconhecimento facial é executado pelas polícias estatuais do Ceará para o reconhecimento de suspeitos quando

estes não possuem documentos pessoais ou suspeita-se que a identidade repassada seja falsa. Isso é executado através de um aplicativo desenvolvido pela secretária de segurança pública e defesa social, SSPDS, e o instituto federal de educação, ciência e tecnologia do Ceará. Outro exemplo de aplicação é o utilizado pelo Tribunal de Justiça do Ceará, Tribunal de Justiça do Estado do Ceará (TJCE), que busca auxiliar na segurança dos tribunais com o reconhecimento de emoções e expressões faciais de risco em potencial (Tribunal de Justiça do Ceará-CE - TJCE, 2023). Há ainda outros campos em que o reconhecimento de emoção e expressão facial poderá ser utilizado como: jogos *on line*, em que a temática do jogo poderá mudar conforme a percepção das expressões do jogadores; Atendimento de pessoas com deficiência, autismo ou outro transtorno semelhante.

Uma técnica bastante empregada na resolução de problemas de classificação, em especial com amostras compostas por imagem, são as redes neurais. Algumas soluções utilizando redes neurais são aquelas usadas para classificar sementes em um processo de análise de qualidade. Outra, é o reconhecimento facial utilizada pelos governos estudais com vistas a prover o serviço de segurança pública.

## 1.2 Objetivos

### *1.2.1 Objetivos gerais*

Este trabalho tem por escopo principal implementar um sistema *backend* para detecção facial automática com a utilização de redes neurais convolucionais, descrever as técnicas comumente empregadas como *data augmentation*, *transfer learning* e avaliar o desempenho de convolutional neural networks/ redes neurais convolucionais (*CNN*) em termos de tempo de predição e as seguintes métricas:

### *1.2.2 Objetivos específicos*

Obter e verificar as seguintes métricas:

1. Análise do tempo médio de predição por amostra;
2. Análise de desempenho em termos de Acurácia;
3. Análise de desempenho em termos de Precisão;
4. Análise de desempenho em termos de F1-score;
5. Análise de desempenho em termos de Recall,

6. Produção de curvas ROC e AUC para todos os modelos;

### *1.2.3 Organização do trabalho*

O trabalho está dividido da seguinte forma: no **capítulo 2**, fora enunciada breve explicação sobre os fundamentos de redes neurais artificiais, com a conceituação de neurônio artificial, do *perceptron simples*, das redes multilayer perceptron/ multicamada perceptron (*MLP*), e das redes neurais convolucionais.

No **capítulo 3**, a base de dados fora detalhada, com a exposição de algumas amostras com o intuito de demonstrar a complexidade de tal conjunto de dados. Há a demonstração da aplicação das técnicas de *data augmentation*, *tranfer learning* e validação cruzada. Um infograma simples resume o pré-processamento aplicado às amostras. São enunciadas também os modelos empregados neste estudo, quais sejam, *VGG16*, *VGG19* e *MobileNetV2*.

No **capítulo 4**, são mostrados para o teste de predição das redes. As métricas, acima enunciadas, foram avaliadas. Foram produzidos gráficos para matrizes de confusão e curva receiver operating characteristics (*ROC*). Há ainda os valores obtidos para o tempo de predição médio por amostra.

No **capítulo 5**, fora desenvolvida uma breve discussão acerca dos resultados relacionando os aos objetivos propostos. Há ainda uma relação de trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta de modo breve os conceitos essenciais para o entendimento do desenvolvimento do treinamento das redes neurais. Há ainda breve explicação sobre a técnica de *transfer learning*. O capítulo divide-se nas seguintes sessões:

1. Reconhecimento Facial - breve explicação sobre a tarefa de reconhecimento facial;
2. Redes Neurais Convolucionais - breve introdução sobre as CNNs, sua estrutura e aplicações;
3. Transfer Learning - conceituação sobre a técnica de transfer learning.

## 2.1 Reconhecimento Facial

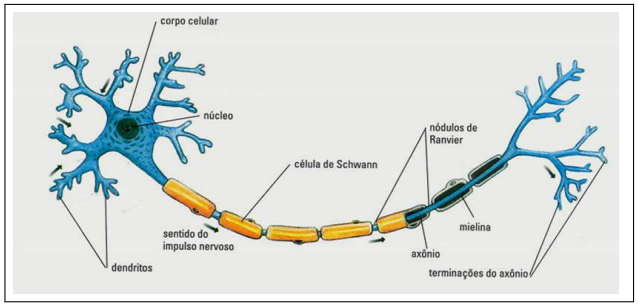
Reconhecimento Facial trata-se de uma aplicação de aprendizado de máquina que tem por escopo, dada uma amostra, rotulá-la conforme classes previamente definidas. Podem-se ser usados alguns modelos de diferentes tipos, como *Support Vector Machine (SVM)*, que é modelo linear, ou redes neurais convolucionais, *CNN*, do inglês. Há inúmeras situações em que o reconhecimento facial poderá ser aplicado, por exemplo, no auxílio à Segurança Pública com câmeras de vigilância, automação de processos com vistas a permissão de acesso etc.

## 2.2 Redes Neurais Artificiais(RNA)

Com o intuito de compreender e desenvolver máquinas inteligentes, pesquisadores da área de inteligência artificial buscaram inspiração nos processos cerebrais do homem. A premissa era entender o funcionamento do cérebro humano e adquirir o conhecimento necessário para aplicar em dispositivos computacionais(FACELI *et al.*, 2011). Assim, é válido conhecer, minimamente, o funcionamento do cérebro para compreender a dinâmica das redes neurais artificiais. O dispositivo principal do funcionamento cerebral é o neurônio. Ele é responsável por processar os sinais elétricos recebidos e estabelecer conexões com outros neurônios, formando centenas de redes neurais, que torna possível ao nosso cérebro realizar as atividades de reconhecer padrões, voz e controle do corpo etc.Ele possui, basicamente, os seguintes componentes: dendritos, axônio e corpo ou *soma*. Os dendritos são ramificações do corpo celular. Eles recebem estímulos externos, que são enviados ao corpo celular. Este por sua vez processa esses sinais e envia as respostas para o axônio. Os neurônios podem fazer conexões com outros através de seus dendritos ou de seu axônio. Conforme os sinais recebidos pelo neurônio, as conexões são

estabelecidas. Esse comportamento é semelhante ao que ocorre com as redes neurais artificiais. Abaixo, um figura com a representação de um modelo de neurônio biológico.

Figura 1 – Neurônio biológico

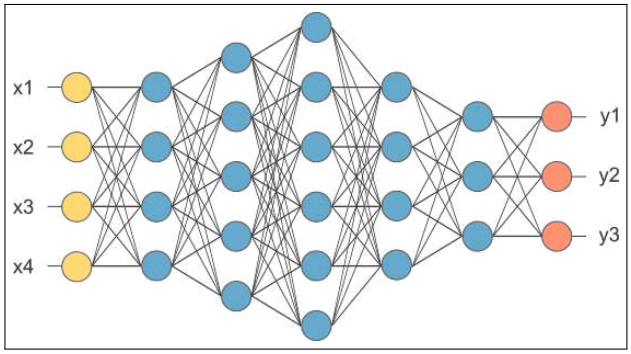



Fonte: (Universidade Federal de Alfena - MG - Histologia Interativa, 2023)

Os pesquisadores McCulloch e Pitts realizaram estudos com intuito de desenvolver um modelo matemático de neurônio artificial. Este poderia resolver funções lógicas simples. O cérebro humano possui bilhões de neurônios que podem se conectar entre si, construindo assim diversas redes. Essa arquitetura complexa é o que permite que o cérebro realize tarefas de reconhecimento de padrões, de voz, percepção etc.

As redes neurais artificiais são compostas de vários neurônios. Eles são dispostos em camadas. Ou seja, neurônios conectam se a outros, que por sua vez, se conecta a outros seguintes etc. A primeira camada é chamada de camada de entrada. É a camada responsável por receber os sinais de entrada. No final da rede, há a camada de saída, que indicará o valor de saída da rede neural. Entre essas camadas, existem as chamadas camadas ocultas. Semelhante ao que ocorre

Figura 2 – Neural com múltiplas camadas ocultas.




Fonte: (Universidade Federal de Ouro Preto - MG - Laboratório Mobilis Computação Móvel, 2017)

com as ligações de neurônios biológicos, que efetuam ligações com base nos sinais recebidos, os neurônios artificias fazem conexões baseadas em pesos que lhe são atribuídos. Esses pesos são

valores reais que podem ser positivos ou negativos a depender se a conexão é de excitação ou inibição respectivamente.

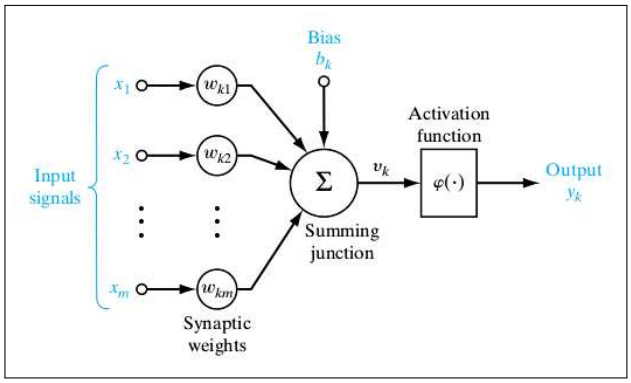
## 2.3 Redes Perceptron Simples

Para mostrar o modelo matemático para o neurônio simples, suponhamos que ele possua d entradas. Seja uma amostra $x$ com seus $d$ atributos $x = [x_1, x_2, x_3, ...x_d]$, $w = [w_1, w_2, w_3, ...w_d]$, que é o vetor de pesos. A equação que modela o comportamento do neurônio é apresentada seguir:

$$u = \sum_{j=1}^{d} x_1 w_j \tag{2.1}$$

O sinal de entrada é inserido na camada de entrada, onde se observam os valores de atributo para uma amostra $x$ conforme a figura. Em seguida, os valores dos atributos são multiplicados pelos pesos w, que são enviados a soma ou corpo celular para efetuar o processamento. Em seguida a saída $v_k$ é colocada na entrada da função de ativação, que produzirá uma saída $y_k$.

Figura 3 – Modelo Matemático do neurônio LTU




Fonte: (Universidade Federal de Ouro Preto - MG - Laboratório Mobilis Computação Móvel, 2017)

## 2.4 Redes multicamadas (MLP)

As redes *MLP* utilizam o conceito de LTU, *unidade linear com threshold*, que é um neurônio artificial demonstrado na figura 3. A estrutura de um *MLP* possui diversas camadas e pode ser empregada para resolver alguns problemas simples, como proposições matemáticas (GéRON, 2019). A seguir é demonstrado um pouco sobre a arquitetura e o

algoritmo de treinamento de uma *MLP*, que serve de fundamento para o treinamento de redes neurais convolucionais.

### *2.4.1 Arquitetura e treinamento*

#### *2.4.1.1 Arquitetura*

As redes *MLP* possuem a característica, no que concerne à arquitetura, de ter todos os neurônios conectados às camadas adjacentes como é mostrado na Figura 2. Essa característica produz uma grande quantidade de parâmetros em relação as *CNN*, o que será mostrado à frente.

#### *2.4.1.2 Treinamento*

O método de treinamento de uma rede *MLP* fora desenvolvido por Frank Rosenblatt, que utiliza o aprendizado Hebbiano. Quando uma amostra consegue rotulá-a corretamente, a conexão entre os neurônios é reforçada e os pesos ajustados (GéRON, 2019). A equação que ajusta os pesos é mostrada a seguir:

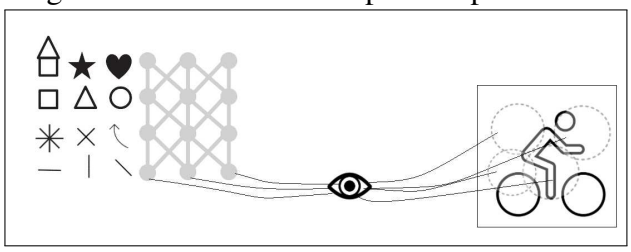
$$w_{i_j} = w_{i_j} + \eta(y_j - y_j)x_i \tag{2.2}$$

## 2.5 Redes Neurais Convolucionais

As redes Neurais Convolucionais surgem, também, sob a inspiração de fenômenos cerebrais humanos. Em especial, o córtex cerebral humano. Um tipo especial de neurônio é o responsável por processar as informações visuais que chegam ao cérebro. Pesquisadores descobriram que tais neurônios são especializados em captar apenas uma parte do campo visual. E que eles detectam um tipo específico de informação. Isto é, recebida uma dada imagem, certos neurônios ficam responsáveis por capturam certos padrões, como linhas retas, linhas curvas, pontos etc (GéRON, 2019). E as saídas desses neurônios são enviadas como entrada para neurônios adjacentes, ou seja, para uma camada seguinte, semelhante ao que ocorre com as RNA. As camadas superiores utilizam o conhecimento adquirido pelas camadas inferiores para assimilar padrões mais complexos. Com isso, é possível para rede reconhecer todos os tipos de padrões, formas e detalhes. A figura a seguir ilustra a ideia de campo receptivo.

As CNNs possuem, além dos componentes de estruturais semelhantes ao das RNA, componentes especiais quais sejam: as camadas convolucionais e as camadas de *pooling*. A

Figura 4 – Modelo de campos receptivos



Fonte: elaborado pelo autor (2023).

seguir, uma breve explicação sobre esses componentes estruturais. Os campos receptivos dos neurônios da primeira camada estão conectados a apenas alguns neurônios da camada de entrada. Ou seja, As CNNs não possuem, em geral, camadas completamente conectadas. O campo receptivo de cada neurônio capta apenas alguma faixa limitada da camada adjacente em busca de reconhecer determinados padrões.

### *2.5.1 Arquitetura*

#### *2.5.1.1 Camadas convolucionais*

As camadas convolucionais são componentes fundamentais das CNNs. Isso acontece devido ao fato que estas camadas não precisam necessariamente estar conectadas a todos os pixels das camadas subjacentes. Esse fato implica em uma quantidade de parâmetros bem menor em relação às redes *MLP* totalmente conectadas. Neurônios de uma determinada camada são conectados à saída da camada vizinha.

## 2.6 Transfer Learning

A linguagem python com suas bibliotecas nativas, como keras, scikit-learn, disponibilizam métodos e classes para a criação de um modelo de rede neural. É possível criar uma rede personalizada com quantas camadas convolucionais, *pooling*, densas for necessário. A camada de saída pode ser ajusta para o número de classes que o problema possuir. Mas em virtude da capacidade de máquinas com grande capacidade de processamento, a quantidade de parâmetros para uma CNN, e conjunto de treinamento a ser utlizado, uma técnica bastante comum empregada é a transferência de aprendizagem, *Transfer learnning*. Ela consiste em utilizar os pesos de um modelo, fornecido nativamente, já ajustados com o treinamento prévio em um conjunto de dados. (YANG *et al.*, 2021), em seu trabalho, avaliou o desempenho de 11 redes neurais, entre elas VGG16 e VGG19, utilizando a técnica de *Transfer Learning* para

o problema de classificação em que as amostras eram de imagens de exames de traumatismo craniano. Devido à complexidade e à quantidade da amostras do conjunto de treinamento e o poder de processamento requerido, a transferência de aprendizagem auxilia em uma melhora no que concerne ao tempo necessário ao treinamento das redes neurais.

# 3 METODOLOGIA

Este capítulo descreve a base de dados utilizada, a técnica de *data augmentation* e a técnica de *Transfer Learning*. Demonstra ainda como foi realizado o treinamento dos modelos com a técnica de validação cruzada.

## 3.1 Descrição da Base de Dados *labeled faces in the wild* (LFW)

A base de dados utilizada neste trabalho fora a *labeled faces in the wild/ rostos rotulados na natureza (LFW)*, que contém 13.233 imagens de personalidades públicas com grande variação de iluminação, expressões e posições. Todas as imagens encontram-se rotuladas em suas devidas pastas. A Figura 5 mostra algumas amostras extraídas da base *LFW*.

Figura 5 – Base de dados *labeled faces in the wild* (LFW).

Fonte: elaborado pelo autor (2023).

As amostras neste conjunto de dados possui uma grande variedade no que concerne ao tipo físico de pessoas. As faces não estão em um ângulo favorável no referencial do observador, há grande variação de posição da inclinação do queixo Figura 6. Algumas amostras possuem ruído, o que é caracterizado pelo borramento da imagem, o que dificulta na extração de características. Outras possuem personagens com acessórios, como chapeis e cabelo, escondem certos detalhes da face Figura 7.

Trata-se pois de um conjunto de dados de complexidade, em comparação com outros conjuntos como *MNIST*, que possui milhares de imagens de números escritos à mão por funcionários de repartições públicas do reino unido.

Figura 6 – Amostras com diferentes posições da face.

Fonte: elaborado pelo autor (2023).

Figura 7 – Amostras com grande oclusão.

Fonte: elaborado pelo autor (2023).

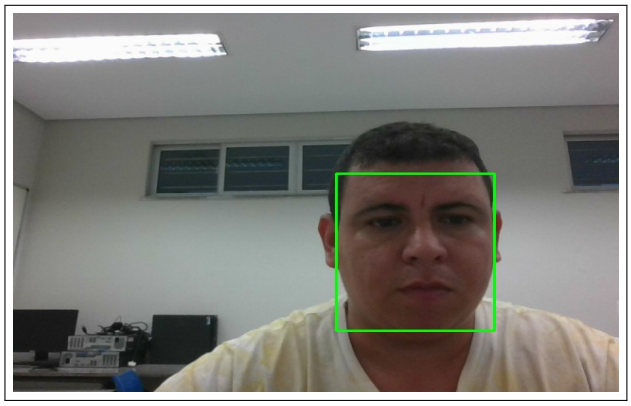
## 3.2 Pré-processamento dos dados

Antes da implementação do treinamento, fora necessário aplicar um pré-processamento nas imagens. Com intuito de evitar a necessidade maior poder computacional, as amostras foram redimensionadas utilizando métodos da biblioteca *OpenCV-python* e *mediapipe* do *Google*.

### *3.2.1 Region Of Interest (ROI)*

Para este experimento, foi construído um novo conjunto de amostras. A aquisição de imagens para a classe positiva ocorreu utilizando também a webcam de um notebook. Mas um pré-processamento nas imagens foi realizado, com o intuito de avaliar o desempenho dos modelos no que tange ao tempo de treinamento e memória. Em virtude de o presente trabalho ter por escopo, como primeira tarefa, a detecção de faces humanas em imagens digitais, alguns detalhes podem ser desnecessários, como fundo da imagem, vestimenta, cor de camisas, membros superiores etc. Dessa forma, é necessário apenas que o rosto seja capturado para servir de amostra para a entrada da rede neural. A equipe do *Google* desenvoleu uma bilbioteca chamada *mediapipe*. Ela possui os métodos process(), detection() e draw_detection() entre outros que traçam a *Region Of Interest/ região de interesse (ROI)*, quais sejam, as faces presentes na imagem Figura 8. Para outros campos de pesquisa, como reconhecimentos de expressões facias e emoções, a *ROI* pode ser escolhida para uma região mais específica do rosto. Por exemplo, através de determinados algoritmos, a região do olho é mapeada em 6 pontos, que serão utilizadas para calcular a *ROI*. (KIM *et al.*, 2018). Para este trabalho, importa detectar as faces presentes em uma imagem, com isso a *ROI* utilizada é um retângulo que envolve toda a face, não apenas um região específica do rosto.

Figura 8 – Localização da região de interesse com mediapipe



Fonte: elaborado pelo autor (2023).

Figura 9 – Extração de ROI

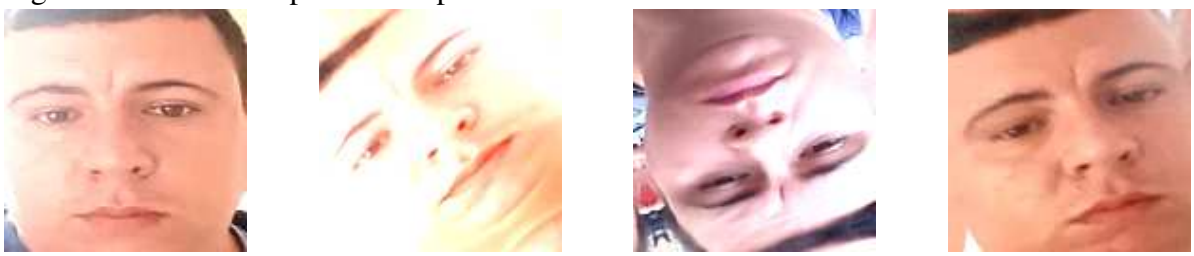
Fonte: elaborado pelo autor (2023).

### *3.2.2 Data Augmentation*

O conjunto total possui duas classes. A classe *Negativa* possui 13.233 amostras, que são todas as amostras de *LFW* e a *Positiva*. A aquisição de amostras para classe positiva se deu utilizanado a webcam de um notebook *Acer, aspire 5*. Cerca de 3000 amostras foram capturadas em diferentes cenários, horário e ilumicação com o intuito de construir um dataset bem representativo. Com isso, inicialmente, o conjunto de dados encontra-se desbalanceado. O que não é adequado, uma vez que pode acrescentar viés aos modelos (POLAT *et al.*, 2021). Para resolver esse problema, fora utilizada a técnica de *Data Augmentation*, que consiste em, a partir da amostra original, produzir novas amostras aplicando métodos de processamento de imagens como rotação, ajuste de brilho etc. A linguagem *Python* já prover biblitecas que facilitam a aplicação de *Data Augmentation*. Depois de executada esta técnica, as classes possuiram as seguintes quantidades: *Positiva:14.381* e *Negativa: 13.233*. Para a classe positiva, como parte do pré-processamento, utilizou-se a técnica de de *data augmentation*. Foram realizados nas amostras rotações, flip horizontal e vertical, zoom e alterações de brilho, semelhantemente, (HE, 2023), usou a técnica *Data Augmentation*. Ele aplicou uma série de processamentos em seus conjuntos de dados, tais como *flip horizontal*, *flip vertical*, rotação, *zoom aletório* e *Shear rage* Figura 10.

Para amostras negativas fez-se a extração da *ROI*, uma vez que detalhes como membros superiores, vestimenta, cor das roupas, acessórios como relógio etc. não identifica necessariamente uma pessoa. Abaixo, a Figura 11 resume de forma ilustrativa todo o pré-processamento aplicado ao conjunto de dados. Um *script Python* que extrai a região de interesse é aplicado às amostras, que estão em disco, de rótulo negativa. E para a outra classe, é feita a aquisição de imagens com uma *webcam*, que passam por um pré-processamento, conforme

Figura 10 – Amostras para classe positiva

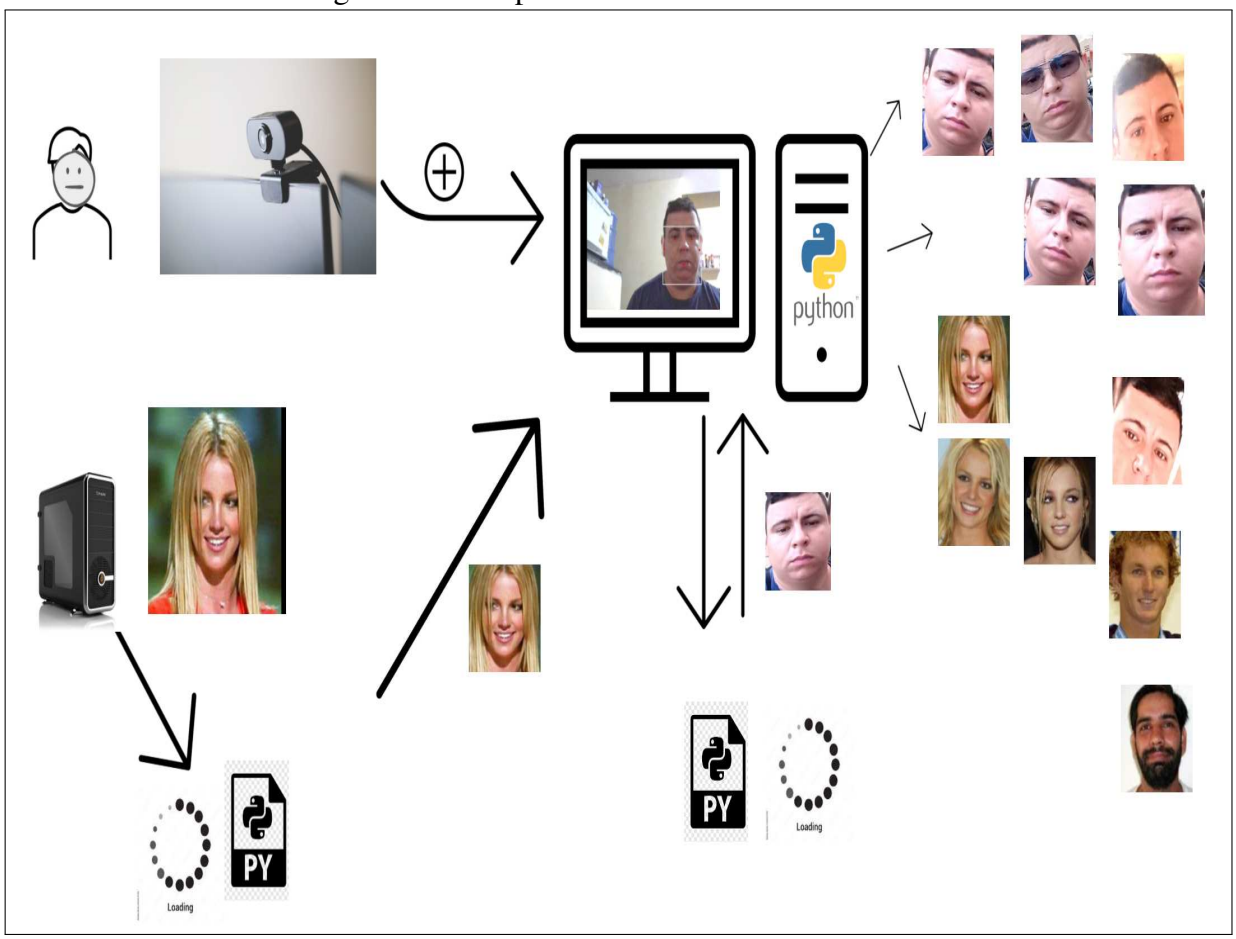


(1) Amostra comum (2) Brilho e rotação (3) Flip Horizontal (4) Zoom

Fonte: elaborado pelo autor (2023).

ilustrado pelo ícone do arquivo *Python*, e são retornadas e adicionadas ao conjunto total de amostras.

Figura 11 – Pré-processamento




Fonte: elaborado pelo autor (2023).

### *3.2.3 Seleção dos subconjuntos*

O conjunto de amostras fora separado em subconjuntos: de treinamento, de validação e de teste. Para efetuar uma separação de amostras aleatoriamente para cada subconjunto fora

utilizados um *script Python* que implementa movimentação de arquivos entre pastas pelo terminal *bash shell* do *linux* adaptado para Windows. Dessa forma, garante-se que não haja modelos treinados em subconjuntos enviesados. Para isso, utilizou-se a bibliotecas *os* nativa do *Python 3.6*.

## 3.3 Descrição de métodos

Inicialmente fora escrito um *script* em *python* para a aquisição das imagens referentes a classe de referência, que indicará um *label positivo* caso amostra seja a de referência. Utilizou-se a biblioteca do google *mediapipe* que detecta se não imagem há uma face humana. Caso isso ocorra, será destacada, com uma moldura retangular, a parte da face detectada. Dessa forma, fora realizada a aquisição de grande parte das imagens de referência. Para as outras imagens que serão rotulados como negativas, fora usado o conjunto de dados público: *LFW*, que possui 13.233 imagens de personalidades públicas, entre elas Barack Obama. A princípio, fez-se, para a classe positiva, apenas a aquisição de 3000 imagens. Para uma melhor representação de um ambiente cotidiano, a aquisição foi efetuada em diferentes ambientes e horários, com isso, fazendo variar o brilho e a iluminação de uma imagem para outra. Para o treinamento com validação cruzada, fora utilizada a classe *GridSearch()*, que é disponibilizada nativamente em *Python*. Dessa forma, a própria classe possui métodos que executam o treinamento. Com isso, o conjunto de dados foi separado em conjunto de treinamento e teste, na proporção de 80% e 20% respectivamente.

Todos os *scripts* foram escritos na *IDE-Pycharm Community Edition 2022.1.3 (Integrated Development Environment)*. Mas, devido ao tempo de treinamento, que é longo no *Pycharm*, a execução do treinamento e teste foi efetuada em *google Colaboratory(Colab)*. Que automaticamente disponibiliza as todas as bibliotecas compatíveis entre si. O que não acontece com o *Pycharm*.

### *3.3.1 As redes neurais*

Foram treinados os seguintes modelos: *VGG16*, *VGG19*, e *MobileNetV2*. Essas redes já são disponibilizadas nativamente pelo *Python*, e já possuem pesos referentes ao treinamento com o conjunto de dados *imagenet*. Dessa forma, toda a arquitetura convolucional já está configurada. Foram adicionadas 4 camadas para cada rede. Uma camada de pooling para redução de dimensionalidade, 2 camadas densas totalmente conectadas e uma camada de saída com 2

neurônios para a classificação binária. A seguir, uma tabela com a quantidade de parâmetros de cada rede.

Tabela 1 – Parâmetros das redes neurais convolucionais

| | Parâmetros treináveis | Parâmetros não treináveis | Total de Parâmetros |
|---|---|---|---|
| VGG16 | 18,890,754 | 14,714,688 | 33,605,442 |
| VGG19 | 18,890,754 | 20,024,384 | 38,915,138 |
| MobileNet | 1,283,002 | 2,257,984 | 3,540,986 |

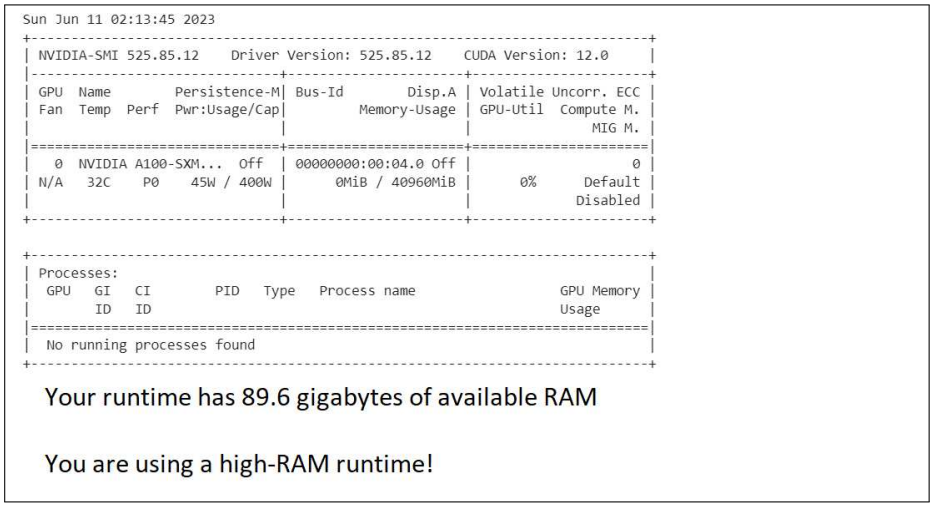
Fonte: elaborado pelo autor (2023).

### *3.3.2 Treinamento dos modelos*

#### *3.3.2.1 Implementação*

Os scripts foram escritos utilizando a IDE *PyCharm*. Devido ao poder de processamento exigido, os modelos foram treinados no *Google Colab*. Os conjuntos de dados foram adicionados ao drive para facilitar o treinamento. Para treinar os modelos em tempo hábil, fora necessário utilizar uma GPU fornecida pelo *Colab* através de uma conta paga. *Colab* forneceu uma placa de vídeo NVIDIA A100 com 40GB de memória RAM, 83GB de memória RAM e 120GB de memória em disco.

Figura 12 – Configuração da GPU utilizada para treinar os modelos com validação cruzada




Fonte: elaborado pelo autor (2023).

#### *3.3.2.2 Validação cruzada com GridSearch*

Uma forma de ajustar o modelo para determinado conjunto de amostras é buscar os melhores parâmetros de tal modo que este alcance os melhores resultado quando da predição de novas amostras. Consiste-se em uma técnica mais robusta, pois executa diversos treinamentos para diferentes valores de parâmetros, tais quais, taxa de aprendizado, épocas, otimizadores etc. O conjunto de dados será divido em dois: conjunto de teste e conjunto de treinamento, na proporção de 20% e 80% respectivamente. A scikit-learn possue uma classe chamada GridSearch, que implementa a função .fit(estimator, param_grid, cv, scoring ...), que tem por parâmetros: cv, que consiste no números de folds em que o conjunto de treinamento será separado para o treinamento. Por exemplo, para cv=5, o conjunto de treinamento é separado em 5 subconjuntos. 4 deles serão treinados e o outro restante será usado para validação. Isso se repete para cada um deles. Obtendo assim melhores resultados de desempenho para treinamento. Estimator é o parâmetro que recebe o modelo a ser treinado. scoring é o argumento que recebe a métrica a ser avaliada para o treinamento. Pode ser utilizado, mais de uma métrica, contudo, o método não retorna um objeto do tipo estimator, ou seja, não retorna um modelo com os melhores hiper-parâmetros. O argumento param_grid recebe um dicionárioPython, que consiste nas variáveis em que o modelo possue e que o GridSearch visa encontrar a combinação ótima. O retorno do função .fit() é um clone do modelo com os melhores parâmetros encontrados. Utilizando a instrução:

Código-fonte 1 – Criação do objeto GridSearch

```
1 gridsearch = GridSearch(estimator=model_vgg16, param_grid=
    param, scoring, cv=5)
```

O objeto do tipo GridSearch é criado. Em seguida, o modelo é treinado:

Código-fonte 2 – Treinamento do modelo

```
1 gridsearch,fit(X_train,y_train)
```

Onde são passados as amostras, X_train, e os seus respectivos rótulos, y_train. Finalmente, o melhor modelo é criado:

Código-fonte 3 – Melhor modelo

```
best_model = gridsearch.best_estimator_
```

Dado que obteve-se um modelo com os melhores hiper-parâmetros, é possível documentar todas as configurações avaliadas e dispô-las em arquivo .txt para futuras avaliações. Para isso, utilzam-se outras instruções fornecidas pela classe GridSearch. Os métodos são: best_param, best_score, e cv_results.

Em virtude da alta capacidade de processamento requerida para este estudo, utilizar-se-á os seguinte valores de hiper-parâmetros:

1. Épocas = [5, 10];
2. Folds = [5];
3. E taxa de aprendizado = [0.2, 0.1 e 0.01].

## 3.4 Métricas de desempenho

É necessário alguma forma de avaliar o desempenho dos modelos durante o treinamento e teste. Há uma grande variedade de métricas utilizadas para tal finalidade, mas de modo geral, as métricas mencionadas abaixo são as mais utlizadas.

### *3.4.1 Matriz de confusão*

A matriz de confusão é uma ferramenta gráfica que auxilia na rápida percepção do desempenho de um classificador. Ela é disposta em forma de matriz quadrada, onde suas linhas e colunas são compostas pelas classes envolvidas no problema. Logo, ela possui o seguinte formato: Neste estudo, o classificador possui a tarefa de verificar amostras que possuem duas classes-alvo. Dessa forma, terá-se uma matriz $M_{2,2}$, cuja células possuem os valores computados para classe. As linhas e colunas serão nomeadas com os rótulos das classes. Para as colunas, terá-se os valores preditos pelo classificador. Para as linhas, os valores verdadeiros dos rótulos, a figura abaixo exemplifica uma matriz de confusão para um problema binário.

A partir da matriz de confusão, pode se extrair todos os valores possíveis para as classes. Os verdadeiros positivos indicam a quantidade de amostras positivas corretas que o classificador conseguiu obter. Ou seja, a amostra possuía rótulo verdadeiro e o modelo conseguiu predizer que a amostra era realmente positiva. Para falso positivo, o modelo indica que a amostra possui rótulo positivo, mas na verdade ela possui rótulo negativo. Ocorre semelhantemente para a

Tabela 2 – Matriz de confusão para classificação binária

| Matriz de Confusão | | | |
|---|---|---|---|
| | | Rótulos Preditos | |
| | | Positivo | Negativo |
| Rótulos Reais | Positivo | Verdadeiro Positivo | Falso Negativo |
| | Negativo | Falso Positivo | Verdadeiro Negativo |

Fonte: elaborado pelo autor (2023).

classe negativa. Um verdadeiro negativo indica que o modelo classificou corretamente a amostra negativa, e falso negativo indica que o modelo errou para ela. Com base nesses valores, obtém-se as taxas, que são métricas importantes na avaliação de um modelo. Para alguns problemas, as classes possuem importâncias diferentes. Por exemplo: para um classificador que prediz se um paciente possui ou não câncer, os valores de falso negativo podem críticos, uma vez que se a classe positiva é estar com câncer, um falso negativo indica que o paciente foi classificado como saudável pelo modelo, mas na realidade ele está enfermo. Dito isto, as métricas utilizadas para avaliar um modelo depender do problema em análise.

$$Precision = \frac{TP}{TP+FP} \tag{3.1}$$

$$Acuracy = \frac{TP+TN}{TP+TN+TF+NF} \tag{3.2}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

$$F_1 - score = \frac{TP}{TP+\frac{TP+FN}{2}} \tag{3.4}$$

### *3.4.2 Hiper-parâmetros*

As redes foram treinadas utilizando a validação cruzada. Para cada configuração, foi obtido um modelo, que é comparado aos demais em termos de acurácia de treinamento. Os hiper-parâmetros utilizados foram: épocas e taxa de aprendizagem. E para todos foram utilizados 5 folds. Abaixo são mostradas as melhores configurações para cada modelo. Os pesos que foram salvos para o modelo definitivo foram obtidos para a configuração que obteve a melhor acurácia.

1. Para*VGG16*: Melhores parâmetros encontrados: ’epochs’: 5, ’lr’: 0,1 Melhor acurácia encontrada: 0,994
2. Para *VGG19*: Melhores parâmetros encontrados: ’epochs’: 5, ’lr’: 0,2 Melhor acurácia encontrada: 0,990
3. Para *MobileNet*: Melhores parâmetros encontrados: ’epochs’: 10, ’lr’: 0,2 Melhor acurácia encontrada: 0,992

# 4 RESULTADOS

Os modelos foram submetidos ao teste de predição, em que novas amostras são apresentadas aos modelos e este indica a qual rótulo cada amostra pertence. Os resultados obtidos são mostrados em forma de tabelas, matrizes de confusão e curvas *ROC*.

## 4.1 Resultados com validação cruzada

Para um modelo ideal, que acertaria todas as predições para amostras do conjunto de teste, a matriz de confusão possuiria todos os valores nulos exceto para a diagonal principal. E a curva *ROC*, possuiria um formato de reta horizontal, com taxa de verdadeiros positivos igual 1, para taxas de falsos positivos maior ou igual a 0. Somente a rede VGG16 chegou próximo a esse padrão ideal. O erro foi de 3 para falso positivos e 25 falso negativos. A rede VGG19 obteve erro de 352 para falsos negativos, um valor considerável, inaceitável para aplicações de vigilância por exemplo. A rede MobileNet obteve os piores resultados com 845 para falsos negativos e 229 para falsos positivos.

A rede VGG16 obteve bons resultados, com mais de 98% de acurácia figura 13. Mas a VGG19, submetida aos mesmos parâmetros de treinamento, obteve desempenho inferior, com acurácia de 87% figura 17. Vale salientar que todos os três modelos obtiveram acurácia acima de 99% durante o treinamento. O que pode indicar que ocorreu o fenômeno de *overfitting*, quando o modelo se ajusta demais às amostras de treinamento. E durante a predição, com amostras não antes vistas, a rede obtem um baixo desempenho.

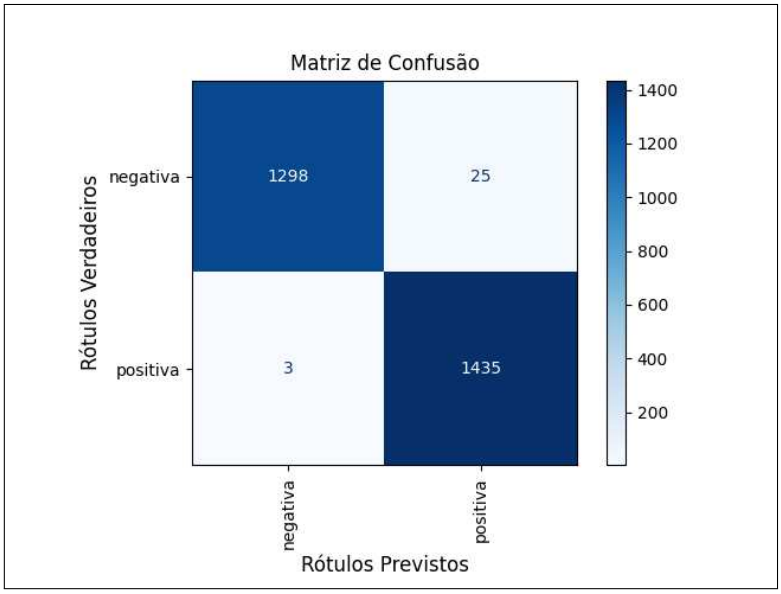
Hamming loss: 0.010141 Accuracy: 0.989859 Precision Macro: 0.990285 Recall Macro: 0.989509 F1-score: 0.989855 Média do tempo de predição: 0.286256 +/- 0.199098 seconds Mediana do tempo de predição: 0.163926 seconds Máximo do tempo de predição: 0.860744 seconds Mínimo do tempo de predição: 0.106289 seconds

Tabela 3 – Resultado da predição com VGG16 por classe

| Rótulos | precision | recall | f1-score | support |
|---|---|---|---|---|
| Negativa | 1.00 | 0.98 | 0.99 | 1323 |
| Positiva | 0.98 | 1.00 | 0.99 | 1438 |
| | | | | |
| Accuracy | | | 0.99 | 2761 |
| macro avg | 0.99 | 0.99 | 0.99 | 2761 |
| weighted avg | 0.99 | 0.99 | 0.99 | 2761 |

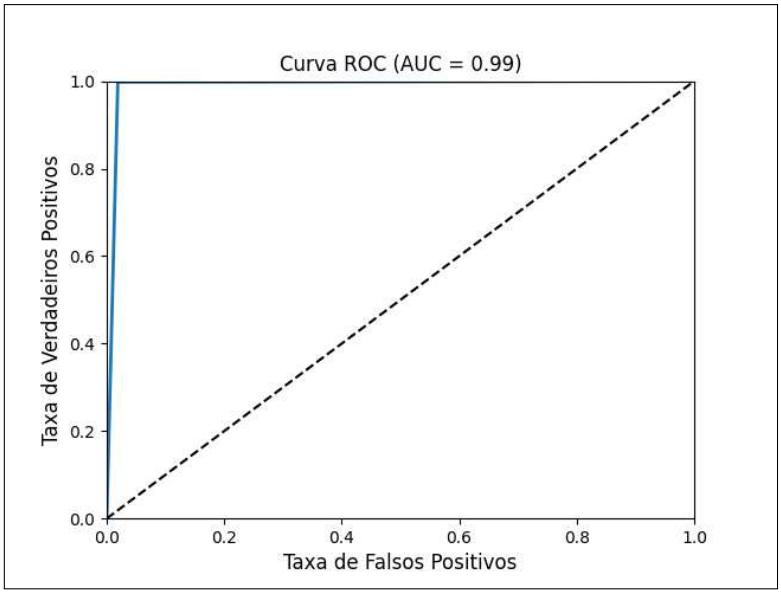
Fonte: elaborado pelo autor (2023).

Figura 13 – Matriz de Confusão para a rede VGG16 com validação cruzada.




Fonte: elaborado pelo autor (2023).

Figura 14 – Curva ROC para a rede VGG16 com validação cruzada.




Fonte: elaborado pelo autor (2023).

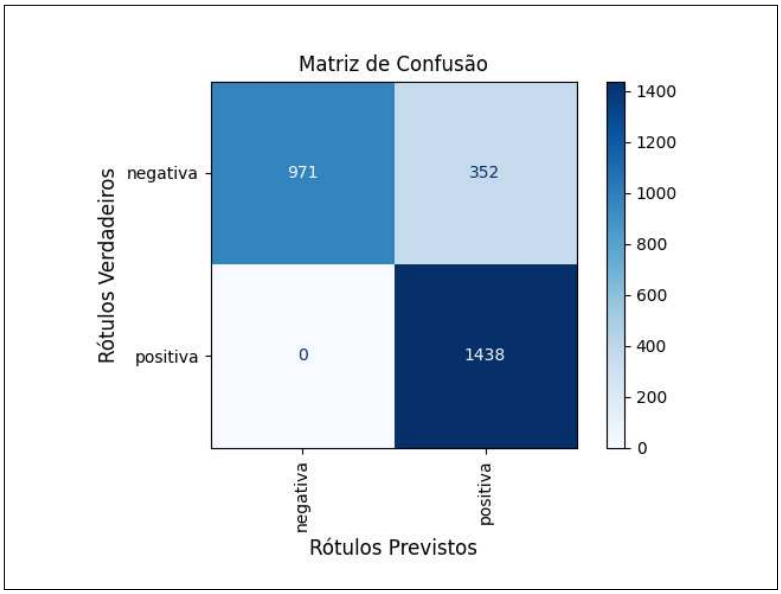
Melhor configuração para o treinamento da VGG19: ———————————— Modelo: VGG19 Data e hora: 2023-06-11 18:27:32 Melhores parâmetros encontrados: 'epochs': 5, 'lr': 0.2 Melhor pontuação encontrada: 0.9902439024390244 Hamming loss: 0.127490 Accuracy: 0.872510 Precision Macro: 0.901676 Recall Macro: 0.866969 F1-score: 0.869680 Média do tempo de predição: 0.300765 +/- 0.182452 seconds Mediana do tempo de predição: 0.205718 seconds Máximo do tempo de predição: 0.856426 seconds Mínimo do tempo de predição: 0.125874 seconds

Tabela 4 – Resultado da predição com VGG19 por classe

| Rótulos | precision | recall | f1-score | support |
|---|---|---|---|---|
| Negativa | 1.00 | 0.73 | 0.85 | 13223 |
| Positiva | 0.80 | 1.00 | 0.89 | 1438 |
| | | | | |
| Accuracy | | | 0.87 | 2761 |
| macro avg | 0.90 | 0.87 | 0.87 | 2761 |
| weighted avg | 0.90 | 0.87 | 0.87 | 2761 |

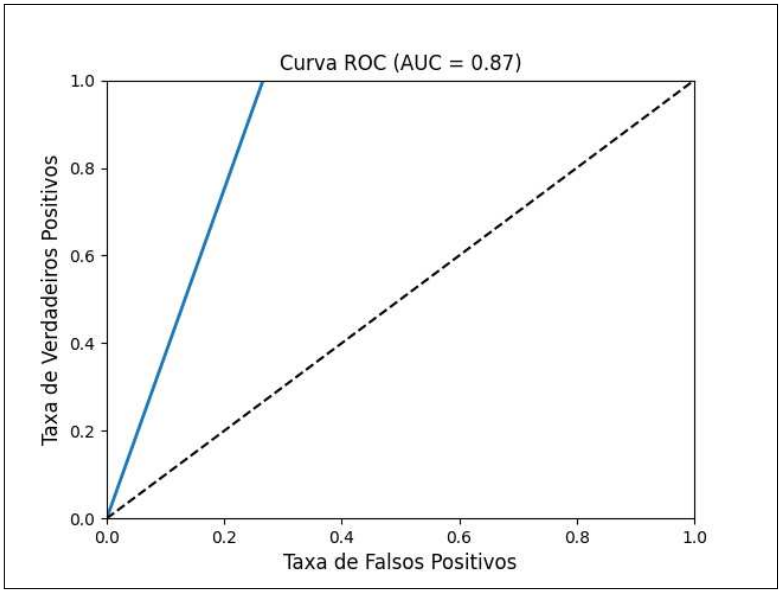
Fonte: elaborado pelo autor (2023).

Figura 15 – Matriz de Confusão para a rede VGG19 com validação cruzada.




Fonte: elaborado pelo autor (2023).

Figura 16 – Curva ROC para a rede VGG19 com validação cruzada.




Fonte: elaborado pelo autor (2023).

Para rede *MobileNet*, obteve-se o seguinte resultado:

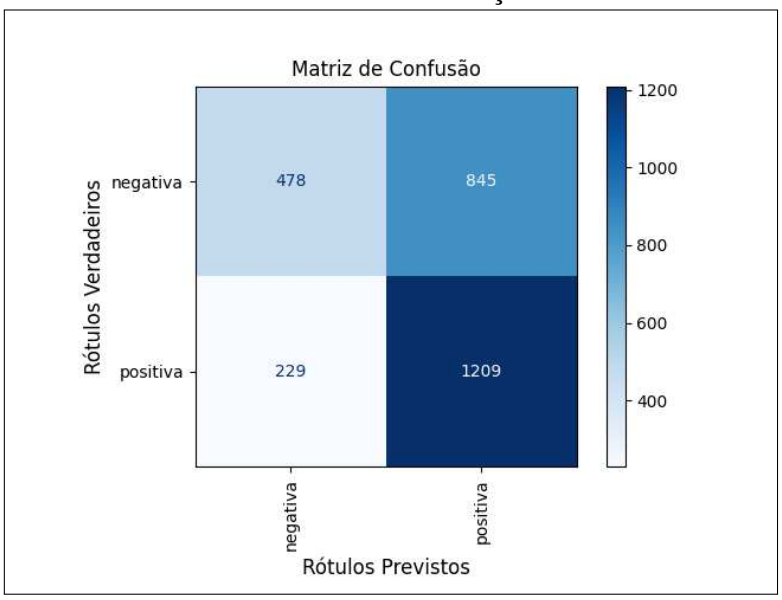
Hamming loss: 0.388989 Accuracy: 0.611011 Precision Macro: 0.632352 Recall Macro: 0.601026 F1-score: 0.586301 Média do tempo de predição: 0.201040 +/- 0.050955 seconds Mediana do tempo de predição: 0.196039 seconds Máximo do tempo de predição: 0.624630 seconds Mínimo do tempo de predição: 0.047093 seconds

Tabela 5 – Resultado da predição com MobileNet por classe

| Rótulos | precision | recall | f1-score | support |
|---|---|---|---|---|
| Negativa | 0.68 | 0.36 | 0.47 | 1323 |
| Positiva | 0.80 | 1.00 | 0.89 | 1438 |
| | | | | |
| Accuracy | | | 0.61 | 2761 |
| macro avg | 0.63 | 0.60 | 0.58 | 2761 |
| weighted avg | 0.63 | 0.61 | 0.59 | 2761 |

Fonte: elaborado pelo autor (2023).

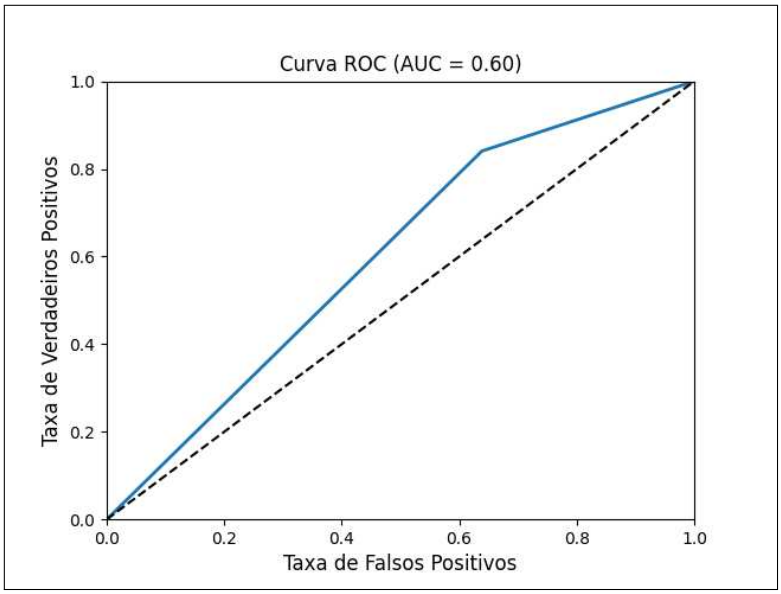
Figura 17 – Matriz de Confusão para a rede MobileNet com validação cruzada.




Fonte: elaborado pelo autor (2023).

Para a rede vgg16, (YANG *et al.*, 2021) obteve x% de acurácia

Figura 18 – Curva ROC para a rede MobileNet com validação cruzada.




Fonte: elaborado pelo autor (2023).

# 5 CONCLUSÕES E TRABALHOS FUTUROS

## 5.1 Conclusão

Os resultados mostraram que apenas a rede VGG16 conseguiu assimilar, de modo satisfatório, as características do conjunto de dados. Em parte devido à complexidade do conjunto de imagens, que possuem uma grande variação entre si. Com intuito de construir um conjunto de imagens que representasse mais situações reais possíveis, o dataset tornou-se bastante complexo. Por exemplo, (POLAT *et al.*, 2021), com seu modelo de rede neural, chamado nCov-NET, alcançou uma acurácia de 97.1% para classificar imagens obtidas de raio-x de pacientes que apresentaram covid-19. Vale salientar, que essas imagens possuem uma menor variação entre si em comparação ao conjunto utilizado neste estudo. Similarmente, (YANG *et al.*, 2021) obteve em seu estudo, que se propunha a desenvolver modelos de CNN, baseados nas redes *MobileNet* e *ResNet*, para diagnosticar Covid-19 a partir de imagens de radiografia de tórax e tomografia computadorizada, acurácias de 99.3% e 99.6%. Dessa forma, julga-se que a rede VGG16 alcançou um bom resultado neste estudo. Vale salientar, que com a utilizando de apenas 2 hiper-parâmetros, taxa de aprendizagem com 3 valores: [0.2, 0.1 e 0.01] e épocas: [5 e 10], e 5 folds eram necessários 60 fits, que são a quantidade treinamentos para escolher a melhor configuração dos pesos e parâmetros de cada modelo. O que evidencia ser necessário um grande poder de processamento e tempo gasto para treinar as redes neurais.

## 5.2 Trabalhos Futuros

Em alguns países como a China, possuem um sistema de vigilância pública capaz, em tempo real, de detectar pessoas e verificar lhe a identidade com certa probabilidade. Como visto neste experimento, isso requer um grande poder de processamento. Dito isto, é interessante desenvolver modelos baseados em *CNN*, que possuem uma menor quantidade de parâmetros, assim, exigindo menor quantidade de poder computacional.

As algumas pesquisas mais recentes tem por escopo desenvolver nos algoritmos e arquiteturas para redes neurais convolucionais que sejam capazes de reconhecer expressões faciais e que denotam emoções como medo, angustia, tristeza, ansiedade entre outras (HE, 2023). Estes trabalhos podem auxiliar, por exemplo, no diagnóstico e tratamento de crianças com autismo, auxiliar na identificação de pessoas com depressão etc. Isto posto, pode-se propor

um sistema de vigilância que pode auxiliar na prevenção de crimes e abusos cometidos contra crianças que necessitem frequentar creches e abrigos através da análise das expressões e emoções.

# REFERÊNCIAS

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. de. **Inteligência Artificial - Uma abordagem de aprendizado de máquina**. [*S. l.*: *s. n.*], 2011.

GéRON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn TensorFlow Conceitos, Ferramentas e Técnicas para a Construção de Sistemas Inteligentes**. [*S. l.*: *s. n.*], 2019.

HE, Y. Facial expression recognition using multi-branch attention convolutional neural network. **IEEE Access**, v. 11, p. 1244–1253, 2023.

KIM, M. C.; KOO, J. H.; CHO, S. W.; BAEK, N. R.; PARK, K. R. Convolutional neural network-based periocular recognition in surveillance environments. **IEEE Access**, v. 6, p. 57291–57310, 2018.

POLAT, ; KARAMAN, O.; KARAMAN, C.; KORKMAZ, G.; BALCı, M. C.; KELEK, S. E. Covid-19 diagnosis from chest x-ray images using transfer learning: Enhanced performance by debiasing dataloader. **Journal of X-ray Science and Technology**, v. 8, p. 19–36, 2021.

SHI, M.; XU, L.; CHEN, X. A novel facial expression intelligent recognition method using improved convolutional neural network. **IEEE Access**, v. 8, p. 57606–57614, 2020.

Tribunal de Justiça do Ceará-CE - TJCE. **Parceria entre Tribunal de Justiça do Ceará e IFCE vai ampliar uso de tecnologia e IA no Judiciário**. 2023. Disponível em: https://www.tjce.jus.br/noticias/parceria-entre-tribunal-de-justica-do-ceara-e-ifce-vai-ampliar-uso-de-tecnologia-e-ia-no-judiciario/.

Universidade Federal de Alfena - MG - Histologia Interativa. **"Tecido Nervoso"**. 2023. Disponível em: https://www.unifal-mg.edu.br/histologiainterativa/wp-content/uploads/sites/38/2019/09/. Acesso em: 16 mai. 2023.

Universidade Federal de Ouro Preto - MG - Laboratório Mobilis Computação Móvel. **Fundamentos De Redes Neurais**. 2017. Disponível em: http://www2.decom.ufop.br/imobilis/fundamentos-de-redes-neurais/. Acesso em: 17 mai. 2023.

YANG, Y.; ZHANG, L.; DU, M.; BO, J.; LIU, H.; REN, L.; LI, X.; DEEN, M. A comparative analysis of eleven neural networks architectures for small datasets of lung images of covid-19 patients toward improved clinical decisions. **Comput Biol Med**, p. 139–104887, 2021. Epub 2021 Sep 24. PMID: 34688974; PMCID: PMC8461289.