

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR FRANCISCO RAMOS

**IDENTIFICAÇÃO BIOMÉTRICA DE PESSOAS POR MEIO DO
RECONHECIMENTO FACIAL UTILIZANDO *VISION
TRANSFORMERS***

BAURU
Novembro/2023

ARTHUR FRANCISCO RAMOS

**IDENTIFICAÇÃO BIOMÉTRICA DE PESSOAS POR MEIO DO
RECONHECIMENTO FACIAL UTILIZANDO *VISION
TRANSFORMERS***

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Orientador: Prof. Dr. Aparecido Nilceu Marana

BAURU
Novembro/2023

Arthur Francisco Ramos

Identificação Biométrica de Pessoas por Meio do Reconhecimento Facial Utilizando *Vision Transformers*

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Aparecido Nilceu Marana
Universidade Estadual Paulista "Júlio de
Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Dra. Simone Domingues Prado
Universidade Estadual Paulista "Júlio de
Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Kelton A. Pontara da Costa
Universidade Estadual Paulista "Júlio de
Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 14 de novembro de 2023.

Agradecimentos

Agradeço inicialmente a minha família, por todo o apoio e confiança que me proporcionaram e que me permitiram hoje estar aqui e ser quem eu sou, e não sendo suficiente chamar apenas de "família", sou obrigado a citá-los nominalmente: Angela, Mayara, Matilde, João, Israel, Marilza, Salvador, Juan, Claudia, Bruna e Sara.

Agradeço também aos meus amigos, Danilo, João, Nathan, Renato e Ronaldo, que me acompanharam desde o primeiro semestre, em períodos turbulentos de pandemia, até o momento de hoje, próximo ao encerramento de um dos ciclos mais importantes da minha vida, minha graduação.

Agradeço imensamente também a amigos, que por mais que não estejam em meu ciclo diário da graduação, sou imensamente grato a todos os momentos que compartilhamos e que ainda compartilharemos juntos, em especial ao Caio, Isadora, Júlia, Leon, Samuel e Thiago.

Sou grato aos meus professores e por todos os ensinamentos que me foram passados ao longo desses quatro anos em salas de aula e laboratórios, não podendo esquecer também das salas virtuais do *Meet*, *Classroom* e *Moodle*. Dedico agradecimento especial ao meu orientador, Nilceu, que das aulas de Informação Profissional e Acadêmica às aulas de Banco de Dados, me espelho no pesquisador e profissional que você é.

E por fim, e não menos importante, agradeço a UNESP, pois realizei um sonho de estudar o que mais amo em uma das maiores universidades do país, e que me proporcionou as melhores experiências e oportunidade que poderia desejar, que vão de conhecer meus amigos e colegas de sala até conhecer outros lugares do país representando minha segunda casa.

Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.

Alan Turing

Resumo

A biometria se tornou uma das formas mais seguras na tarefa de reconhecimento de indivíduos, sendo o reconhecimento facial um dos problemas clássicos na área da visão computacional. Proposto pela primeira vez há 50 anos, os sistemas de reconhecimento automático de rostos passaram por diversas mudanças ao longo do tempo, desde algoritmos tradicionais até o uso de aprendizado de máquina profundo, com destaque nas redes neurais convolucionais, que hoje predominam as pesquisas na área. Visando novas alternativas de métodos para a tarefa de reconhecimento facial, este trabalho propôs avaliar o desempenho de uma arquitetura baseada em transformadores e autoatenção com foco em imagens, o *Vision Transformer*, em ambientes controlados e não controlados, além do desenvolvimento de uma aplicação completa para analisar o funcionamento do modelo de forma prática. Para atingir tal objetivo, a metodologia aplicada consiste no uso de técnicas de detecção e alinhamento facial, para aperfeiçoar o treinamento e validação do modelo de reconhecimento, em conjunto com métodos de análise comuns a sistemas de identificação e verificação, a fim de mensurar o desempenho da arquitetura proposta na resolução do problema de reconhecimento facial. Os resultados demonstraram que o *Vision Transformer* é capaz de desempenhar a função de reconhecimento com eficácia, todavia apresentando algumas limitações em ambientes com maior instabilidade de iluminação e variações de expressões faciais, principalmente devido ao tamanho limitado das bases de dados de imagens utilizadas, mas não prejudicando a experiência do usuário e a confiabilidade do aplicativo desenvolvido.

Palavras-chave: Aplicativo; Aprendizado de Máquina; Biometria; Reconhecimento Facial; *Vision Transformer*.

Abstract

Biometrics has become one of the most secure ways to identify individuals, with facial recognition being one of the classic problems in the field of computer vision. First proposed 50 years ago, automatic face recognition systems have undergone various changes over time, from traditional algorithms to the use of deep learning, with a focus on convolutional neural networks, which today dominate research in the field. Aiming at new alternatives for methods for the task of facial recognition, this work proposed to evaluate the performance of an architecture based on transformers and self-attention focused on images, the Vision Transformer, in controlled and uncontrolled environments, in addition to the development of a complete application to analyze the functioning of the model in a practical way. To achieve this goal, the methodology applied consists of the use of facial detection and alignment techniques, to improve the training and validation of the recognition model, together with common analysis methods for identification and verification systems, in order to measure the performance of the proposed architecture in solving the problem of facial recognition. The results showed that the Vision Transformer is capable of performing the recognition function effectively, however presenting some limitations in environments with greater lighting instability and variations in facial expressions, mainly due to the limited size of the image databases used, but not compromising the user experience and the reliability of the application developed.

Keywords: Application; Biometrics; Facial Recognition; Machine Learning; Vision Transformer.

Listas de figuras

Figura 1 – Estrutura de uma CNN.	16
Figura 2 – Estrutura de um ViT.	17
Figura 3 – Exemplos de Mapas de Atenção.	18
Figura 4 – Processo de Pré-Processamento e Reconhecimento Facial.	19
Figura 5 – Exemplo de Detecção com <i>BlazeFace</i> .	19
Figura 6 – Processo de Alinhamento Facial.	20
Figura 7 – Estrutura do Modelo <i>FaceNet</i> .	21
Figura 8 – Processo de Reconhecimento Facial usando ViT.	22
Figura 9 – Exemplos de Curvas ROC.	24
Figura 10 – Exemplos de Curvas CMC.	25
Figura 11 – Comparação entre <i>Softmax</i> e <i>ArcFace</i> .	26
Figura 12 – Processo de Verificação Facial.	26
Figura 13 – Exemplos de Imagens da FRGC.	29
Figura 14 – Exemplos de Imagens da LFW.	29
Figura 15 – Exemplos de Imagens da <i>Faces94</i> .	29
Figura 16 – Fluxograma da Metodologia.	30
Figura 17 – Exemplo de Pré-processamento usando <i>BlazeFace</i> .	31
Figura 18 – Exemplo de Reconhecimento Facial usando ViT.	31
Figura 19 – Curva de Perda no Treinamento.	33
Figura 20 – Curva ROC, <i>AUC</i> e <i>EER</i> na FRGC.	35
Figura 21 – Curva CMC na FRGC.	35
Figura 22 – Curva ROC, <i>AUC</i> e <i>EER</i> na <i>Faces94</i> .	36
Figura 23 – Curva CMC na <i>Faces94</i> .	37
Figura 24 – Curva ROC, <i>AUC</i> e <i>EER</i> na LFW.	38
Figura 25 – Curva CMC na LFW.	38
Figura 26 – Tela Inicial.	40
Figura 27 – Tela de Informações.	40
Figura 28 – Tela de Cadastro Padrão.	40
Figura 29 – Tela de Cadastro Preenchida.	40
Figura 30 – Tela de Verificação Padrão.	41
Figura 31 – Tela de Verificação Preenchida.	41
Figura 32 – Tela de Identificação Padrão.	41
Figura 33 – Tela de Identificação Preenchida.	41
Figura 34 – Tela de Usuários.	42
Figura 35 – Tela de Entrada do Usuário.	42
Figura 36 – Registro do Usuário.	42

Figura 37 – Autenticação do Usuário.	43
Figura 38 – Diagrama do Armazenamento de Dados.	44
Figura 39 – <i>Cloud Firestore</i>	44
Figura 40 – <i>Cloud Storage</i>	45
Figura 41 – Curva ROC, <i>AUC</i> e <i>EER</i> no Aplicativo.	46
Figura 42 – Curva CMC no Aplicativo.	46

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
AUC	<i>Area Under Curve</i>
CMC	<i>Cumulative Match Characteristic</i>
CNN	<i>Convolutional Neural Network</i>
EER	<i>Equal Error Rate</i>
FN	<i>False Negative</i>
FNR	<i>False Negative Rate</i>
FP	<i>False Positive</i>
FPR	<i>False Positive Rate</i>
FRGC	<i>Face Recognition Grand Challenge</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IR	<i>Identification Rate</i>
LFW	<i>Labeled Faces in the Wild</i>
LN	<i>Layernorm</i>
MAS	<i>Multihead Self-Attention</i>
MLP	<i>Multilayer Perceptron</i>
MTCNN	<i>Multi-task Cascaded Convolutional Networks</i>
ROC	<i>Receiver Operating Characteristic</i>
TN	<i>True Negative</i>
TP	<i>True Positive</i>
TPR	<i>True Positive Rate</i>
ViT	<i>Vision Transformer</i>

Sumário

1	INTRODUÇÃO	12
1.1	Problema	12
1.2	Justificativa	13
1.3	Objetivos	13
1.3.1	Objetivo Geral	13
1.3.2	Objetos Específicos	13
1.4	Organização do Trabalho	14
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	Aprendizado de Máquina	15
2.1.1	Rede Neural Convolucional	15
2.1.2	<i>Vision Transformer</i>	16
2.2	Sistemas Biométricos	18
2.2.1	Pré-processamento Facial	19
2.2.2	Reconhecimento Facial	21
2.3	Métricas de Desempenho	22
2.3.1	Sistemas de Verificação	23
2.3.2	Sistemas de Identificação	24
2.4	Funções de Perda	25
2.4.1	<i>Additive Angular Margin Loss (ArcFace)</i>	25
2.5	Funções de Similaridade	26
2.5.1	Distância Euclidiana	26
2.5.2	Similaridade por Cosseno	27
3	MATERIAL E METODOLOGIA	28
3.1	Hardware	28
3.2	Bases de Dados	28
3.3	Metodologia	30
3.3.1	Pré-processamento Facial	30
3.3.2	Reconhecimento Facial	31
3.3.3	Métricas de Avaliação	32
4	RESULTADOS E DISCUSSÃO	33
4.1	Treinamento e Validação	33
4.2	Experimentos	34
4.2.1	Base FRGC	34

4.2.2	Base <i>Faces94</i>	36
4.2.3	Base LFW	37
5	SISTEMA BIOMÉTRICO	39
5.1	Aplicativo	39
5.2	Registro e Autenticação	42
5.2.1	Armazenamento de Dados	43
5.3	Resultados e Discussão	45
6	CONCLUSÃO	47
6.1	Trabalhos Futuros	47
	REFERÊNCIAS	48
	APÊNDICE A – PRÉ-PROCESSAMENTO FACIAL	51
	APÊNDICE B – RECONHECIMENTO FACIAL	53
	APÊNDICE C – CURVA ROC, AUC E EER	55
	APÊNDICE D – CURVA CMC	57

1 Introdução

Segundo Guo e Zhang (2019), o rosto é a característica mais comum usada pelos humanos para reconhecimento de outras pessoas, e o reconhecimento facial é um problema clássico e ainda muito estudado na área de visão computacional, sendo que o seu processo consiste basicamente de três etapas: (i) pré-processamento das imagens, ou seja, a etapa de normalização, detecção e alinhamento das faces; (ii) extração de características; e (iii) comparação de características, onde o indivíduo pode ser identificado através das características extraídas na etapa anterior.

O primeiro sistema para reconhecimento automático de faces foi proposto em 1973 (LI; JAIN, 2011), portanto, há 50 anos. Nos últimos anos, observou-se um rápido crescimento na demanda por este tipo de sistema, sendo que os métodos tradicionais de reconhecimento facial, como os baseados em *Eigenfaces* (TURK; PENTLAND, 1991) e *Local Binary Patterns* (AHONEN; HADID; PIETIKAINEN, 2006), dentre outros, conhecidos como métodos *handcrafted*, que são bastante sensíveis à vários fatores, como baixa resolução das imagens, variações de pose e mudanças de iluminação, passaram a serem substituídos por métodos baseados em aprendizado de máquina, mais especificamente pelas redes neurais artificiais, em especial as redes neurais convolucionais profundas, que apresentam maior capacidade de aprendizado, generalização e robustez (GUO; ZHANG, 2019).

Recentemente, técnicas baseadas em mecanismos de atenção surgiram na área de Aprendizado de Máquina. Estas técnicas têm sido incorporadas à área de Visão Computacional, sendo uma delas conhecida como *Vision Transformer* (DOSOVITSKIY et al., 2020), e têm sido utilizadas em várias aplicações (KHAN et al., 2022). Neste trabalho, objetivou-se analisar o desempenho de técnicas baseadas em mecanismos de atenção na tarefa de reconhecimento facial, como também analisar a viabilidade de se utilizar este tipo de método em aplicativos móveis para a identificação automática de pessoas por meio do reconhecimento biométrico facial.

1.1 Problema

Como verificado por Guo e Zhang (2019), com o desenvolvimento dos *hardwares* de computador e das tecnologias de imagem, a tarefa de reconhecimento facial passou a ser aplicada de forma constante, principalmente em atividades como controle de acesso e sistemas de vigilância. Entretanto, devido a variações como posicionamento, iluminação e qualidade das imagens coletadas, se faz necessário sempre buscar novas técnicas em prol de melhorar o desempenho, acurácia e portabilidade dos sistemas de reconhecimento.

Além disso, dado o constante crescimento no uso de dispositivos móveis e o aumento de informações relevantes e pessoais sobre os indivíduos nos aplicativos, considera-se necessário desenvolver sistemas de autenticação que garantam segurança e eficiência para seus usuários.

1.2 Justificativa

Considerando os problemas expostos, a atual pesquisa, através de técnicas de desenvolvimento *mobile* e aprendizagem de máquina, analisou a performance do modelo *Vision Transformer* na atividade de reconhecimento facial com o uso de um sistema móvel de identificação biométrica.

Por se tratar de uma tecnologia recente, proposta em 2020 (DOSOVITSKIY et al., 2020), estudos na área do *Vision Transformer* são importantes para ampliar os conhecimentos sobre o modelo, principalmente na atividade de reconhecimento biométrico, sendo possível também realizar comparativos entre outras arquiteturas já consolidadas.

Inclusive, a implementação de um modelo para reconhecimento facial permite adaptações para outros elementos de identificação, como íris e impressões digitais, através do uso de técnicas de transferência de aprendizado, ou seja, o reuso de modelos treinados para novas tarefas (AUNG et al., 2022).

1.3 Objetivos

1.3.1 Objetivo Geral

Avaliar o desempenho do modelo de classificação de imagens *Vision Transformer* na atividade de reconhecimento facial com o uso de um aplicativo móvel para a identificação biométrica de pessoas.

1.3.2 Objetos Específicos

- Obter bases de dados públicas contendo imagens de faces;
- Estudar métodos de processamento de imagens, reconhecimento facial e aprendizado de máquina;
- Aplicar técnicas de pré-processamento de imagens para realçar e detectar detalhes importantes;
- Desenvolver, treinar, validar e testar o modelo *Vision Transformer* para a tarefa de reconhecimento facial;
- Analisar os resultados obtidos e compará-los com o estado da arte;

- Criar uma aplicação móvel de um sistema de biometria facial utilizando as técnicas a serem estudadas e o modelo desenvolvido.

1.4 Organização do Trabalho

O presente trabalho foi organizado da seguinte maneira:

Capítulo 2. Apresenta os fundamentos sobre aprendizado de máquina e sistemas biométricos, além de conceitos sobre métricas de desempenho, funções de perda e funções de similaridade;

Capítulo 3. Através dos fundamentos e conceitos apresentados, este capítulo compreende as etapas de desenvolvimento e a metodologia aplicada para atingir os objetivos e resultados da pesquisa;

Capítulo 4. Evidencia os resultados obtidos através dos conceitos estudados e da metodologia aplicada, utilizando de métricas para avaliar o desempenho do modelo;

Capítulo 5. Demonstra o funcionamento e o fluxo de processos do sistema biométrico aplicado no aplicativo móvel;

Capítulo 6. Discorre sobre as conclusões sobre a pesquisa e indica trabalhos futuros a serem realizados na área da visão computacional com uso do *Vision Transformer*.

2 Fundamentação Teórica

Este capítulo busca conceituar e destacar os principais fundamentos teóricos utilizados na pesquisa, visando compreender as principais etapas do processo de aprendizado de máquina, o funcionamento de sistemas biométricos, como o pré-processamento e o reconhecimento, quais métricas e como são usadas para mensurar o desempenho de modelos de verificação e identificação e as funções de perda e similaridade mais comuns utilizadas para treinamento e inferência na tarefa de reconhecimento facial.

2.1 Aprendizado de Máquina

Segundo Li e Jain (2011), o problema do reconhecimento facial possui diversas limitações, como grande variação nos formatos, poses e iluminações dos rostos, a não-linearidade das características faciais, dessa forma, se fez necessário o desenvolvimento de novas técnicas que visam minimizar essas dificuldades.

Antes já utilizados, métodos como *Eigenfaces* (TURK; PENTLAND, 1991) e *Local Binary Patterns* (AHONEN; HADID; PIETIKAINEN, 2006), conhecidos como *handcrafted*, passaram a ser substituídos por técnicas baseadas em aprendizado de máquina, com destaque no aprendizado profundo, como redes neurais convolucionais, ou *Convolutional Neural Networks* (CNNs) (GUO; ZHANG, 2019), e transformadores de visão, ou *Vision Transformer* (ViT) (DOSOVITSKIY et al., 2020).

Uma rede neural artificial, de acordo com Guo e Zhang (2019), é um modelo computacional não-linear inspirado no sistema biológico no processamento de informações, consistindo em três principais camadas, a camada de entrada, as camadas ocultas, onde o processamento é realizado, e a camada de saída. Dessa forma, uma rede neural com várias camadas ocultas entre a entrada e saída é chamada de rede neural profunda, assim modelando relações mais complexas entre os dados.

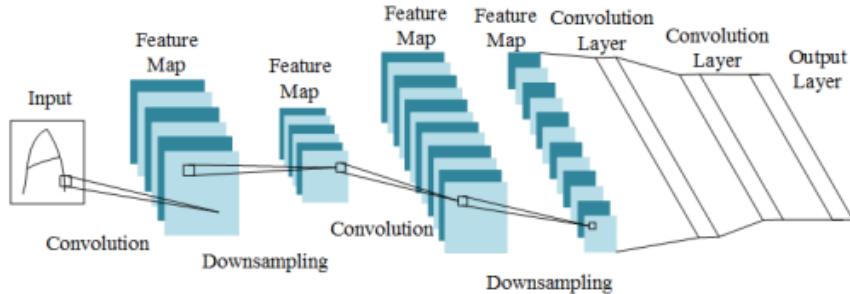
2.1.1 Rede Neural Convolucional

Em relação aos modelos de aprendizado profundo, destaca-se a CNN, a técnica mais popular na área da visão computacional, em tarefas como classificação de imagens, detecção de objetos e reconhecimento facial (GUO; ZHANG, 2019).

A estrutura de uma CNN, visível na Figura 1, é composta basicamente de, segundo Guo e Zhang (2019), camadas convolucionais, responsáveis por extrair as características dos dados de entrada, camadas de agrupamento, que visam reduzir as dimensões dos mapas de

características, e as camadas totalmente conectadas, onde o processo de classificação das informações das camadas anteriores é iniciado.

Figura 1 – Estrutura de uma CNN.



Fonte: (LI et al., 2020).

As camadas convolucionais consistem em um conjunto de filtros de aprendizado $W = W_1, W_2, \dots, W_k$ e viéses $B = b_1, b_2, \dots, b_k$, onde é aplicada o operação de convolução para gerar um mapa de características X_k , que passa por uma transformação não-linear $\sigma(\cdot)$, sendo esse processo repetido por cada camada convolucional t e definido pela seguinte equação:

$$X_k^t = \sigma(W_k^{t-1} * X^{t-1} + b_k^{t-1}) \quad (2.1)$$

2.1.2 Vision Transformer

Arquiteturas baseadas em autoatenção, como os transformadores, se tornaram técnicas importantes na área de processamento de linguagem natural, dessa forma, inspirados nesse modelo, Dosovitskiy et al. (2020) realizaram adaptações para uso direto em imagens.

Como é possível verificar na Figura 2, a imagem de entrada $x \in \mathbb{R}^{H \times W \times C}$ é transformada em uma sequência de partes $x_p \in \mathbb{R}^{N \times (P^2 \times C)}$, sendo (H, W) a resolução da imagem, C o número de canais, (P, P) o tamanho de cada parte e $N = HW/P^2$ a quantidade de partes da imagem.

A imagem, já separada em partes, é transformada em vetores de tamanho D através de uma projeção linear treinável, chamada de incorporações, ou *embeddings*. Para guardar as informações posicionais das partes das imagens, incorporações de posição são adicionadas as incorporações das partes da imagem, sendo também necessário uma sequência treinável $z_0^0 = x_{class}$, onde o estado de saída do codificador do transformador z_L^0 é a representação da imagem y .

Já a etapa de codificação consiste em alternadas camadas de autoatenção, ou *multihead dead self-attention* (MSA), blocos de perceptron multicamadas, ou *multilayer perceptron*

(MLP), camadas de normalização, ou *layernorm* (LN) e camadas de conexão residual, que podem ser definidas pelas seguintes equações:

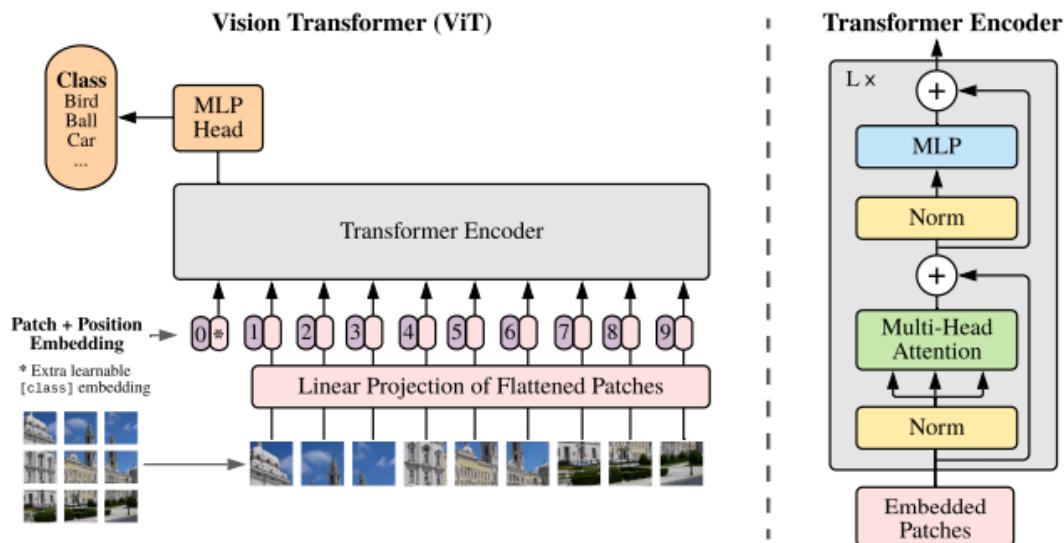
$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \dots; x_p^N E] + E_{pos} \quad E \in \Re^{(P^2 \times C) \times D}, E_{pos} \in \Re^{(N+1) \times D} \quad (2.2)$$

$$z_l' = MSA(LN(z_{l-1})) + z_{l-1} \quad l = 1 \dots L \quad (2.3)$$

$$z_l = MLP(LN(z_l')) + z_l' \quad l = 1 \dots L \quad (2.4)$$

$$y = LN(z_L^0) \quad (2.5)$$

Figura 2 – Estrutura de um ViT.

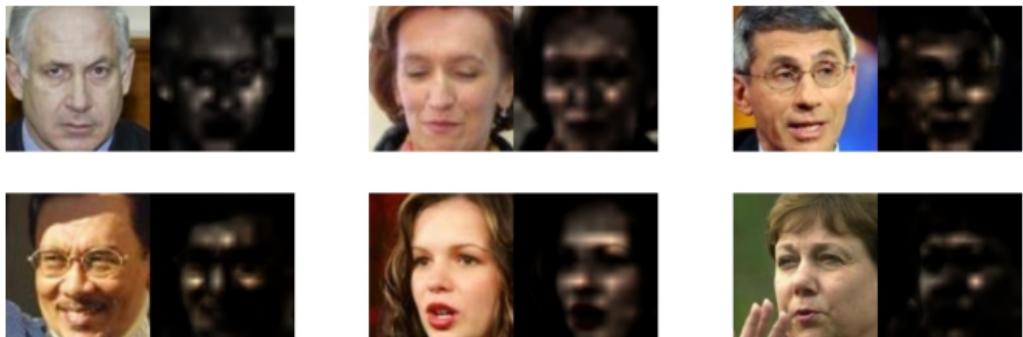


Fonte: (DOSOVITSKIY et al., 2020).

A técnica dos transformadores ainda permite que seja possível visualizar os pontos de atenção, apresentados na Figura 3, para facilitar a interpretação das decisões do modelo, o conjunto dos pontos de atenção é chamado de mapa de atenção, e conforme Abnar e Zuidema (2020), um dos métodos, conhecido como distribuição de atenção, ou *attention rollout*, realiza uma recursão multiplicando a matriz de atenção atual (A) com a distribuição de atenção da camada anterior (\tilde{A}).

$$\tilde{A}(l_i) = \begin{cases} A(l_i) \times \tilde{A}(l_{i-1}) & i > 0 \\ A(l_i) & i = 0 \end{cases} \quad (2.6)$$

Figura 3 – Exemplos de Mapas de Atenção.



Fonte: (ZHONG; DENG, 2021).

2.2 Sistemas Biométricos

A tarefa de reconhecimento facial é realizada pelos humanos de forma rotineira e através da disponibilidade de potência e baixo-custo de implementação hoje é possível automatizar essa atividade em uma variedade de aplicações, como autenticação biométrica, vigilância e interações humano-computador (LI; JAIN, 2011).

Conforme Li e Jain (2011), o uso da face como característica biométrica possui inúmeras vantagens em relação a outras modalidades, como impressão digital e íris, com destaque na baixa invasividade e ser capturada em diversas situações e distâncias distintas, ou seja, baixa restrição de extração. Outros fatores importantes que tornam o reconhecimento facial alvo de pesquisas são os rápidos avanços em dispositivos de captura de vídeo e imagem e a disponibilidade de grandes bases de dados, principalmente para estudos na área de aprendizado de máquina.

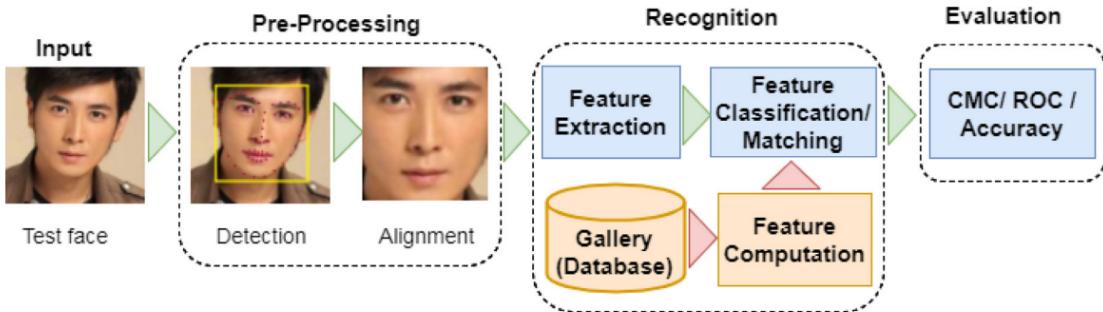
Apesar do progresso no desenvolvimento de novas técnicas de reconhecimento facial, a performance de um sistema pode ser afetada por diversos fatores como iluminação, posições e expressões faciais e enviesamento no reconhecimento, dessa maneira, baseando-se nesses fatores Li e Jain (2011) classificam um sistema de reconhecimento facial como cenários controlados, como entrada em contas de usuário, e cenários não controlados, como câmeras de vigilância.

Em relação ao modo de operação de um sistema biométrico, o reconhecimento facial, segundo Li e Jain (2011), pode operar tanto em modo de identificação, que envolve a comparação um-para-muitos, ou seja, um rosto de entrada será comparado com rostos já armazenados em uma base de dados, ou em modo de verificação, logo um-para-um, portanto, dois rostos de entrada serão comparados para identificar a similaridade entre eles.

Como é possível verificar na Figura 4, o processo de reconhecimento facial pode ser resumido em três principais etapas, a etapa de pré-processamento, onde o rosto será detectado e alinhado, e assim adaptado para o modelo de reconhecimento, a etapa de reconhecimento,

onde as características do rosto serão extraídas pelo modelo e comparadas, dependendo do modo de operação, para identificação de similaridade, e por fim, a etapa de avaliação, onde será verificado o desempenho do sistema.

Figura 4 – Processo de Pré-Processamento e Reconhecimento Facial.

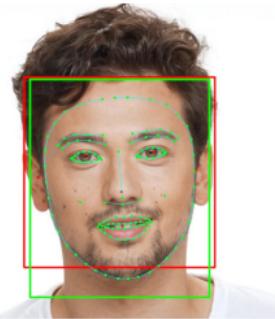


Fonte: (GUO; ZHANG, 2019).

2.2.1 Pré-processamento Facial

Conforme Minaee et al. (2021), os métodos utilizados para a detecção já passaram, devido as suas limitações, como posicionamento, expressões faciais, escala e iluminação, por grandes avanços, deixando de lado o uso de algoritmos como *Viola-Jones* (VIOLA; JONES, 2001) para modelos de redes neurais convolucionais como *Multi-task Cascaded Convolutional Networks* (MTCNN) (ZHANG et al., 2016) e *BlazeFace* (BAZAREVSKY et al., 2019), sendo esse destacado na Figura 5.

Figura 5 – Exemplo de Detecção com *BlazeFace*.



Fonte: (BAZAREVSKY et al., 2019).

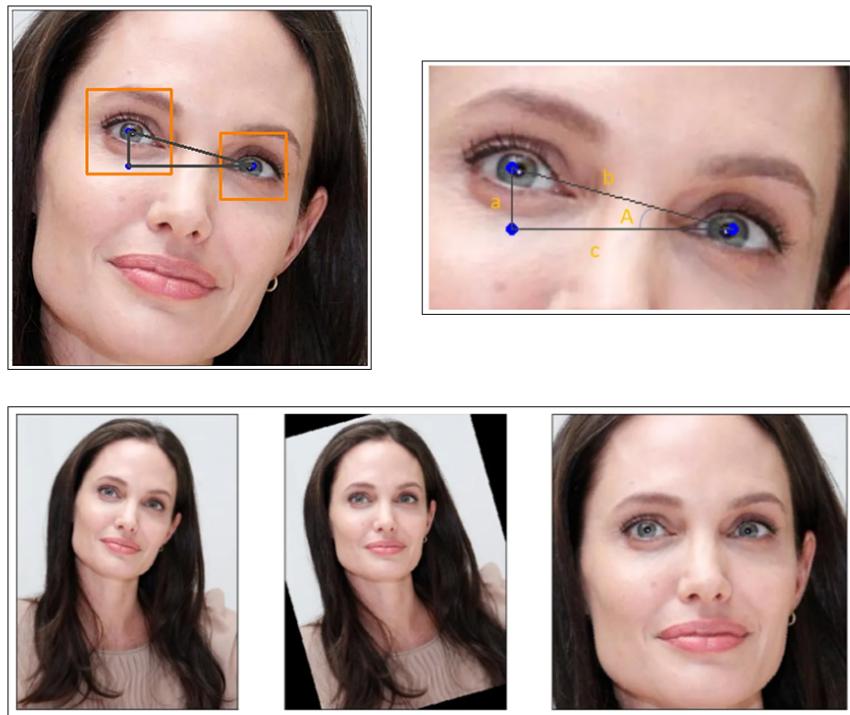
Nesta pesquisa é destacado o método *BlazeFace*, em especial devido a sua alta velocidade e implementação específica para dispositivos móveis, que consiste, segundo Bazarevsky et al. (2019), em três principais componentes:

- Extração de características: Responsável por extrair as representações de características das imagens de entrada;
- Esquema de âncora: Encarregado de gerar um conjunto de possíveis caixas de demarcação para os rostos detectados;
- Pós-processamento: Responsável por realizar a seleção da melhor caixa de demarcação para cada rosto identificado na imagem de entrada.

As técnicas de detecção facial geralmente não retornam apenas as demarcações dos rostos identificados, mas também pontos de referência a certos atributos como olhos, nariz, boca e as orelhas, e através desses pontos, em especial os olhos, é possível realizar o alinhamento da face.

Dessa forma, Serengil (2020) propõe uma técnica, exemplificada na Figura 6, utilizando de trigonometria para identificar a direção e o ângulo de rotação do rosto para um alinhamento conciso, baseada nos passos descritos abaixo:

Figura 6 – Processo de Alinhamento Facial.



Fonte: Adaptado de (SERENGIL, 2020).

1. Inicialmente, é identificado um terceiro ponto, que dependerá de qual olho está mais pra cima em relação ao outro, esse ponto intermediário também determinará a direção que a imagem será rotacionada;

2. Após a identificação do ponto intermediário e sua posição em relação aos olhos, será calculada a distância entre todos os pontos;
3. Com a distância entre todos os pontos e a direção de rotação definida, através da lei dos cossenos, conforme a equação 2.7, é possível identificar o ângulo de rotação da imagem e assim realizar o alinhamento.

$$\cos(A) = \frac{b^2 + c^2 - a^2}{2bc}$$

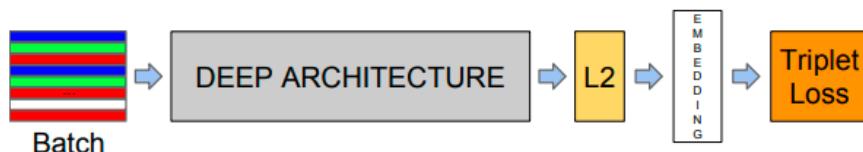
$$A = \arccos(\cos(A)) \quad (2.7)$$

2.2.2 Reconhecimento Facial

Devido às limitações encontradas nos algoritmos tradicionais, por exemplo *Eigenfaces* (TURK; PENTLAND, 1991) e *Local Binary Patterns* (AHONEN; HADID; PIETIKAINEN, 2006), como baixa resolução, variações de pose e iluminação, eles passaram a ser substituídos pelas redes neurais artificiais, que apresentam vantagens na habilidade de aprendizagem e generalização (GUO; ZHANG, 2019), com destaque em modelos como *VGGFace* (PARKHI; VEDALDI; ZISSERMAN, 2015) e *FaceNet* (SCHROFF; KALENICHENKO; PHILBIN, 2015), baseados em CNNs.

Por exemplo, o modelo *FaceNet*, visualizável na Figura 7 e descrito por Schroff, Kalenichenko e Philbin (2015), é composto por quatro principais partes: a camada de entrada em lote do conjunto de imagens que serão processadas; a rede neural profunda convolucional, responsável por extrair as características dos rostos nas imagens vindas da camada anterior; a camada de normalização, que reduz a escala das características, tornando a comparação mais eficaz; e por fim a função de perda *triplet loss*, que visa minimizar a distância entre imagens do mesmo indivíduo enquanto maximiza a distância entre pessoas distintas.

Figura 7 – Estrutura do Modelo *FaceNet*.



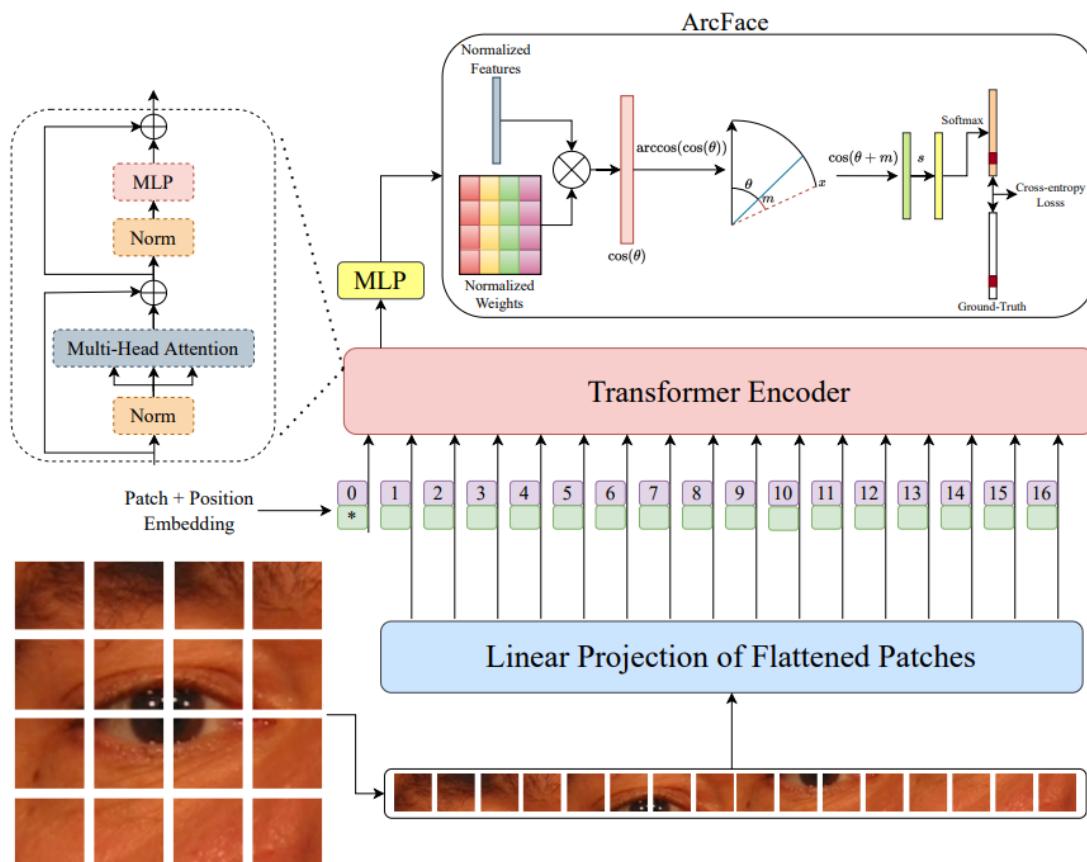
Fonte: (SCHROFF; KALENICHENKO; PHILBIN, 2015).

Apesar do sucesso do uso de CNNs para a tarefa de reconhecimento facial, atualmente novas alternativas surgiram, principalmente através de técnicas, descritas na Subseção 2.1.2, de atenção, como autoatenção, e uso dos transformadores, com enfoque no modelo ViT, especializado em visão computacional. Portanto, utilizando dessas novas técnicas de aprendizado

profundo, a pesquisa consiste em uma metodologia fundamentada em Manesco e Marana (2023), inicialmente aplicada para reconhecimento periocular e adaptada para reconhecimento facial.

Como é possível visualizar na Figura 8, a imagem do rosto passará pela arquitetura do ViT, responsável por extraír as características do rosto em um vetor de características, esse vetor passará pela função de perda *Additive Angular Margin Loss (ArcFace)* (DENG et al., 2019), muito utilizada na tarefa de reconhecimento, que irá aprimorar a classificação dos elementos e otimizar o treinamento do modelo.

Figura 8 – Processo de Reconhecimento Facial usando ViT.



Fonte: (MANESCO; MARANA, 2023).

2.3 Métricas de Desempenho

Uma parte essencial no desenvolvimento de um sistema biométrico, utilizando tanto o método de identificação quanto de verificação, é mensurar seu desempenho, e conforme El-Abed, Charrier e Rosenberger (2012), pode ser categorizado em três principais características:

- Qualidade dos dados: A qualidade e variações das imagens, como distância e iluminação, utilizadas no processo de reconhecimento é um dos principais fatores que afetam a

performance de um sistema biométrico;

- Usabilidade: Um sistema biométrico deve atingir seu objetivo de forma eficaz e eficiente, visando uma melhor experiência ao usuário;
- Segurança: A segurança é primordial no processo de reconhecimento, minimizando as vulnerabilidades e brechas sem prejudicar a usabilidade do modelo.

2.3.1 Sistemas de Verificação

Acerca do método de verificação, Li et al. (2020) comentam a importância de algumas taxas para medir o desempenho do sistema, sendo elas a taxa de falsos positivos, ou *false positive rate* (*FPR*), taxa de verdadeiros positivos, ou *true positive rate* (*TPR*) e a taxa de falsos negativos, ou *false negative rate* (*FNR*), sendo *true positive* (*TP*) pares positivos que foram considerados positivos, *false negative* (*FN*) pares positivos que foram considerados negativos, *true negative* (*TN*) pares negativos que foram considerados negativos e *false positive* (*FP*) pares negativos que foram considerados positivos.

$$TPR = TP/(TP + FN) \quad (2.8)$$

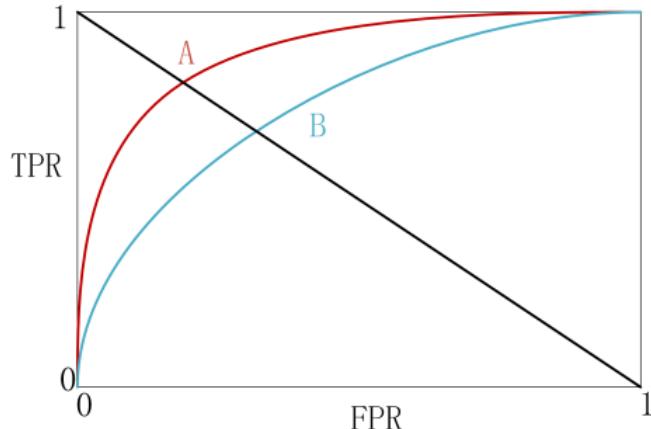
$$FPR = FP/(FP + TN) \quad (2.9)$$

$$FNR = 1 - TPR \quad (2.10)$$

Através dessas taxas, é possível calcular a curva característica de operação do receptor, ou *receiver operating characteristic* (ROC), definida pelas taxas *TPR* e *FPR*, onde quanto mais próxima do canto superior esquerdo, melhor o desempenho do sistema.

Um exemplo de curva ROC pode ser visualizada na Figura 9, onde são apresentadas duas curvas que representam dois modelos distintos A e B e uma reta diagonal que representa um algoritmo aleatório, onde o primeiro modelo é o que mais se aproxima do canto superior esquerdo, dessa forma demonstrando um desempenho superior ao segundo.

Figura 9 – Exemplos de Curvas ROC.



Fonte: (Li et al., 2020).

Outras métricas importantes podem ser destacadas, como, conforme Li et al. (2020) e El-Abed, Charrier e Rosenberger (2012), a área abaixo da curva, ou *area under curve (AUC)*, um valor numérico que representa a curva ROC, e proporcional ao desempenho do sistema, e também a taxa de erro igual, ou *equal error rate (EER)*, que corresponde ao ponto onde a *FPR* é igual a *FNR*, normalmente utilizada como limiar de corte para considerar um exemplar válido ou impostor.

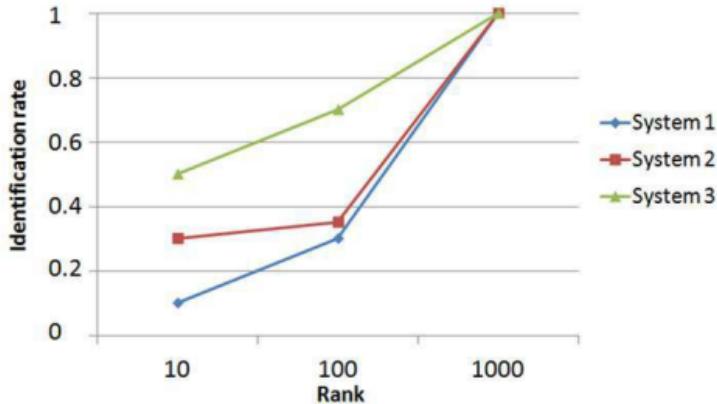
2.3.2 Sistemas de Identificação

Em relação a identificação, El-Abed, Charrier e Rosenberger (2012) comentam sobre a taxa de identificação, ou *identification rate (IR)*, que consiste na proporção de usuários que foram corretamente identificados dentro de uma lista com t indivíduos, ou seja, se o usuário desejado está presente entre os t mais similares.

O cálculo da *IR* para diversos valores t resulta na curva característica de correspondência cumulativa, ou *cumulative match characteristic (CMC)*, com destaque para $t = 1$, métrica comum para definir o desempenho de um sistema de identificação.

Na Figura 10 é possível verificar um exemplo de curva CMC, onde são demonstradas três curvas que representam três sistemas de identificação individuais. É possível concluir pelos resultados que o terceiro sistema apresenta o melhor desempenho, possuindo a maior taxa *rank-1* entre os modelos presentes.

Figura 10 – Exemplos de Curvas CMC.



Fonte: (EL-ABED; CHARRIER; ROSENBERGER, 2012).

2.4 Funções de Perda

As funções de perda, de acordo com Wang et al. (2020), representam um importante papel na aprendizagem de características discriminativas, sendo utilizadas para mensurar a diferença entre a predição obtida com o esperado e assim guiar o treinamento do modelo de redes neurais.

2.4.1 Additive Angular Margin Loss (*ArcFace*)

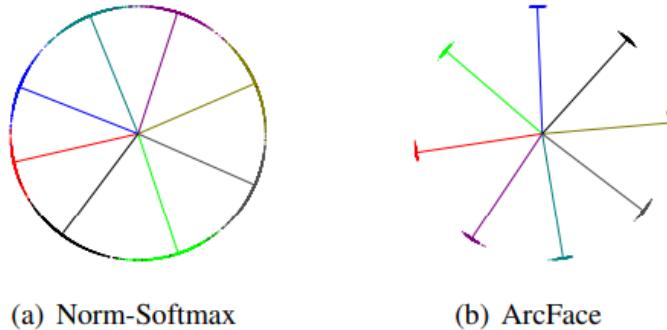
Segundo Deng et al. (2019), o *ArcFace* foi proposto devido as limitações encontradas em técnicas como *Softmax*, que não é suficientemente discriminativo para o problema de reconhecimento facial, e *Triplet Loss*, onde um conjunto de dados em larga escala pode ocasionar um aumento no número de passos de iteração.

A função de perda *ArcFace* pode ser definida pela equação 2.11, onde os elementos são representados em uma hiperesfera de raio s , e exemplares similares estarão localizados em uma distância angular de intervalo m (MANESCO; MARANA, 2023).

$$L = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s(\cos(\theta_{y_i} + m))}}{e^{s(\cos(\theta_{y_i} + m))} + \sum_{j=1, j \neq y_i}^n e^{s \cos(\theta_j)}} \quad (2.11)$$

Dessa forma, como pode-se observar na Figura 11, a função busca minimizar a distância angular entre rostos pertencentes a mesma pessoa, enquanto maximiza a distância angular entre elementos de classes distintas.

Figura 11 – Comparação entre *Softmax* e *ArcFace*.

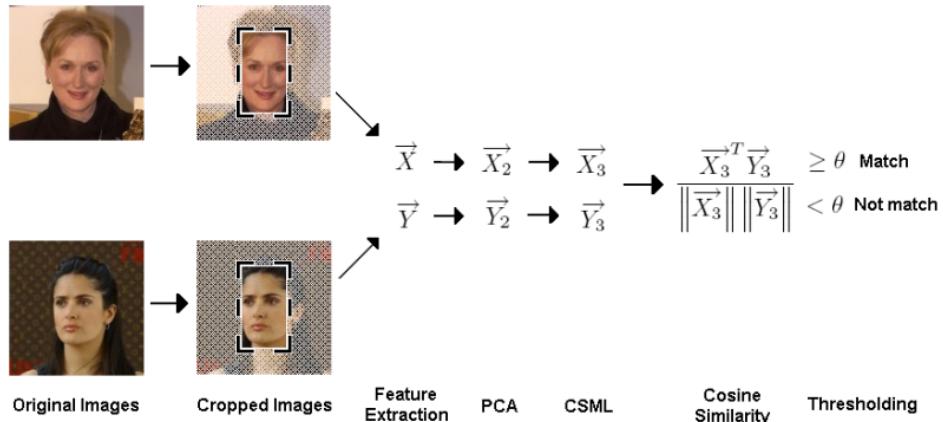


Fonte: (DENG et al., 2019).

2.5 Funções de Similaridade

As funções de similaridade têm como objetivo mensurar quão semelhantes são dois rostos. Como é possível verificar na Figura 12, o processo consiste em extrair as características faciais de duas imagens e através dos resultados obtidos no uso de tais funções em comparação a um limiar de corte, verificar se as faces pertencem ou não a mesma pessoa.

Figura 12 – Processo de Verificação Facial.



Fonte: (NGUYEN; BAI, 2010).

2.5.1 Distância Euclidiana

Considerando dois vetores $x = \{x_1, x_2, \dots, x_n\}$ e $y = \{y_1, y_2, \dots, y_n\}$, a distância euclidiana $ED(x, y)$ é dada por:

$$ED(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} \quad (2.12)$$

Na área de reconhecimento facial, conforme Wu et al. (2021), uma menor distância euclidiana implica em faces mais similares, podendo ser utilizada junto a um limiar de corte para indicar sucesso ou falha no reconhecimento.

2.5.2 Similaridade por Cosseno

Segundo Nguyen e Bai (2010), a similaridade por cosseno se apresenta como uma eficaz alternativa a distância euclidiana, contribuindo com um aumento na habilidade de generalização em relação as métricas já existentes.

A similaridade por cosseno $CS(x, y)$ entre dois vetores $x = \{x_1, x_2, \dots, x_n\}$ e $y = \{y_1, y_2, \dots, y_n\}$ possui valor resultante no intervalo $[-1, 1]$, sendo -1 para vetores opostos, 0 para vetores ortogonais e 1 para vetores na mesma direção, ou semelhantes, e pode ser definida através da seguinte fórmula:

$$CS(x, y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (2.13)$$

3 Material e Metodologia

Este capítulo apresenta os recursos computacionais e os dados utilizados neste trabalho, bem como a metodologia utilizada na pesquisa, ou seja, o processo realizado durante as etapas de obtenção das bases de dados, o pré-processamento das imagens através do *BlazeFace*, assim dizendo, a detecção e o alinhamento das faces, a fase de reconhecimento facial, do treinamento à validação, para realizar a extração das características utilizando do modelo ViT, e por fim a etapa de avaliação do modelo, através de métricas como a curva ROC, o valor *AUC* e a taxa *EER*.

3.1 Hardware

Em relação aos recursos computacionais utilizados nos experimentos, para a etapa de pré-processamento das imagens foi utilizado um computador com um processador *Intel(R) Core(TM) i5-4440* em conjunto com uma placa de vídeo *NVIDIA GeForce GTX 1050*. Já para a etapa de reconhecimento facial, devido ao desempenho necessário para o treinamento e validação do modelo ViT, considerou-se o uso de uma plataforma de desenvolvimento em nuvem, sendo escolhido o *Google Colab*, em seu plano *Colab Pro*, que dispõe de placas gráficas e unidades de processamento específicas para inteligência artificial.

3.2 Bases de Dados

Bases de dados de imagens são essenciais para o treinamento e a validação de um modelo de reconhecimento facial, e a qualidade e quantidade da base é um fator chave para garantir um bom desempenho e eficiência na realização da tarefa.

Para a realização da pesquisa, foram utilizadas três bases distintas, sendo a base de dados *Face Recognition Grand Challenge* (FRGC) (PHILLIPS et al., 2006) para treinamento e validação, e as bases *Labeled Faces in the Wild* (LFW) (HUANG et al., 2007) e *Faces94* (SPACEK, 2009) para validação e avaliação da capacidade de generalização do modelo.

A FRGC, demonstrada na Figura 13, consiste em aproximadamente 30 mil imagens capturadas entre os anos de 2002 e 2004 na Universidade de Notre Dame, em ambientes controlados e não controlados, e com variações de posição, expressão, distância e iluminação (PHILLIPS et al., 2006).

Figura 13 – Exemplos de Imagens da FRGC.



Fonte: Adaptada de (PHILLIPS et al., 2006).

Em relação a LFW, visualizável na Figura 14, é considerada uma das bases mais estudadas quando trata-se de um ambiente não controlado, sendo composta por aproximadamente 13 mil imagens de personalidades em diversas posições, iluminações e ambientes (HUANG et al., 2007).

Figura 14 – Exemplos de Imagens da LFW.



Fonte: Adaptada de (HUANG et al., 2007).

Já a base de dados *Faces94*, como é possível ver na Figura 15, consiste em aproximadamente três mil imagens capturadas na mesma distância, em um ambiente controlado, e durante uma conversa entre os modelos, buscando registrar expressões faciais diversas (SPACEK, 2009).

Figura 15 – Exemplos de Imagens da *Faces94*.

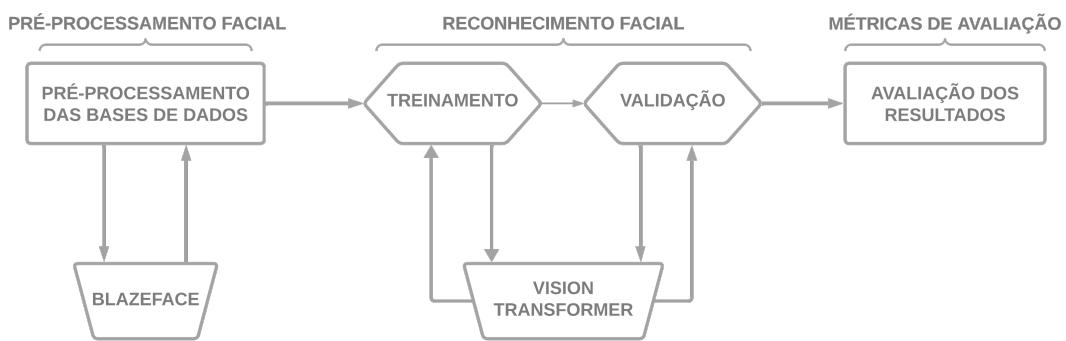


Fonte: Adaptada de (SPACEK, 2009).

3.3 Metodologia

A Figura 16 mostra um fluxograma da metodologia utilizada neste trabalho, fundamentada basicamente em três principais etapas: pré-processamento facial, onde as bases de dados são detectadas e alinhadas, através do modelo de detecção *BlazeFace*; a etapa de reconhecimento facial, onde é realizado o treinamento e a validação do modelo ViT; e por fim a de avaliação, onde serão utilizadas as métricas comuns a sistemas biométricos para avaliar o desempenho do modelo na tarefa de reconhecimento.

Figura 16 – Fluxograma da Metodologia.



Fonte: Elaborada pelo autor.

3.3.1 Pré-processamento Facial

Para realizar o pré-processamento das bases de dados, foi utilizado um modelo pronto do BlazeFace disponibilizado pela biblioteca mediapipe, que permite uma prática e intuitiva implementação, e as bibliotecas `math`, `numpy` e `PIL` para realizar os cálculos trigonométricos e as manipulações necessárias, como rotação, nas imagens.

A implementação da etapa de pré-processamento facial, exemplificada na Figura 17 e descrita no Apêndice A, utiliza a função `detector.detect()`, que retorna tanto a caixa demarcadora da face detectada quanto os pontos de referência do rosto, que são utilizados na etapa de alinhamento. Após a detecção, os pontos de referência dos olhos são passados para uma função de alinhamento, que utiliza de trigonometria, demonstrada na Subseção 2.2.1, para identificar o ângulo necessário para rotacionar a imagem corretamente, consequentemente, com a imagem alinhada, é possível realizar uma nova detecção para extrair o recorte correto do rosto.

Figura 17 – Exemplo de Pré-processamento usando BlazeFace.



Fonte: Elaborada pelo autor.

3.3.2 Reconhecimento Facial

Devido à necessidade de uma grande base de dados para alcançar bons resultados com o modelo estudado (DOSOVITSKIY et al., 2020), optou-se pelo uso de um modelo pré-treinado oferecido pela biblioteca `timm`, especializada em modelos no estado da arte de aprendizado profundo, com um tamanho fixo de imagem de 224×224 pixels que divide a imagem em pedaços de 16×16 pixels.

A base de dados de imagens já detectadas passou por um processo de pós-processamento antes de ser consumida pelo modelo, sendo aplicadas transformações, como rotação horizontal, alterações de cor, contraste e saturação e remoção de partes da imagem, aleatórias para aumentar o conjunto de dados, além de normalização baseada na média e no desvio padrão de cores da base de treinamento FRGC (STEINER et al., 2021), como é possível visualizar no processo A da Figura 18.

Além dos já pré-definidos pelo modelo pré-treinado, considerou-se também parâmetros como tamanho de lote, que define o processamento paralelo de imagens, de tamanho 32, e a dimensão do vetor de características, utilizado como entrada para o número de classes do modelo, de tamanho 1024, indicado por Manesco e Marana (2023) e demonstrado no processo B da Figura 18.

Figura 18 – Exemplo de Reconhecimento Facial usando ViT.



Fonte: Elaborada pelo autor.

A função de perda utilizada foi a função *ArcFace*, da biblioteca `pytorch_metric_learning`, com parâmetros de margem angular e escala de característica definidas em 0,5 radianos e 64, respectivamente (DENG et al., 2019). Em relação ao algoritmo de otimização, foi utilizado o *Lamb*, fornecido também pela biblioteca `timm`, com uma taxa de aprendizado de 0,001 e regularização de 0,02 (MANESCO; MARANA, 2023).

E por fim, para as rotinas de treinamento e validação, cujos códigos podem ser visualizados no Apêndice B, foi utilizado um número inicial de 50 épocas, porém foi verificado sobreajuste a partir da época 26, onde o treinamento foi encerrado.

3.3.3 Métricas de Avaliação

Para avaliar o desempenho do modelo e do sistema biométrico desenvolvido, foram utilizadas as métricas obtidas através da curva ROC, como a *AUC* e a *EER*, e as métricas obtidas através da curva CMC, de forma a mensurar o desempenho do modelo tanto em sistemas de verificação quanto em sistemas de identificação, respectivamente.

Para construir a curva e encontrar as métricas necessárias para avaliação do modelo, foi utilizada a técnica dos *scores* impostores e genuínos, utilizada por Manesco e Marana (2023), dessa forma, é calculada a similaridade por cosseno para cada par de imagens da base de dados, se forem da mesma pessoa, são considerados um par genuíno, do contrário, um par impostor. Através da lista de distâncias entre pares genuínos e impostores e como é possível verificar no Apêndice C, a biblioteca `scikit-learn` foi utilizada para encontrar as taxas *FPR* e *TPR* com uso da função `roc_curve()`, a métrica *AUC* foi calculada utilizando a função `auc()` e a taxa *EER* foi encontrada através da biblioteca `numpy`.

Já em relação as métricas baseadas na curva CMC, elas foram calculadas com base em uma face selecionada aleatoriamente de cada indivíduo e comparada com todas as imagens da base de dados, dessa forma é possível gerar um *rank-t* de similaridade para cada rosto, utilizando da similaridade por cosseno, e calcular as taxas *IR*, caso a imagem seja corretamente identificada dentro das *t* pessoas retornadas, necessárias para formulação da curva, como demonstrado no Apêndice D.

4 Resultados e Discussão

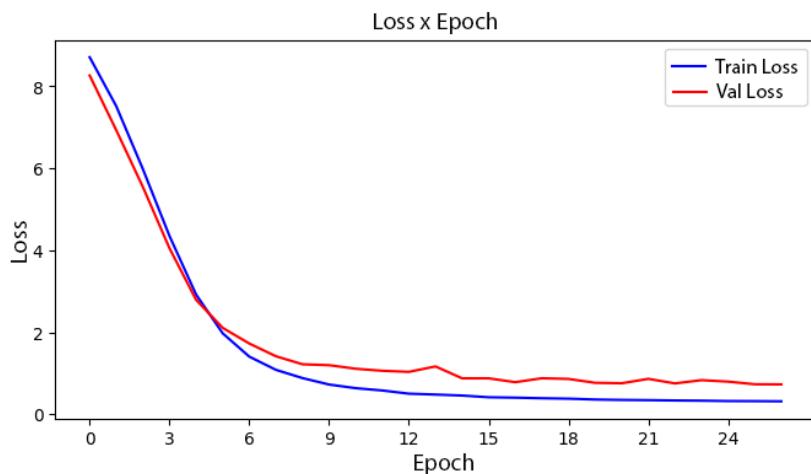
O atual capítulo busca discutir os resultados obtidos durante o processo de reconhecimento facial, incluindo as etapas de treinamento, validação e teste para as bases de dados utilizadas durante a pesquisa. Para a análise dos resultados visou-se, além da análise de tempo de treinamento por época, velocidade de inferência e curvas de perda de treinamento e validação, o uso de métricas como a curva ROC, valor *AUC* e taxa *EER* para sistemas de verificação e curva CMC e taxa *rank-1* para identificação.

4.1 Treinamento e Validação

Utilizando dos parâmetros citados na Subseção 3.3.2 o treinamento e validação do modelo foram realizados utilizando a plataforma *Google Colab* em sua versão *Pro*, devido a necessidade de uma eficiente placa gráfica e rápido processamento, com um tempo aproximado de 5 minutos por época no treinamento e menos de 1 minuto para a validação da base de dados da FRGC.

Em relação a perda, pode-se verificar pelas curvas apresentadas na Figura 19, onde a perda representa o eixo vertical enquanto a época de treinamento ou validação o eixo horizontal, que o treinamento e a validação se mantiveram consistentes até ocorrer um sobreajuste, onde o treinamento foi encerrado após 26 épocas, apresentando um valor de perda de aproximadamente 0,31 e 0,72 para treinamento e validação, respectivamente, sendo a curva azul representando a perda de treinamento enquanto a vermelha a de validação.

Figura 19 – Curva de Perda no Treinamento.



Fonte: Elaborada pelo autor.

No que diz respeito ao sobreajuste apresentado anteriormente, pode-se destacar principalmente como causa o tamanho limitado do conjunto de treinamento, escolhido devido ao equipamento disponível para os experimentos, que por não apresentar amostras de características suficientes para aperfeiçoar a extração do modelo, acaba por reduzir o desempenho na tarefa de reconhecimento facial (STEINER et al., 2021).

4.2 Experimentos

Retomando os conceitos apresentados na Seção 2.3, as métricas de desempenho foram aplicadas em cada base de dados em experimentos individuais e obtiveram os resultados destacados na Tabela 1, sendo verificado que a base FRGC foi a que obteve um melhor desempenho, enquanto a LFW, em especial devido a se tratar de uma base gerada em ambientes não controlados (HUANG et al., 2007), obteve o pior desempenho.

Tabela 1 – Resultados dos Experimentos.

Base de Dados	AUC	EER	Identificação Rank-1
FRGC	99,14%	2,36%	90,68%
Faces94	95,46%	12,05%	67,14%
LFW	75,08%	32,34%	1,74%

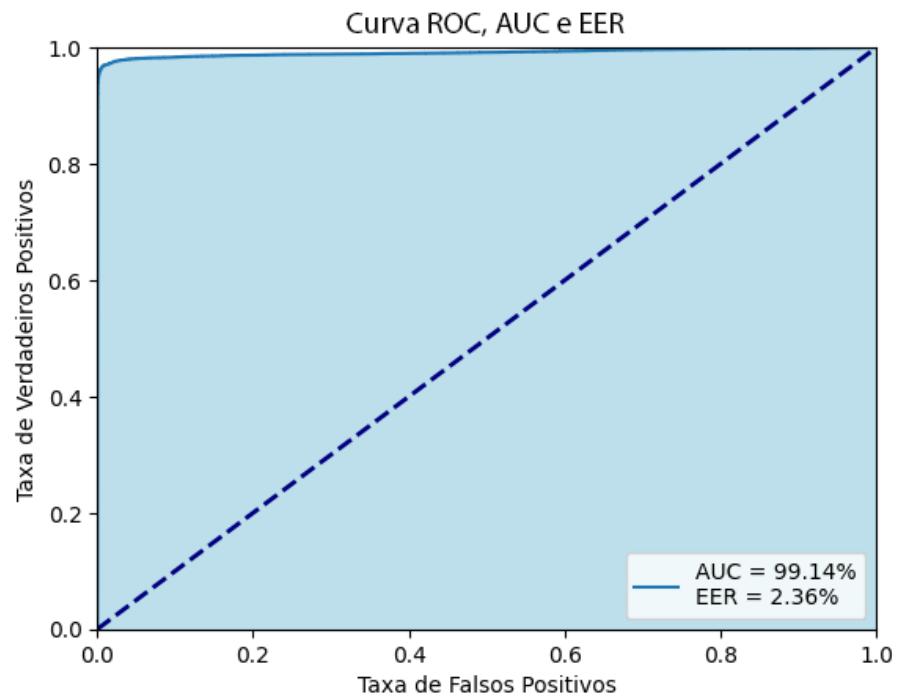
Fonte: Elaborado pelo autor.

4.2.1 Base FRGC

Em relação as métricas para sistema de verificação, apresentadas na Figura 20, ou seja, a curva ROC, o valor *AUC* e a taxa *EER*, aplicadas na base de imagens da FRGC, é possível verificar um excelente resultado, com uma *AUC* de 99,1%, representada pela zona azul preenchida abaixo da curva, e uma *EER* de 2,3%, demonstrando uma alta eficiência na classificação, porém ainda assim enviesada devido ao fato de ser a base utilizada para treinamento.

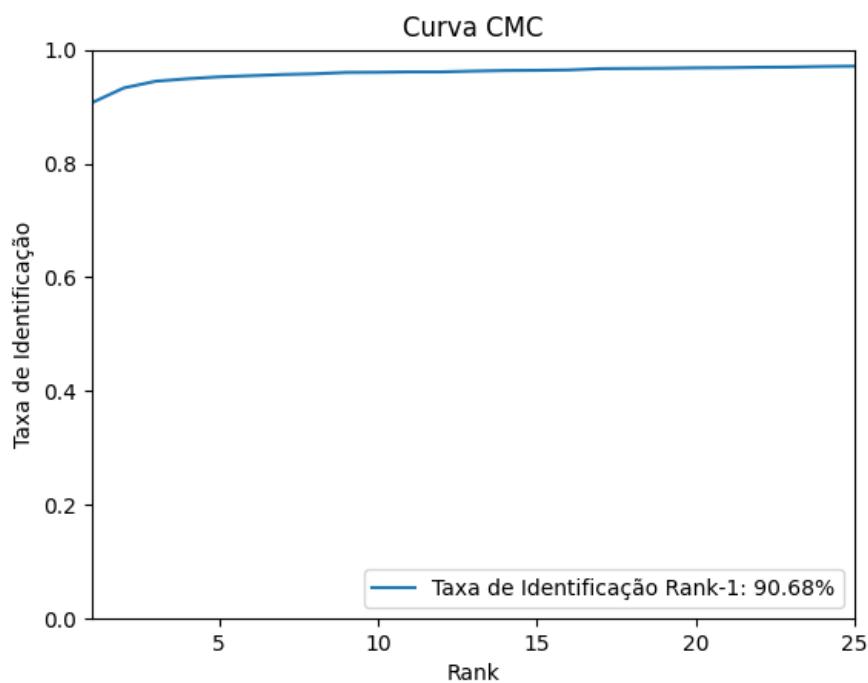
Já considerando o modo de identificação, a curva CMC, demonstrada na Figura 21, apresenta uma alta taxa de identificação para $t = 1$, demonstrando que o modelo é capaz de identificar o indivíduo com maior similaridade com base nos registros armazenados em aproximadamente 90,6% dos casos.

Figura 20 – Curva ROC, *AUC* e *EER* na FRGC.



Fonte: Elaborada pelo autor.

Figura 21 – Curva CMC na FRGC.



Fonte: Elaborada pelo autor.

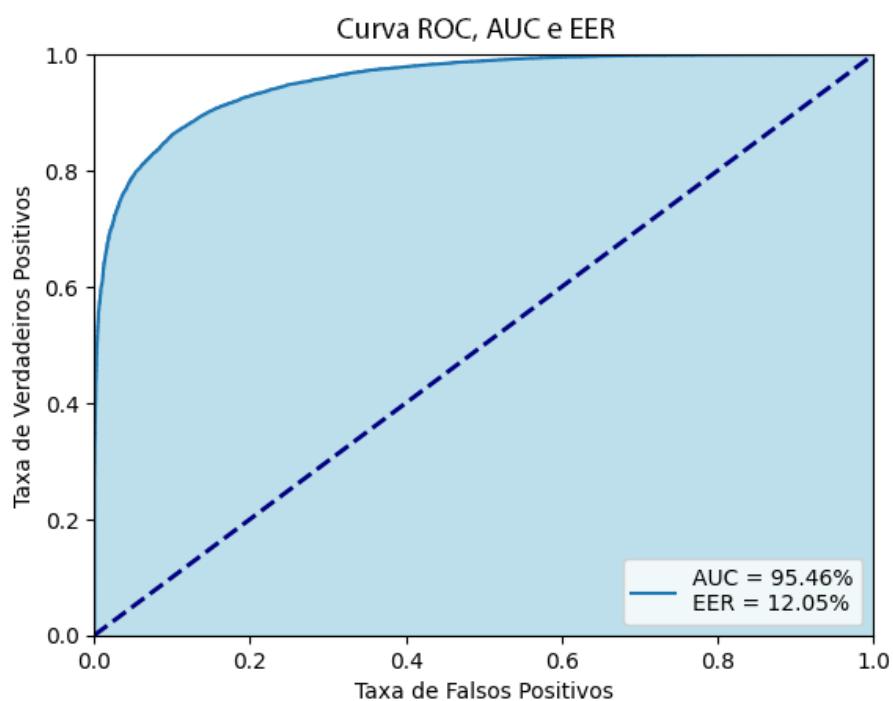
4.2.2 Base *Faces94*

A partir da *Faces94*, devido a semelhança nas características de construção com a base de dados da FRGC, como distância, iluminação e expressões faciais, foi possível obter também um excelente desempenho, resultando em um valor de *AUC* e de taxa *EER* em torno de 95,4% e 12%, respectivamente, como é possível ver na Figura 22.

No que diz respeito a curva CMC, apresentada na Figura 23, é possível verificar que, por mais que seja um resultado inferior em comparação a base anterior, ainda assim foi obtido um resultado superior aos 50% para *rank-1*, constando um valor próximo aos 67,1%, com uma taxa de acertos que aumenta de forma contínua a medida que o valor de *t* aumenta.

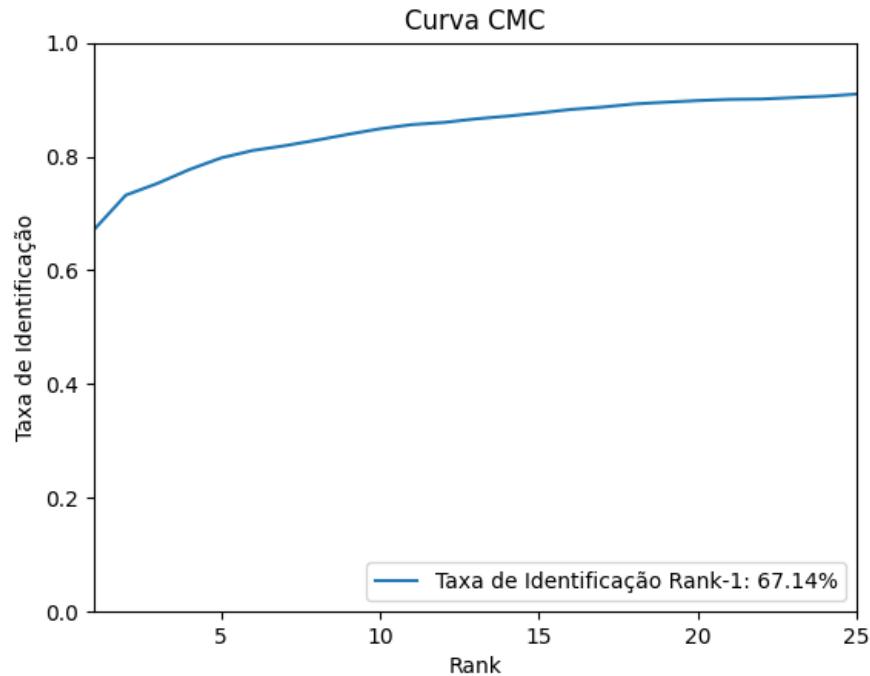
Com base nessas informações, é possível verificar uma redução na capacidade de generalização do modelo, que, por não se tratar da base de dados de treinamento, representou uma diminuição em todos os resultados obtidos, com destaque no valor de *rank-1* da curva CMC, que obteve a maior diferença.

Figura 22 – Curva ROC, *AUC* e *EER* na *Faces94*.



Fonte: Elaborada pelo autor.

Figura 23 – Curva CMC na *Faces94*.



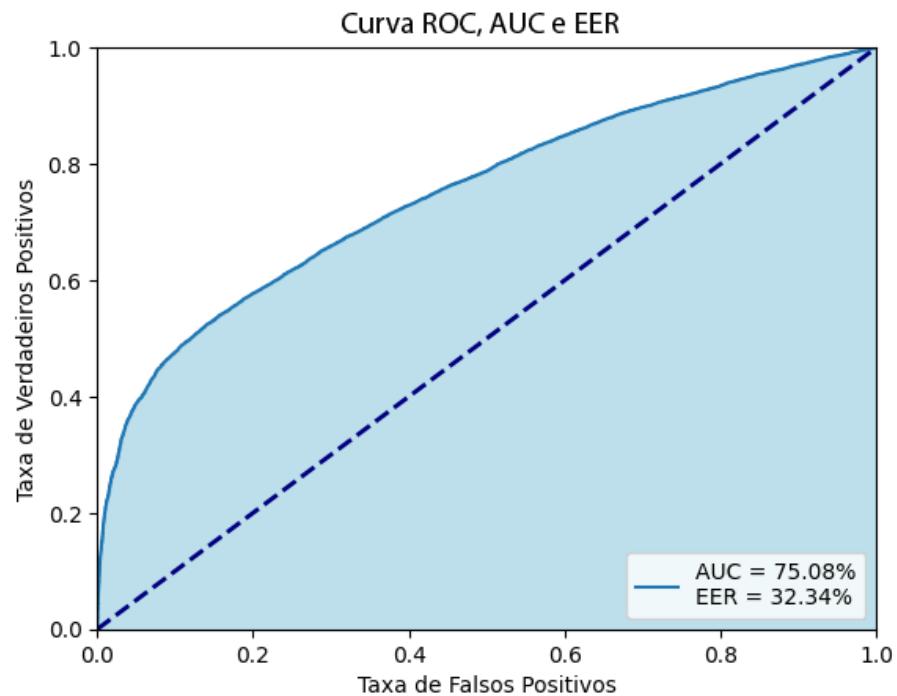
Fonte: Elaborada pelo autor.

4.2.3 Base LFW

E por fim, utilizando a base de dados LFW, foi verificado o pior desempenho, obtendo-se uma *AUC* de aproximadamente 75%, e uma taxa *EER* próxima a 32,3%, como é possível verificar na Figura 24. Todavia, um dos principais motivos a se destacar a partir desses resultados é que o modelo é sensível a ambientes não controlados, diferentemente das outras duas bases avaliadas.

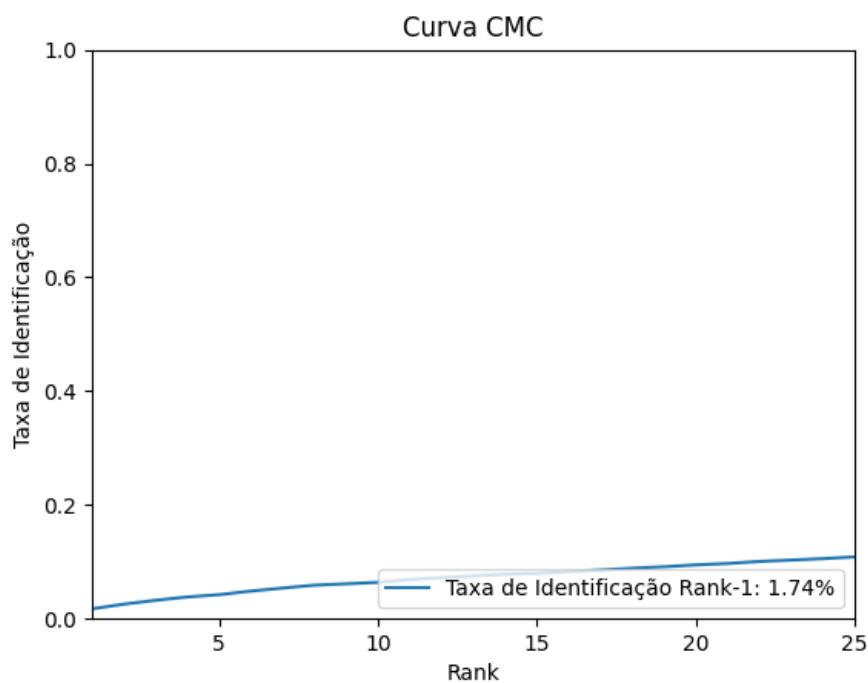
Acerca dos resultados obtidos no sistema de identificação, a base LFW também obteve o pior desempenho, tendo um valor *rank-1* perto de 1,7%, e obtendo um resultado próximo a 10% apenas quando *t* tende a 25, visualizável na Figura 25. Ao contrário das outras duas bases avaliadas, a LFW obteve um resultado muito inferior no sistema de identificação em relação ao de verificação, e conforme Decann e Ross (2012), isso se dá pelo fato de que uma curva ROC aceitável não implica necessariamente em uma boa curva CMC e vice-versa, demonstrando a importância do uso de diversas métricas concomitantemente na avaliação de um sistema biométrico.

Figura 24 – Curva ROC, AUC e EER na LFW.



Fonte: Elaborada pelo autor.

Figura 25 – Curva CMC na LFW.



Fonte: Elaborada pelo autor.

5 Sistema Biométrico

O presente capítulo trata sobre o sistema biométrico criado para avaliar o modelo em um ambiente real, para isso foi necessário o desenvolvimento de um aplicativo utilizando a linguagem de programação *Dart*¹ e sua *framework Flutter*², conectado a uma *Application Programming Interface* (API) modelada em *Python*³ desenvolvida com base na *framework Flask*⁴, em conjunto também com o banco de dados não relacional baseado em documentos *Cloud Firestore* e o banco de imagens *Cloud Storage*, oferecidos pela *Google Firebase*⁵.

5.1 Aplicativo

O aplicativo conta com uma tela inicial, visto na Figura 26, com quatro botões que direcionam para outras telas do sistema, sendo eles: para a tela de entrada do usuário, Figura 30, para uma tela com a lista de todos os usuários cadastrados no sistema, Figura 34, uma para informar sobre o funcionamento do aplicativo, na Figura 27, e por fim uma tela para realizar o cadastro dos usuários, como é possível ver na Figura 28.

A tela de cadastro consiste em diversos campos que serão armazenados na base de dados, destacando-se o nome de usuário, que será utilizado posteriormente na tela de verificação, para designar o usuário que deve ser comparado, e a imagem de cadastro, que será a representação das características do usuário no sistema.

Em relação à tela de entrada do usuário, é possível acessar os dois modos de um sistema de reconhecimento, o de verificação, e a tela de identificação, visualizável na Figura 32, através do botão no canto inferior direito. Na tela de verificação, o usuário dará o nome de usuário e uma foto, que será comparada diretamente com o usuário cadastrado, enquanto na tela de identificação a foto será comparada com todas os registros armazenados, retornando a pessoa com maior similaridade.

Caso o usuário seja validado na tela de verificação, ele será direcionado para uma tela de entrada, com as informações já armazenadas no banco de dados durante a etapa de cadastro, como é possível verificar na Figura 35. Todavia, se a tela de identificação for utilizada, o usuário será direcionado para a tela de entrada do usuário com maior similaridade encontrada em relação a todos os registros da base de dados.

¹ <https://dart.dev/>

² <https://flutter.dev/>

³ <https://python.org/>

⁴ <https://flask.palletsprojects.com/>

⁵ <https://firebase.google.com/>

Figura 26 – Tela Inicial.



Figura 27 – Tela de Informações.



Fonte: Elaboradas pelo autor.

Figura 28 – Tela de Cadastro Padrão.

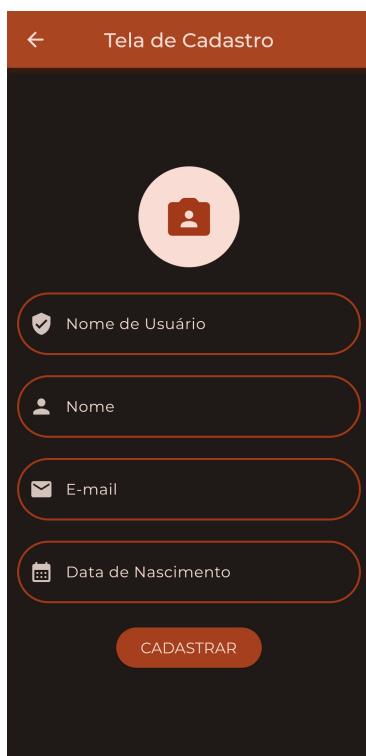


Figura 29 – Tela de Cadastro Preenchida.



Fonte: Elaboradas pelo autor.

Figura 30 – Tela de Verificação Padrão.



Figura 31 – Tela de Verificação Preenchida.



Fonte: Elaboradas pelo autor.

Figura 32 – Tela de Identificação Padrão.

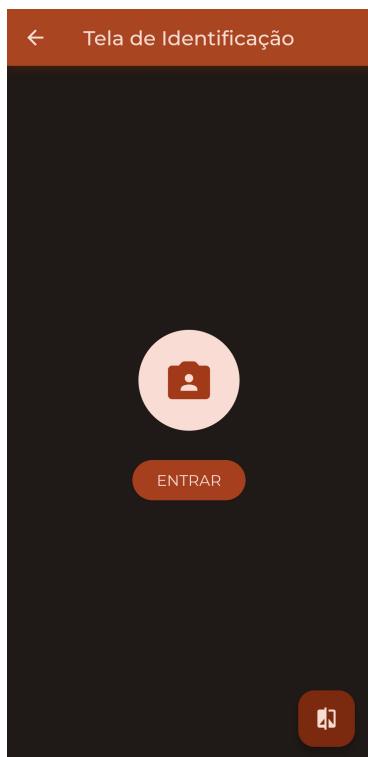


Figura 33 – Tela de Identificação Preenchida.



Fonte: Elaboradas pelo autor.

Figura 34 – Tela de Usuários.



Figura 35 – Tela de Entrada do Usuário.

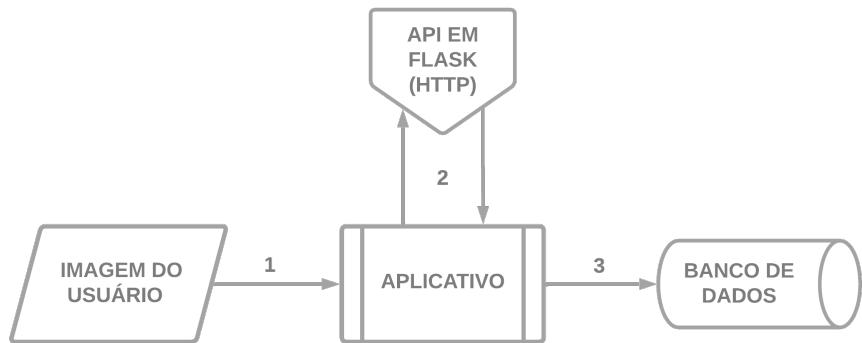


Fonte: Elaboradas pelo autor.

5.2 Registro e Autenticação

A parte principal do aplicativo consiste em dois processos, o processo de registro do usuário, definido no fluxograma apresentado na Figura 36 e a etapa de autenticação do usuário, visualizável no diagrama apresentado na Figura 37.

Figura 36 – Registro do Usuário.

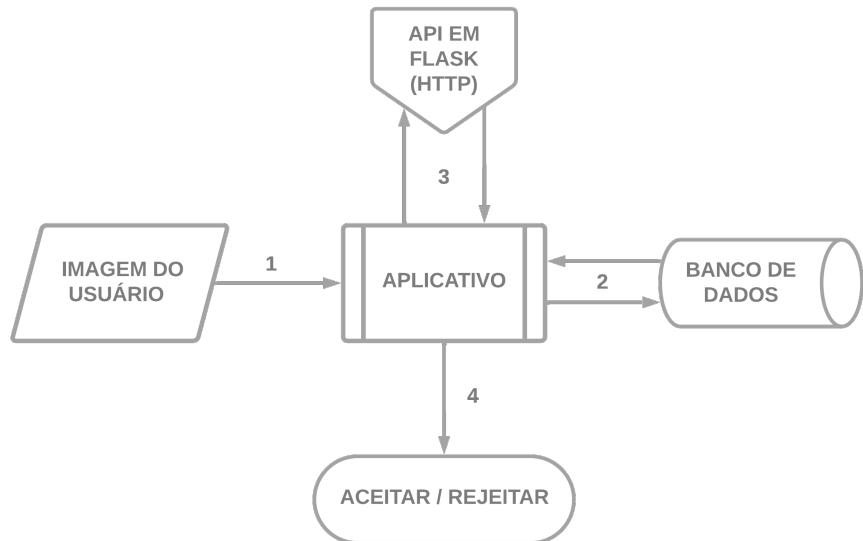


Fonte: Elaborada pelo autor.

A etapa de registro inicia com a entrada da imagem do usuário em conjunto com outros dados que serão armazenados na base de dados, a imagem será passada através do protocolo

Hypertext Transfer Protocol (HTTP) para uma API desenvolvida em *Python*, que irá realizar a extração das características, as retornando para o aplicativo, que em conjunto com as outras informações, serão armazenadas no banco de dados.

Figura 37 – Autenticação do Usuário.



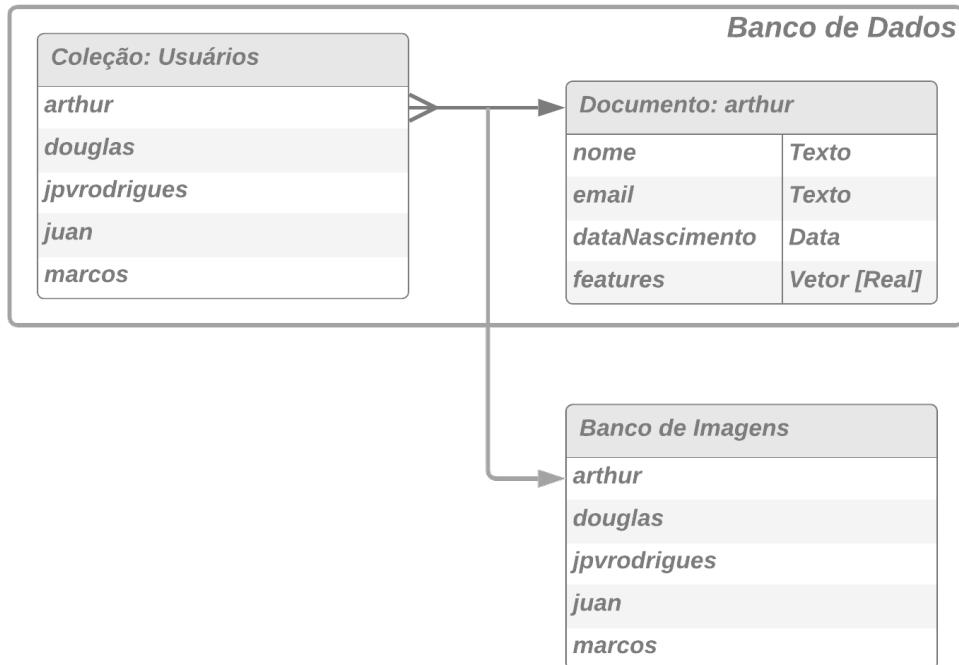
Fonte: Elaborada pelo autor.

Em relação a etapa de autenticação, não apenas a imagem de entrada do usuário é enviada para a API, mas também as características, ou do usuário em específico, no caso de verificação, ou de todos os usuários, no caso de identificação. No primeiro caso a API retornará para o aplicativo o valor de similaridade do usuário, que se for maior que 0,4, definido pela taxa de *EER* do pior caso de teste, da base LFW, ele será aceito, do contrário, rejeitado. Já no segundo caso, o aplicativo receberá a pessoa cadastrada mais similar ao usuário atual, direcionando ela a tela de entrada desse cadastro.

5.2.1 Armazenamento de Dados

Para o banco de dados dos usuários, buscou-se, como é possível verificar na Figura 38, além de armazenar as características extraídas da imagem de entrada, também registrar informações como nome, e-mail e data de nascimento, para isso sendo utilizado um banco de dados não relacional baseado em documentos fornecido pela *Google*, o *Cloud Firestore*. Além dos dados citados, considerou-se guardar também, através do banco de dados não estruturados *Cloud Storage*, a imagem de cadastro do usuário, tanto para apresentação da tela de entrada como para listar os registros existentes no sistema.

Figura 38 – Diagrama do Armazenamento de Dados.



Fonte: Elaborada pelo autor.

Nas Figuras 39 e 40, é demonstrado um exemplo real de como as bases *Cloud Firestore* e *Cloud Storage* são apresentadas e como os dados são estruturados para o administrador da aplicação, podendo verificar principalmente a relação entre o identificador do documento presente no primeiro banco com o nome da imagem presente no segundo.

Figura 39 – *Cloud Firestore*.

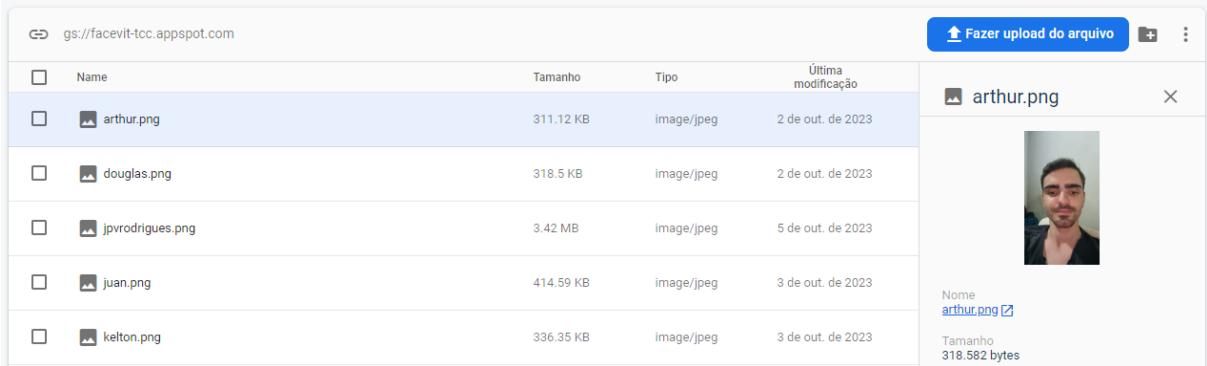
usuarios > arthur

+ Iniciar coleção + Adicionar documento + Iniciar coleção
+ Adicionar campo

(default)	usuarios	arthur
+ Iniciar coleção	+ Adicionar documento	+ Iniciar coleção
usuarios	arthur	+ Adicionar campo
		dataNascimento: "28/05/2002"
	douglas	email: "arthur@email.com"
	jpvrodrigues	features: [0.0323454774916172, 0.12...]
	juan	nome: "Arthur"
	kelton	
	marcos	

Fonte: Elaborada pelo autor.

Figura 40 – *Cloud Storage*.



Fonte: Elaborada pelo autor.

5.3 Resultados e Discussão

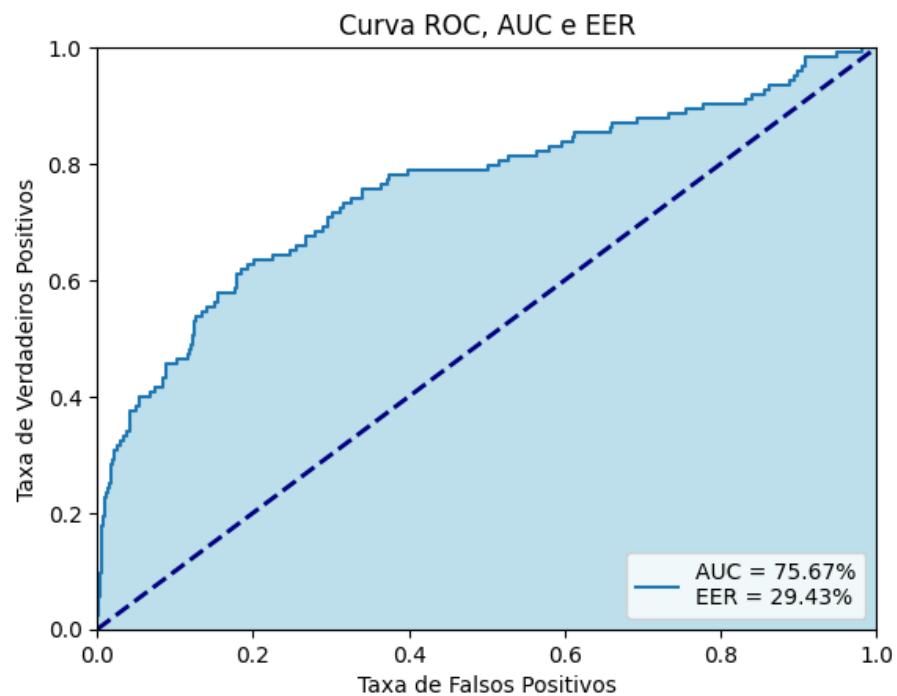
No que diz respeito aos resultados obtidos na aplicação prática do modelo de reconhecimento ViT, além das métricas utilizadas durante a pesquisa, podemos considerar alguns pontos: alta compatibilidade entre a aplicação e o banco de dados, principalmente pelo fato de ambos serem fornecidos pela *Google*, e velocidade de inferência, onde foi verificado que o processo de enviar as informações e receber um retorno da API se mostrou bem rápido, não interferindo na experiência do usuário.

Para realizar a análise dos resultados com base nas métricas utilizadas durante a pesquisa, foi construído um conjunto de imagens, simulando o uso do aplicativo, contando com três a cinco fotos de 16 indivíduos, totalizando aproximadamente 70 imagens distintas de rostos, em momentos diferentes, permitindo analisar o desempenho do modelo em ambientes controlados e não controlados.

Em relação as métricas utilizadas na análise de sistemas de verificação, é possível verificar na Figura 41 que a base de dados construída no aplicativo apresentou um resultado próximo ao encontrado pela base de imagens da LFW durante a fase de testes do modelo, com um valor *AUC* de aproximadamente 75,6% e uma taxa *EER* por volta de 29,4%.

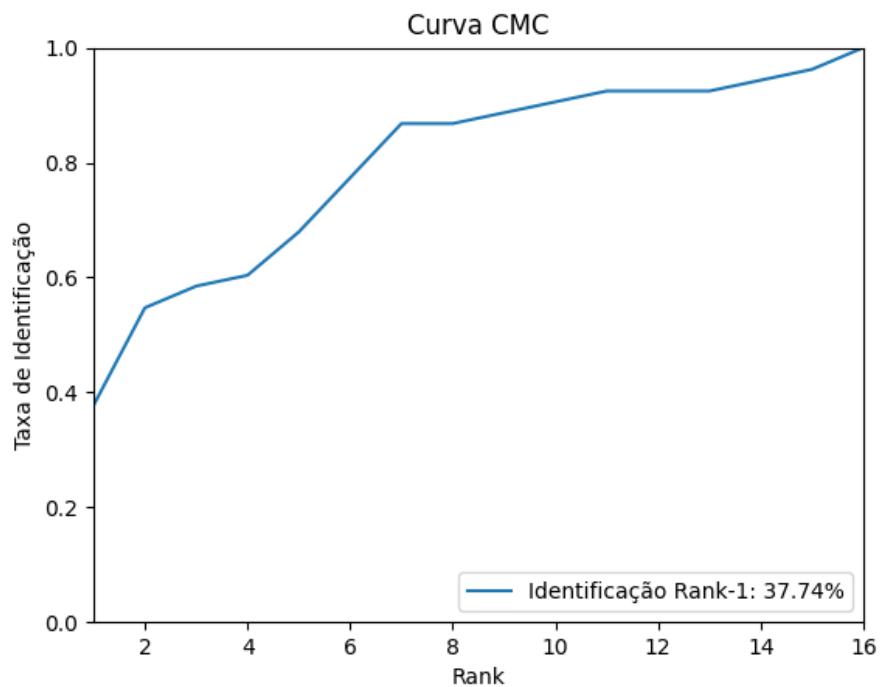
Já no que se refere ao sistema de identificação, a base gerada no aplicativo apresentou um resultado superior sobre a base LFW porém inferior à encontrada pela *Faces94*, constando, como é possível analisar na Figura 42, uma taxa *rank-1* próxima de 37,7%, demonstrando ser capaz de generalizar na atividade de reconhecimento para novos indivíduos, porém ainda assim apresenta uma alta sensibilidade a mudanças de ambiente, iluminação e expressões faciais, diminuindo a acurácia e prejudicando o desempenho, segurança e experiência do usuário do aplicativo.

Figura 41 – Curva ROC, AUC e EER no Aplicativo.



Fonte: Elaborada pelo autor.

Figura 42 – Curva CMC no Aplicativo.



Fonte: Elaborada pelo autor.

6 Conclusão

Como conclusão, o presente trabalho cumpriu com os objetivos propostos, que consistiam em avaliar o desempenho de um modelo de classificação de imagens baseado em transformadores, o ViT, como uma nova alternativa na atividade de reconhecimento facial, onde as técnicas de CNNs ainda são predominantes. Portanto, através do estudo de conceitos teóricos, como aprendizado de máquina, sistemas biométricos e técnicas de análise, foi possível compreender, implementar e avaliar um modelo de reconhecimento próximo ao apresentado por Manesco e Marana (2023), com enfoque inicial para reconhecimento periocular e adaptado para a área de biometria facial.

Os resultados obtidos através das métricas estudadas demonstraram que, mesmo com o uso de uma base de dados de tamanho limitado em comparação às utilizadas por autores como Zhong e Deng (2021), o modelo pré-treinado escolhido e adaptado a base FRGC alcançou bons resultados quando avaliado em ambientes controlados, como visto na base *Faces94*, todavia demonstrou um desempenho inferior em situações de maior instabilidade, como analisado pela base LFW e na elaborada pelo autor, apresentando assim uma maior dificuldade em generalização cruzada e na capacidade de lidar com ruídos e variações de iluminação, posição e expressões faciais.

E por fim, a última etapa do trabalho buscou desenvolver um sistema completo para aplicativos móveis, englobando tanto o processo de registro quanto de autenticação de usuários através de imagens tiradas em tempo real. Assim sendo, foi possível verificar que a aplicação apresentou resultados próximos aos encontrados pela base LFW, porém apresentando uma taxa de identificação superior, e também contou com uma alta velocidade de extração e inferência, permitindo uma melhor usabilidade sem prejudicar a confiabilidade do modelo.

6.1 Trabalhos Futuros

Acerca de novos trabalhos que podem ser realizados a partir dos estudos e resultados obtidos durante a pesquisa, podem-se destacar:

- Avaliar o desempenho do modelo em relação a outros conjuntos de imagens tais como outras técnicas de pré-processamento facial, com destaque na etapa de detecção, a fim de realizar comparativos com métodos do estado da arte.
- Analisar os efeitos no desempenho do modelo ao utilizar outras funções de perda e de similaridade na atividade de reconhecimento facial.

Referências

- ABNAR, S.; ZUIDEMA, W. H. Quantifying attention flow in transformers. *CoRR*, abs/2005.00928, 2020. Disponível em: <<https://arxiv.org/abs/2005.00928>> . Acesso em: 25 de outubro de 2023.
- AHONEN, T.; HADID, A.; PIETIKAINEN, M. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 28, n. 12, p. 2037–2041, 2006. Disponível em: <<https://ieeexplore.ieee.org/document/1717463>> . Acesso em: 13 de setembro de 2023.
- AUNG, H. M. L.; PLUEMPITIWIRIYAWEJ, C.; HAMAMOTO, K.; WANGSIRIPITAK, S. Multimodal biometrics recognition using a deep convolutional neural network with transfer learning in surveillance videos. *Computation*, v. 10, 2022. Disponível em: <<https://www.mdpi.com/2079-3197/10/7/127>> . Acesso em: 11 de abril de 2023.
- BAZAREVSKY, V.; KARTYNNIK, Y.; VAKUNOV, A.; RAVEENDRAN, K.; GRUNDMANN, M. Blazeface: Sub-millisecond neural face detection on mobile gpus. *CoRR*, abs/1907.05047, 2019. Disponível em: <[http://arxiv.org/abs/1907.05047](https://arxiv.org/abs/1907.05047)> . Acesso em: 17 de julho de 2023.
- DECANN, B.; ROSS, A. Can a “poor” verification system be a “good” identification system? a preliminary study. p. 31–36, 2012. Disponível em: <<https://ieeexplore.ieee.org/document/6412621>> . Acesso em: 31 de outubro de 2023.
- DENG, J.; GUO, J.; XUE, N.; ZAFEIRIOU, S. Arcface: Additive angular margin loss for deep face recognition. p. 4690–4699, 2019. Disponível em: <<https://arxiv.org/abs/1801.07698>> . Acesso em: 18 de agosto de 2023.
- DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, 2020. Disponível em: <<https://arxiv.org/abs/2010.11929>> . Acesso em: 08 de abril de 2023.
- EL-ABED, M.; CHARRIER, C.; ROSENBERGER, C. Evaluation of biometric systems. *New Trends and Developments in Biometrics*, 11 2012. Disponível em: <https://hal.science/hal-00990617/file/InTech-Evaluation_of_biometric_systems.pdf> . Acesso em: 11 de junho de 2023.
- GUO, G.; ZHANG, N. A survey on deep learning based face recognition. *Computer Vision and Image Understanding*, v. 189, 2019. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1077314219301183>> . Acesso em: 10 de abril de 2023.
- HUANG, G. B.; RAMESH, M.; BERG, T.; LEARNED-MILLER, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. n. 07-49, October 2007. Disponível em: <<http://vis-www.cs.umass.edu/lfw/lfw.pdf>> . Acesso em: 23 de setembro de 2023.
- KHAN, S.; NASEER, M.; HAYAT, M.; ZAMIR, S. W.; KHAN, F. S.; SHAH, M. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, v. 54, n. 10s, p.

1–41, 2022. Disponível em: <<https://arxiv.org/abs/2101.01169>> . Acesso em: 09 de maio de 2023.

LI, L.; MU, X.; LI, S.; PENG, H. A review of face recognition technology. *IEEE Access*, v. 8, p. 139110–139120, 2020. Disponível em: <<https://ieeexplore.ieee.org/document/9145558>> . Acesso em: 19 de junho de 2023.

LI, S. Z.; JAIN, A. K. *Handbook of Face Recognition*. 2. ed. London: Springer, 2011. Disponível em: <<https://link.springer.com/book/10.1007/978-0-85729-932-1>> . Acesso em: 07 de abril de 2023.

MANESCO, J. R. R.; MARANA, A. N. Combining arcface and visual transformer mechanisms for biometric periocular recognition. *IEEE Latin America Transactions*, v. 21, n. 7, p. 814–820, Jul. 2023. Disponível em: <<https://latamt.ieeer9.org/index.php/transactions/article/view/7811>> . Acesso em: 20 de setembro de 2023.

MINAEE, S.; LUO, P.; LIN, Z.; BOWYER, K. W. Going deeper into face detection: A survey. *CoRR*, abs/2103.14983, 2021. Disponível em: <<https://arxiv.org/abs/2103.14983>> . Acesso em: 17 de junho de 2023.

NGUYEN, H.; BAI, L. Cosine similarity metric learning for face verification. *ACCV*, v. 6493, p. 709–720, 11 2010. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-642-19309-5_55> . Acesso em: 14 de agosto de 2023.

PARKHI, O.; VEDALDI, A.; ZISSERMAN, A. Deep face recognition. 2015. Disponível em: <<https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf>> . Acesso em: 06 de julho de 2023.

PHILLIPS, P.; FLYNN, P.; SCRUGGS, T.; BOWYER, K.; WOREK, W. Preliminary face recognition grand challenge results. p. 15–24, 2006. Disponível em: <<https://ieeexplore.ieee.org/document/1612991>> . Acesso em: 11 de maio de 2023.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. p. 815–823, 2015. Disponível em: <<https://arxiv.org/abs/1503.03832>> . Acesso em: 20 de outubro de 2023.

SERENGIL, S. I. Face alignment for face recognition in python within opencv. 2020. Disponível em: <https://sefiks.com/2020/02/23/face-alignment-for-face-recognition-in-python-within-opencv/> . Acesso em: 13 de outubro de 2023.

SPACEK, L. Libor spacek's facial images databases. *Center for Machine Perception*, 2009. Disponível em: <<https://cmp.felk.cvut.cz/~spacelib/faces/faces94.html>> . Acesso em: 13 de outubro de 2023.

STEINER, A.; KOLESNIKOV, A.; ZHAI, X.; WIGHTMAN, R.; USZKOREIT, J.; BEYER, L. How to train your vit? data, augmentation, and regularization in vision transformers. *CoRR*, abs/2106.10270, 2021. Disponível em: <<https://arxiv.org/abs/2106.10270>> . Acesso em: 15 de junho de 2023.

TURK, M. A.; PENTLAND, A. P. Face recognition using eigenfaces. p. 586–587, 1991. Disponível em: <<https://ieeexplore.ieee.org/document/139758>> . Acesso em: 20 de junho de 2023.

- VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. v. 1, p. I-I, 2001. Disponível em: <<https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>> . Acesso em: 02 de julho de 2023.
- WANG, X.; WANG, S.; CHI, C.; ZHANG, S.; MEI, T. Loss function search for face recognition. *CoRR*, abs/2007.06542, 2020. Disponível em: <<https://arxiv.org/abs/2007.06542>> . Acesso em: 19 de julho de 2023.
- WU, H.; CAO, Y.; WEI, H.; TIAN, Z. Face recognition based on haar like and euclidean distance. *Journal of Physics: Conference Series*, IOP Publishing, v. 1813, n. 1, p. 012036, feb 2021. Disponível em: <<https://dx.doi.org/10.1088/1742-6596/1813/1/012036>> . Acesso em: 14 de agosto de 2023.
- ZHANG, K.; ZHANG, Z.; LI, Z.; QIAO, Y. Joint face detection and alignment using multi-task cascaded convolutional networks. *CoRR*, abs/1604.02878, 2016. Disponível em: <<http://arxiv.org/abs/1604.02878>> . Acesso em: 15 de maio de 2023.
- ZHONG, Y.; DENG, W. Face transformer for recognition. *CoRR*, abs/2103.14803, 2021. Disponível em: <<https://arxiv.org/abs/2103.14803>> . Acesso em: 10 de outubro de 2023.

APÊNDICE A – Pré-processamento Facial

```
import numpy as np
import math
import mediapipe as mp

from PIL import Image
from mediapipe import ImageFormat
from mediapipe.tasks import python
from mediapipe.tasks.python import vision

baseOptions = python.BaseOptions(model_asset_path = "modelDirectory")
options = vision.FaceDetectorOptions(base_options = baseOptions)
detector = vision.FaceDetector.create_from_options(options)

def NormalizeKeypoint(keypoints, imageShape):
    imageHeight, imageWidth, _ = imageShape
    nX, nY = keypoints.x, keypoints.y

    x = min(math.floor(nX * imageWidth), imageWidth - 1)
    y = min(math.floor(nY * imageHeight), imageHeight - 1)

    return x, y

def EuclideanDistance(a, b):
    x1 = a[0]; y1 = a[1]
    x2 = b[0]; y2 = b[1]

    return math.sqrt(((x2 - x1) * (x2 - x1)) + ((y2 - y1) * (y2 - y1)))

def AlignImage(leftEye, rightEye, image):
    if leftEye[1] < rightEye[1]:
        thirdPoint = (leftEye[0], rightEye[1])
        direction = 1
    else:
        thirdPoint = (rightEye[0], leftEye[1])
        direction = -1

    a = EuclideanDistance(leftEye, thirdPoint)
    b = EuclideanDistance(rightEye, leftEye)
    c = EuclideanDistance(rightEye, thirdPoint)

    if c == 0:
        return image
```

```
cosA = (b * b + c * c - a * a) / (2 * b * c)
angle = np.arccos(cosA)
angle = (angle * 180) / math.pi

if direction == -1:
    angle = 90 - angle

rotatedImage = Image.fromarray(image).rotate(direction * angle)
return np.array(rotatedImage)

def DetectAndAlignFace(image):
    mpImage = mp.Image(image_format = ImageFormat.SRGB, data = image)
    detection = detector.detect(mpImage)

    keypoints = detection.detections[0].keypoints
    leftEye = NormalizeKeypoint(keypoints[0], image.shape)
    rightEye = NormalizeKeypoint(keypoints[1], image.shape)

    image = AlignImage(leftEye, rightEye, image)

    mpImage = mp.Image(image_format = ImageFormat.SRGB, data = image)
    detection = detector.detect(mpImage)

    boundingBox = detection.detections[0].bounding_box
    x1, y1 = boundingBox.origin_x, boundingBox.origin_y
    width, height = boundingBox.width, boundingBox.height

    return image[y1:y1 + height, x1:x1 + width]
```

APÊNDICE B – Reconhecimento Facial

```
import torch

from tqdm import tqdm

class Trainer():
    def __init__(self, model, device, optimizer, criterion, scheduler):
        self.model = model

        self.device = device
        self.optimizer = optimizer
        self.criterion = criterion
        self.scheduler = scheduler

        self.train_loss = []
        self.val_loss = []

        self.epochs = 50

    def train(self, train_data, train_size):
        self.model.train()

        running_loss = 0.0

        for inputs, labels in tqdm(train_data):
            inputs = inputs.to(self.device)
            labels = labels.to(self.device)

            self.optimizer.zero_grad()

            with torch.set_grad_enabled(True):
                outputs = self.model(inputs)
                loss = self.criterion(outputs, labels)

                loss.backward()
                self.optimizer.step()

            running_loss += loss.item() * inputs.size(0)

        self.scheduler.step()

        epoch_loss = running_loss / train_size
        self.train_loss.append(float(epoch_loss))
```

```
    print("TRAIN / Loss: {:.4f}\n".format(epoch_loss))

def val(self, val_data, val_size):
    self.model.eval()

    running_loss = 0.0

    for inputs, labels in tqdm(val_data):
        inputs = inputs.to(self.device)
        labels = labels.to(self.device)

        self.optimizer.zero_grad()

        with torch.set_grad_enabled(False):
            outputs = self.model(inputs)
            loss = self.criterion(outputs, labels)

        running_loss += loss.item() * inputs.size(0)

    epoch_loss = running_loss / val_size
    self.val_loss.append(float(epoch_loss))

    print("VAL / Loss: {:.4f}".format(epoch_loss))

def fit(self, train_data, train_size, val_data, val_size):
    for epoch in range(self.epochs):
        print(f'\nEPOCH: {epoch}/{self.epochs - 1}\n')

        self.train(train_data, train_size)
        self.val(val_data, val_size)
```

APÊNDICE C – Curva ROC, AUC e EER

```
import torch
import numpy as np
import matplotlib.pyplot as plt

from scipy.spatial.distance import pdist, squareform
from sklearn.metrics import roc_curve, auc
from tqdm import tqdm

npEmbeddings = np.empty([0, 1024])
npLabels = np.empty(0)

for inputs, labels in tqdm(testDataset):
    inputs = inputs.to(DEVICE)
    labels = labels.to(DEVICE)

    optimizer.zero_grad()

    with torch.set_grad_enabled(False):
        outputs = model(inputs)

    outputs = outputs.cpu().detach().numpy()
    labels = labels.cpu().detach().numpy()

    npEmbeddings = np.vstack(npEmbeddings, outputs)
    npLabels = np.append(npLabels, labels)

x = np.triu(1 - squareform(pdist(npEmbeddings, metric = 'cosine')))
M = x.shape[0]

genuineScores = []
impostorScores = []

for i in tqdm(range(M)):
    for j in range(i, M):
        if (npLabels[i] == npLabels[j] and i != j):
            genuineScores.append(str(x[i][j]))
        elif (npLabels[i] != npLabels[j] and i != j):
            impostorScores.append(str(x[i][j]))

for score in genuineScores:
    genuineScores.append(score)

for score in impostorScores:
```

```
    impostorScores.append(score)

y_score = np.concatenate([genuineScores, impostorScores])
y_true = np.concatenate([
    np.ones(len(genuineScores)), np.zeros(len(impostorScores))
])

fpr, tpr, thresholds = roc_curve(y_true = y_true, y_score = y_score)
roc_auc = auc(fpr, tpr)

eer_index = np.argmin(np.abs(fpr - (1 - tpr)))
eer = (fpr[eer_index] + (1 - tpr[eer_index])) / 2

plt.plot(fpr, tpr, label = f'AUC = {roc_auc:.2%}\nEER = {eer:.2%}')
plt.plot([0, 1], [0, 1], color = 'navy', lw = 2, linestyle = '--')
plt.fill_between(fpr, tpr, color = 'lightblue', alpha = 0.8)

plt.title('Curva ROC, AUC e EER')
plt.legend(loc = 4)

plt.xlabel('Taxa de Falsos Positivos')
plt.xlim([0.0, 1.0])
plt.ylabel('Taxa de Verdadeiros Positivos')
plt.ylim([0.0, 1.0])

plt.show()
```

APÊNDICE D – Curva CMC

```
import torch
import csv
import os
import numpy as np
import matplotlib.pyplot as plt

from numpy.linalg import norm
from PIL import Image
from torchvision import transforms as T

dataset = 'frgc'

folders = os.scandir(f'dataset/gallery/{dataset}')
galleryFeatures = {}

for folder in folders:
    files = os.scandir(f'dataset/gallery/{dataset}/{folder.name}')

    files = [file for file in files]
    file = files[0]

    imageDir = f'dataset/gallery/{dataset}/{folder.name}/{file.name}'
    imagem = Image.open(imageDir)

    faceEmbeddings = model.extractFeatures(imagem)
    galleryFeatures[folder.name] = faceEmbeddings

keys = galleryFeatures.keys()

folders = os.scandir(f'dataset/{dataset}')
folders = [folder for folder in folders]

k = len(folders)

dictCRMAcertos = {i: 0 for i in range(1, k + 1)}
dictCRMTotal = {i: 0 for i in range(1, k + 1)}

for folder in folders:
    files = os.scandir(f'dataset/{dataset}/{folder.name}')
    files = [file for file in files]

    for file in files:
        imageDir = f'dataset/{dataset}/{folder.name}/{file.name}'
```

```

imagem = Image.open(imageDir)

features = model.extractFeatures(imagem)
pares = []

for key in keys:
    npDot = np.dot(features, galleryFeatures[key])
    npNorm = norm(features) * norm(galleryFeatures)

    pares.append((key, npDot / npNorm))

pares = sorted(pares, key = lambda x: -x[1])

for k in range(1, k + 1):
    if any(folder.name == x[0] for x in pares[:k]):
        dictCRMAcertos[k] += 1

dictCRMTOTAL[k] += 1

taxa = [dictCRMAcertos[t] / dictCRMTOTAL[t] for t in range(1, k + 1)]
rank = [i for i in range(1, k + 1)]

plt.plot(rank, taxa, label = f'Identificacao Rank-1: {taxa[0]:.2f}%')

plt.title('Curva CMC')
plt.legend(loc = 4)

plt.xlabel('Rank')
plt.xlim([1, 25])
plt.ylabel('Taxa de Identificacao')
plt.ylim([0.0, 1.0])

plt.show()

```