API para Detecção e Reconhecimento de Faces: Análise da Viabilidade de Fine-tuning em um Modelo Zero-shot

Thiago Edgar Bauce Venancio e Edson Takashi Matsubara

Fundação Universidade Federal do Mato Grosso do Sul

Resumo

Atualmente existem diversos repositórios com partes importantes para reconhecimento facial, no entanto, não é de nosso conhecimento nenhum repositório que contém todos os passos para o reconhecimento com o atual estado da arte. Para solucionar esse problema, propõe-se a construção de uma API que integra modelo de detecção e reconhecimento facial. Deste modo, neste trabalho foram treinados dois modelos: o Retina Face com o conjunto de dados Wider Face, obteve uma taxa de acerto de 94% na detecção de faces, e o Partial FC zero-shot com o MegaFace com uma taxa de acerto de 99,7% no teste LFW. Experimentos de fine-tuning foram conduzidos no modelo Partial FC zero-shot buscando compreender quais são as camadas mais afetadas para o fine-tunning do modelo. A API, implementada em Flask, alcançou o objetivo de detectar e reconhecer faces e encontra-se livremente disponível no github (https://github.com/thiagobauce/API_DET_REC.git).

Abstract

Currently, there are several repositories containing important components for facial recognition. However, to our knowledge, there is no repository that provides all the steps for facial recognition with the current state-of-the-art. To address this issue, we propose the development of an API that integrates facial detection and recognition models. In this work, we trained two models: Retina Face using the Wider Face dataset, achieving a 94% accuracy rate in face detection, and Partial FC zero-shot using MegaFace with a 99.7% accuracy rate on the LFW test. Fine-tuning experiments were conducted on the Partial FC zero-shot model to understand which layers are most affected by fine-tuning. The API, implemented in Flask, successfully achieves the goal of detecting and recognizing faces and is freely available on GitHub (https://github.com/thiagobauce/API_DET_REC.git).

Palavras-chave: Aprendizado de Máquina, ResNet50, ArcFace, Partial FC, Mega Face, Reconhecimento, *Fine-tuning*, *zero-shot* Learning, Retina Face, Flask, API, Métricas.

Keywords: Machine Learning, ResNet50, ArcFace, Partial FC, Mega Face, Recognition, Fine-tuning, zero-shot Learning, Retina Face, Flask, API, Metrics.

1 Introdução

A visão computacional procura capacitar as máquinas a entenderem e interpretarem o mundo visual de forma similar aos seres humanos [3]. Essa área envolve o desenvolvimento de algoritmos e técnicas avançadas para permitir que as máquinas extraiam informações valiosas a partir de imagens e/ou vídeos utilizando algoritmos e técnicas avançadas, abrindo um vasto leque de aplicações em diversos setores, como: medicina, segurança, entretenimento, automação industrial e muito mais [3].

O crescimento contínuo dos estudos em detecção e reconhecimento de faces têm sido impulsionados por avanços significativos na disponibilidade de dados, poder computacional e algoritmos de aprendizado de máquina. Além disso, a crescente demanda por aplicações práticas, como a autenticação biométrica, vigilância de segurança, análise de emoções, realidade virtual e a personalização de serviços, tem estimulado a pesquisa e o desenvolvimento nessa área [4, 32, 33].

Para realizar reconhecimento facial é necessário fazer o recorte das faces, tarefa conhecida por detecção facial, seguida pelo reconhecimento. A detecção de faces envolve a localização de rostos em imagens ou vídeos. A detecção facial é realizada por versões modificadas de algoritmos de detecção de objetos. Neste trabalho é utilizado e apresentado o RetinaFace [1, 2, 7].

Já o reconhecimento facial é feito pelo Partial FC, aliado ao ArcFace, que utiliza uma rede neural profunda para extrair representações faciais, usadas para comparar e identificar faces. O ArcFace supera os desafios tradicionais de varia-

ções de iluminação, expressões faciais e ângulos de pose [1].

O modelo obtido pelo Partial FC e a grande maioria dos modelos de reconhecimento facial gera um vetor, também conhecido por *embedded*. O espaço de representação deste vetor é construído de modo que faces da mesma pessoa fiquem com uma menor distância do que pessoas distintas. A verificação e/ou identificação é feita utilizando o vizinho mais próximo nesta base. Dessa maneira não é necessário treinar o modelo novamente, basta adicionar um novo *embedding* na base de dados. Esta maneira de avaliação é uma das maneiras de fazer o aprendizado *zero-shot* (*zero-shot learning*).

O objetivo do trabalho implementar em uma API com o estado da arte de detecção (Retina-Face) e reconhecimento (Partial FC). Como contribuições para um Trabalho de Conclusão de Curso pode-se elencar:

1. API reconhecimento facial: foi implementada uma API utilizando `Flask` com o pipeline completo que vai desde detectar faces (Retina Face) ao seu reconhecimento (Partial FC).

2. O modelo do Retina Face: foi treinado um modelo do Retina Face utilizando o conjunto de dados Wider Face.

3. O modelo do Partial FC: foi treinado um modelo do Partial FC com o Mega Face, utilizando ArcFace como *loss* para ser o modelo *zero-shot*.

4. Avaliação e influência das camadas no *backbone*: foram feitas variações no processo de *fine-tuning* com congelamentos e treinamentos em diferentes camadas do *backbone* para avaliação da eficiência desse método.

2 Materiais e Métodos

2.1 Resnet50

Essa rede é composta por 50 camadas e possui uma estrutura de blocos residuais, formados por duas ou mais camadas convolucionais seguidas de uma conexão de salto. A informação é passada entre as camadas através dessas conexões de salto, as quais permitem a transmissão direta da informação do gradiente e dos recursos sem passar por muitas camadas intermediárias, facilitando, assim, o treinamento e melhorando o desempenho da rede [5].

Essa arquitetura foi utilizada, neste projeto, como *backbone* do Retina Face e do ArcFace, pois é um dos principais *backbones* utilizados no treinamento de *deep metric learning*, por sua robustez e desempenho [36].

2.2 Retina Face

O Retina Face consegue detectar faces em escalas pequenas, tornando-o adequado para aplicações que envolvem detecção precisa de rostos em imagens com resolução limitada [2] e foi escolhido para este projeto por estar entre os melhores métodos de detecção facial, no conjunto de dados de *benchmark Wider Face (Hard)* [39].

O retina Face utiliza técnicas de *Feature Pyramid Network (FPN)* e *Single Stage Headless (SSH)* incorporadas à arquitetura da ResNet-50 para melhorar o desempenho em tarefas de detecção de faces [20, 21].

A FPN lida com a escala das características da imagem. As características extraídas pela rede base, que poderiam ter diferentes tamanhos devido à pirâmide de convoluções usada pela rede, foram recebidas pela FPN. Ela Foi projetada para combinar e reescalar essas características, criando uma pirâmide de características em diferentes escalas. Essa pirâmide permitiu que o modelo detectasse rostos em várias escalas, tornando-o robusto a rostos de diferentes tamanhos na imagem [20].

A SSH, por sua vez, processa as saídas da FPN e extrai características específicas para a detecção de rostos. Ela consistiu em três sub-redes convolucionais que foram aplicadas nas saídas da FPN em diferentes escalas. Essas sub-redes foram projetadas para identificar padrões e características faciais discriminativas em cada escala da pirâmide de características. Elas ajudaram a refinar e segmentar as regiões de rosto detectadas, fornecendo informações detalhadas para a etapa de detecção e localização de rostos [21].

A função de perda do Retina Face foi definida a partir do *Multibox* [38] e é estendido para lidar com várias categorias de objetos [7, 8]. A função de perda combina a perda de localização, perda de classificação e perda dos pontos de referência, utilizando um hiper parâmetro α para controlar o equilíbrio entre as perdas [8], dada pela Equação

1.

$$L = \alpha L_{loc} + L_{cls} + L_{landm} \tag{1}$$

Onde L_{loc} é a perda de localização, L_{cls} é a perda de classificação e L_{landm} é a perda dos pontos de referência e o hiper parâmetro α que controla o equilíbrio entre as perdas [8].

L_{loc} é definida na Equação 2 sobre uma tupla de *targets* $v = (v_x, v_y, v_w, v_h)$ de regressão de caixa delimitadora verdadeira para a classe u, e uma tupla predita $t_u = (t_{u_x}, t_{u_y}, t_{u_w}, t_{u_h})$, novamente para a classe u [6].

$$L_{loc}(t^u, v) = \sum_{i \in x,y,w,h} Smooth_{L1}(t_i^u - v_i) \tag{2}$$

A $Smooth_{L1}$, Equação 3, é mais robusta que a L2, pois quando os *targets* não são limitados, o treinamento com a L2 exige ajustes cuidadosos das taxas de aprendizado para prevenir gradientes degenerativos [8] .

$$Smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{se} \quad |x| < 1 \\ |x| - 0.5, & \text{caso contrário} \end{cases} \tag{3}$$

L_{cls} é definida na Equação 4 como a softmax para classes binárias (face/não-face), onde p_i é a probabilidade prevista da âncora i ser um rosto e p_i^* é 1 para a âncora positiva e 0 para a âncora negativa [12].

$$L_{cls}(p_i, p_i^*) = \left(\frac{e^{x_1}}{e^{x_1} + e^{x_2}}, \frac{e^{x_2}}{e^{x_1} + e^{x_2}} \right) \tag{4}$$

Por fim, a $L_{landm}(l_i, l^i)$ é a regressão da perda dos cinco pontos de referência faciais, onde $l_i = (l_{x1}, l_{y1}, ..., l_{x5}, l_{y5})_i$ e $l_i^* = (l_{x1}^*, l_{y1}^*, ..., l_{x5}^*, l_{y5}^*)_i$ representam os pontos de referência faciais previstos e os pontos de referência verdadeiros associados à âncora positiva. Assim como na L_{loc}, a função dos cinco pontos de referência faciais também emprega a $Smooth_{L1}$ [2].

2.3 ArcFace

O ArcFace foi projetado para aprimorar o desempenho do reconhecimento facial em tarefas desafiadoras, como a identificação de rostos em diferentes ângulos e condições de iluminação. Essa é uma das razões pelas quais foi escolhido para este projeto, além de estar no estado-da-arte em reconhecimento facial e ser um dos melhores métodos nos benchmarks. [41]

Surgiu como uma alternativa à Softmax que é tradicionalmente utilizada para essa tarefa. Porém a Softmax não otimiza explicitamente a incorporação de características para garantir maior similaridade para amostras da mesma classe e diversidade entre amostras de classes diferentes, o que resulta em uma lacuna de desempenho para o reconhecimento facial profundo sob grandes variações de aparência dentro das classes.

A Softmax convencional [10, 11] é definida como:

$$L_1 = -log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{N} e^{W_j^T x_i + b_j}} \tag{5}$$

onde $x_i \in \mathbb{R}^d$ representa a característica profunda da i-ésima amostra pertencente à classe y_i. A dimensão da característica d é definida como 512 segundo [22, 23]. W representa uma matriz de pesos e b representa um vetor de *bias*.

Em [22, 23] o *bias* é fixado $b_j = 0$, $||W_j|| = 1$ e $||x_2||$ pela normalização l_2 e o reescala para s = 64. E então (Deng et al., 2019) [1] transforma o *logit* [26] como $||W_j^T x_i|| = ||W_j||\ ||x_i||\ cos\theta_j$, onde θ_j é o ângulo entre o peso W_j e a característica x_i.

$$L_2 = -log \frac{e^{s*cos(\theta_{yi})}}{e^{s*cos(\theta_{yi})} + \sum_{j=1, j \neq yi}^{N} e^{s*cos(\theta_j)}} \tag{6}$$

A função de perda, definida pela Equação 7, no ArcFace é definida para maximizar a similaridade entre as características da mesma classe e aumentar a separação entre classes diferentes. Isso é alcançado adicionando uma penalidade angular aditiva m entre as características e os pesos. Essa penalidade pode ser entendida como uma medida da distância angular entre dois pontos em uma hiper esfera unitária.

$$L_3 = -log \frac{e^{s*cos(\theta_{yi}+m)}}{e^{s*cos(\theta_{yi}+m)} + \sum_{j=1, j \neq yi}^{N} e^{s*cos(\theta_j)}} \tag{7}$$

Além do ArcFace, (Deng et al., 2019) [1] propôs a Equação 8 que combina outros os métodos CosFace [23] e SphereFace [22] para melhorar o desempenho do reconhecimento facial, independentemente de a adição ser aplicada no espaço dos ângulos [22] ou no espaço dos cossenos [23].

$$L_4 = -log\frac{e^{s(m_1 cos(\theta_{yi}+m_2)-m_3)}}{e^{s(m_1 cos(\theta_{yi}+m_2)-m_3)} + \sum_{j=1,j\neq yi}^{N} e^{s*cos(\theta_j)}} \tag{8}$$

Na Equação 8 os hiper parâmetros m_1, m_2 e m_3 são, respectivamente, margem angular multiplicativa, margem angular aditiva e margem cosseno aditiva. Cada uma dessas margens tem o objetivo de melhorar a compactação dentro das classes e a diversidade entre as classes, penalizando o valor *logit* alvo [26].

2.4 Partial FC

Na camada *fully-connected* tradicional, todos os neurônios de entrada estão conectados a todos os neurônios de saída, resultando em um grande número de parâmetros [28]. Isso pode ser problemático, especialmente quando o número de classes é muito grande, como no reconhecimento facial, onde podem existir milhares ou até milhões de classes.

A função Softmax é amplamente utilizada para o reconhecimento facial e foi definida por (X. An et al., 2022) [28] através da Equação 9.

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} log\frac{e^{W_{yi}^T xi}}{e^{W_{yi}^T xi} + \sum_{j=1,j\neq y_i}^{C} e^{W_j^T xi}} \tag{9}$$

Onde $W_j \in \mathbb{R}^d$ denota a j-ésima coluna dos centros de cada classe, $x_i \in \mathbb{R}^d$ denota a característica da i-ésima amostra pertencente à classe y_i, d é a dimensão da característica, C é o número de classes e B é o tamanho do lote (batch size).

A perda Softmax é calculada a partir das características e dos centros de classe, conforme a Equação 10.

$$\frac{\partial \mathcal{L}}{\partial xi} = -(1-p^+)W^+ - \sum_{j\in\mathbb{S},j\neq yi}^{C} p_j^- W_j^- \tag{10}$$

Onde p^+ e W^+ representam o centro da classe positiva, p_j^- e W_j^- representam os centros das classes negativas e $\mathbb{S}$ é o subconjunto das classes negativas. A Equação 11 é usada para atualizar os centros de classe.

$$W_j^t = W_j^{t-1} + \eta\Big(\sum_{i\in\mathbb{B}^+}(1-p_i^+)x_i^+ - \sum_{i\in\mathbb{B}^+} p_i^- x_i^-\Big) \tag{11}$$

Onde η é a taxa de aprendizado e $\mathbb{B}^+$ e $\mathbb{B}^-$ são conjuntos de amostras positivas e negativas, respectivamente.

A camada *fully-connected* na função Softmax apresenta três desvantagens principais no reconhecimento facial em larga escala [28]:

1. Confusão do gradiente em caso de conflito entre classes, onde imagens de uma pessoa são erroneamente distribuídas em diferentes classes. Isso afeta a otimização da rede tanto nas características quanto nos centros.

2. Os centros de classes menos populares sofrem poucas atualizações passivas, pois muitas identidades têm poucas imagens na base de dados. Isso pode levar a uma divergência entre os centros e o centro atualizado pelo algoritmo de treinamento [29].

3. O armazenamento e os cálculos da camada *fully-connected* podem sobrecarregar as capacidades das GPUs, especialmente em treinamentos em larga escala. Aumentar o número de GPUs não resolve eficientemente esse problema [1].

O *Partial Fully-Connected* (PFC) é uma variante da camada *fully-connected* desenvolvida para lidar com grandes conjuntos de dados e melhorar a eficiência computacional no treinamento de redes neurais [1]. Ele utiliza amostragem aleatória de uma parte dos centros de classe negativos para calcular a perda Softmax baseada em margem.

$$\frac{\partial \mathcal{L}}{\partial xi} = -(1-p^+)W^+ - \sum_{j\in\mathbb{S},j\neq yi}^{N} p_j^- W_j^- \tag{12}$$

Na Equação 10, $\mathbb{S}$ é o subconjunto de todas as classes negativas e uma classe positiva $|\mathbb{S}| = C*r$, e r é a taxa de amostragem, variando entre 0 e

1. Comparando a Equação 10 com a Equação 12, observa-se que o PFC diminui a possibilidade de conflito entre classes por r, atentando-se que na Equação 10 o itera sobre C (número total de classes), enquanto na Equação 12 itera apenas sobre os centros positivos e parte dos centros negativos representados por N, atualizados pela Equação 11 em cada iteração, reduzindo assim a frequência de atualização dos centros das classes menos frequentes de 1 para r [28].

Isso equilibra o uso de memória e o custo computacional em cada GPU, além de evitar conflitos entre classes e atualizações excessivas nos centros menos populares. O PFC também economiza memória da GPU ao calcular apenas uma parte dos *logit* s Softmax, permitindo que o treinamento em larga escala aproveite o aumento do número de GPUs para melhorar o desempenho [28].

Então, considerando otimizar os recursos e maximizar o desempenho do sistema, deciciu-se por adotar essa abordagem em conjunto com o ArcFace nesse trabalho.

2.5 Métricas de Avaliação

Uma das formas de avaliar os modelos propostos é através das métricas do *classification report* da biblioteca *sklearn*. Neste projeto foram utilizadas as métricas: *Accuracy*, *Precision*, *Recall* e *F1-Score*.

A *Accuracy* é definida na Equação 13 como razão entre a soma das predições corretas (positivas e negativas) e o número total de predições. [42]

$$Accuracy = \frac{True^{+} + True^{-}}{True^{+} + True^{-} + False^{+} + False^{-}} \tag{13}$$

Enquanto, a *Precision*, definida na Equação 14, é a qualidade de predições de uma classe específica, por exemplo, a classe positiva. [42]

$$Precision = \frac{True^{+}}{True^{+} + False^{+}} \tag{14}$$

O *Recall* mede quanto o modelo acerta em relação número total de acertos esperados. A Equação 15 explicita sua fórmula. [42]

$$Recall = \frac{True^{+}}{True^{+} + False^{-}} \tag{15}$$

Já o *F1-Score* é uma métrica que combina as medidas de *precision* e *recall* em um único valor, fornecendo uma medida geral do desempenho do modelo. A fórmula para calcular o F1-score está na Equação 16. [42]

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{16}$$

Neste trabalho foi utilizado o *F1-Score* Macro, pois todas as classes têm a mesma importância e desejou-se obter uma visão geral do desempenho do modelo em todas as classes de forma equilibrada.

3 Proposta

3.1 API

Foi desenvolvida uma API em `Python` na versão 3.7 utilizando o framework `Flask`. Dividida em dois serviços: detecção, promovida pelo Retina Face, e reconhecimento de faces, pelo Partial FC.

Ao iniciar a API, uma página *Web* inicial é carregada, contendo dois campos separados para *upload* de arquivos, conforme ilustrado na Figura 2, e dois botões que redirecionam o usuário para a página de resultados, conforme o serviço selecionado.

A Figura 1 mostra o fluxo seguido pela API. Ao carregar uma imagem para o servidor, que está executando o núcleo da API com os serviços de detecção e reconhecimento, a imagem é armazenada em um diretório com a data atual. Esse diretório é organizado de forma hierárquica, seguindo a estrutura de ano, mês e dia.

Para cada serviço da API, foi criada uma rota específica. Ao clicar no botão `"Detectar faces"`, o serviço de detecção de faces é acionado na rota `"/detect"`. Esse serviço redimensiona a imagem, carrega o modelo treinado Retina Face e realiza as operações necessárias para desenhar caixas delimitadoras ao redor dos rostos e localizar os 5 pontos de referência faciais.

Em seguida, a imagem é recortada nas coordenadas das caixas delimitadoras obtidas pelo modelo. Essa imagem recortada é redimensionada para o tamanho de 150x150 e salva em uma pasta específica na máquina local. Após detectar todos os rostos na imagem, o usuário é redirecionado para a rota `"/faces_detected"`, onde as imagens recortadas são exibidas em uma linha.

Caso o usuário escolha o serviço de reconhecimento facial, o serviço correspondente é acionado. Antes da etapa de reconhecimento, é feita

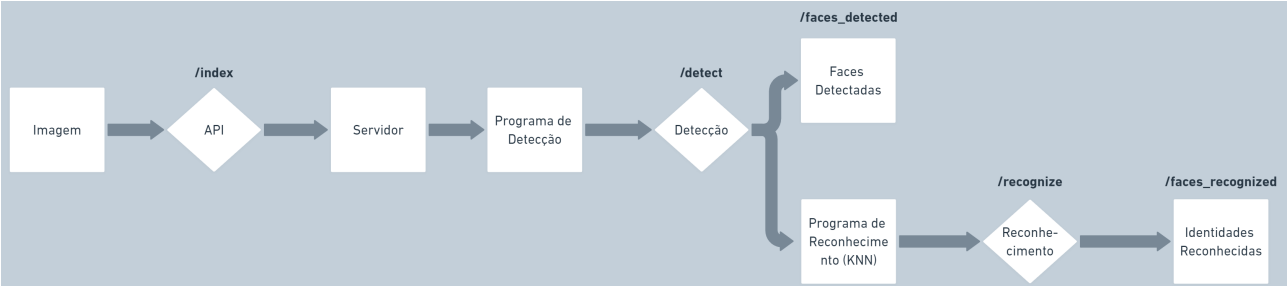

Figura 1: Fluxograma de utilização e funcionamento da API

Figura 2: Faces recortadas nas caixas delimitadoras

uma chamada interna ao serviço de detecção descrito anteriormente, para que o reconhecimento seja realizado apenas nas imagens recortadas das faces.

Com as faces recortadas, o serviço de reconhecimento é acionado na rota `"/recognize"`. Esse serviço carrega o modelo treinado com melhor desempenho e cria um vetor de *embedded* para cada face. Esses vetores são então submetidos a um algoritmo de *K-Nearest Neighbors (KNN)* para verificar a probabilidade da classe que mais se aproxima (`k` = 1). Em seguida, o arquivo da imagem recortada é renomeado com o nome da classe correspondente.

Após esse processo, o usuário é redirecionado para a página na rota `"/faces_recognized"`, onde as imagens são exibidas com os nomes dos arquivos, que são referências às identidades correspondentes.

4 Avaliação Experimental

Para realização de todos os procedimentos desta seção foi utilizada uma máquina com CPU *AMD Ryzen Threadripper 1900X 8-Core Processor*, 64 GB de memória Ram e duas GPUs RTX-3080 TI com 12GB de memória.

4.1 Preparação dos Dados

O conjunto de dados utilizado nos treinos e testes no modelo de detecção facial, foi o Wider Face [9] que contém 12880 imagens de treino e 16097 imagens de teste. Este último é dividido em *easy*, *medium* e *hard*.

1. *Wider Face Easy*: As imagens são selecionadas de forma que as faces sejam mais fáceis de detectar, contêm rostos em poses frontais, com iluminação adequada e fundos simples. As faces podem ser bem visíveis e não apresentam obstruções, como óculos ou mãos na frente do rosto. [9]

2. *Wider Face Medium*: As imagens podem ter maior variabilidade em termos de poses e iluminação, podendo estar em diferentes ângulos e podem ser parcialmente obstruídas por objetos, óculos, cabelos ou mãos. [9]

3. *Wider Face Hard*: As imagens contêm cenas complexas, como multidões, eventos ao ar livre ou situações em que as faces são muito pequenas ou distantes. Além disso, as faces podem estar em poses incomuns, ter iluminação desafiadora, ser obstruídas por objetos e ter expressões faciais variadas. [9]

Foi utilizado o conjunto de dados Mega Face [12] que contém 4574213 imagens e 657078 identidades. Na Figura 3 estão presentes alguns exemplos de imagens desse conjunto. Para teste, foi utilizado o conjunto de teste do Mega Face *(Facescrub)*, com 3530 imagens e 80 identidades.

Um *dataset*, denominado Família, foi montado com 23 identidades de familiares e amigos com 256 imagens. A Figura 4 exemplifica algumas

Figura 3: Algumas imagens do *dataset* Mega Face.

imagens pertencentes a esse conjunto de dados. Esse *dataset* foi separado, sendo 19 classes com 226 imagens para treino e 30 imagens com as 4 classes restantes para teste.

Figura 4: Algumas imagens do *dataset* Família.

A Tabela 1 representa em números os conjuntos de dados de treino, validação e teste.

Tabela 1: Tabela de dados de treino dos experimentos

Dataset	**Quantidade de Imagens**	**Split**
Wider Face	12880	Treino
Mega Face	4574213	Treino
Família	226	Treino
AGEDB_30	12000	Validação
LFW	12000	Validação
CPLFW	12000	Validação
CFP_FF	14000	Validação
CFP_FP	14000	Validação
CALFW	12000	Validação
Família	30	Teste
Mega Face *(Facescrub)*	3530	Teste
Wider Face	16097	Teste

Definiram-se, ainda pares positivos como duas imagens pertencentes à mesma classe (identidade), e pares negativos como duas imagens pertencentes a classes distintas, sendo 8000 pares positivos e 24880 negativos no Mega Face *(Facescrub)* e 110 pares positivos e 325 pares negativos no Família Teste, para os testes dos modelos de reconhecimento facial.

Para a verificação de tais pares utilizou-se a similaridade de cosseno do *Pytorch*. Essa fórmula calcula a similaridade de cosseno como a medida do ângulo entre os dois vetores. Valores próximos a 1 indicam alta similaridade, enquanto valores próximos a -1 indicam dissimilaridade.

$$Similaridade = \frac{x_1 \cdot x_2}{max(||x_1||_2 \cdot ||x_2||_2, \epsilon)} \quad (17)$$

Na Equação 17, $x_1 \cdot x_2$ é o produto escalar dos vetores x_1 e x_2, $||x_1||_2$ e $||x_2||_2$ são as normas desses vetores e ϵ é um valor muito pequeno, apenas para evitar divisão por zero.

Através de testes, foi fixado o $threshold = 0.4$ para definir se era par positivo ou negativo.

4.2 Retina Face

Utilizaram-se apenas as camadas 2, 3 e 4 da ResNet50, onde foram aplicas SSH que processaram as saídas das camadas FPN.

Em seguida, foram criadas três cabeças, uma de classificação, uma de caixa delimitadora e, por fim, a dos pontos de referência.

A cabeça de classificação foi responsável por gerar *scores* de detecção. Ela consistiu em uma camada de convolução 1x1 que recebeu as características de entrada e produziu um tensor de saída.

A cabeça de detecção de caixa delimitadora foi responsável por gerar as coordenadas das caixas delimitadoras. Ela também consistiu em uma camada de convolução 1x1 que recebeu as características de entrada e produziu um tensor de saída.

Por fim, a cabeça de detecção de pontos de referência faciais foi responsável por gerar as coordenadas dos pontos de referência. Ela também consistiu em uma camada de convolução 1x1 que recebeu as características de entrada e produziu um tensor de saída.

Nas 3 cabeças o tensor foi permutado para alterar a ordem dos eixos e, em seguida, redimensionado para obter a forma correta dos *scores* de detecção, das coordenadas dessas caixas e das coordenadas dos pontos de referência, respectivamente.

O modelo foi treinado utilizando o *dataset* Wider Face [9], com um tamanho de lote de 8 e 100 épocas de treinamento, totalizando 1610 iterações em cada época.

Foi empregado $weight\_decay = 0.0005$ nas épocas 70 e 90, SGD com otimizador, taxa de aprendizagem inicial $= 0.001$, $momentum = 0.9$, $\gamma = 0.1$. Além disso, foram utilizados 2 *workers* para o carregamento dos dados.

Tabela 2: Resultados da Inferência Retina Face

Dataset	***Precision***
Easy	0.944
Medium	0.933
Hard	0.843

A Tabela 2 retrata os resultados obetidos pelo modelo Retina Face treinado nos testes com as partições *easy*, *medium* e *hard* do Wider Face.

Figura 5: Imagem de entrada no modelo Retina Face.

Figura 6: Saída do modelo Retina Face com as caixa delimitadoras e os cinco pontos de referência.

A Figura 5 é uma imagem de entrada para o modelo de detecção do Retina Face e tem como saída a Figura 6, onde as faces estão delimitadas pelas caixas e os cinco pontos de referência estão marcados. No processo, apenas os $k = 5$ melhores *scores* são mantidos e assim os resultados ficaram mais assertivos.

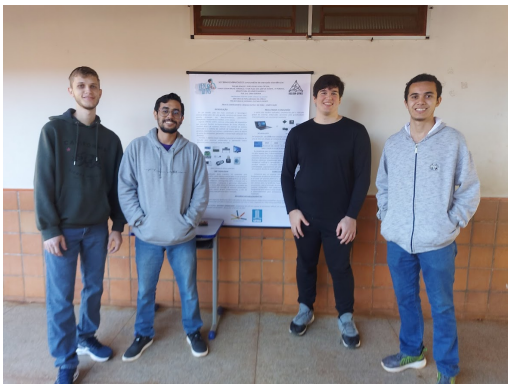

Figura 7: Imagem de entrada na API para detecção.

Após esse processo de detecção das faces e dos pontos de referência, para utilização do serviço no reconhecimento foi de interesse do projeto que essas imagens fossem cortadas nas faces detectadas e delimitadas pelas caixas, pegando a Figura 7 como entrada, tem-se como saída a Figura 8.

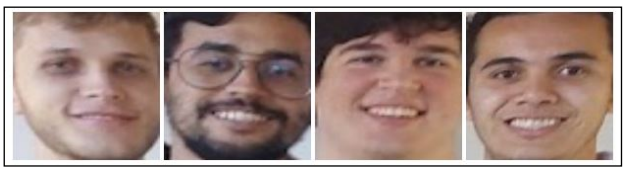

Figura 8: Faces recortadas nas caixas delimitadoras

O modelo obteve êxito em detectar as faces presentes nas imagens de diversos tamanhos, havendo uma limitação quando as faces estavam muito pequenas, ou quando a imagem submetida tinha dimensões grandes. Por exemplo: uma imagem de tamanho 4624x3468 foi submetida e houve

erro na detecção das faces, somente quando a imagem foi redimensionada para 1200x900, o método obteve um resultado assertivo.

4.3 Partial FC

Para o *fine-tuning* modificou-se a última camada da rede neural, substituindo a camada *fully-connected* existente por uma nova camada linear.

Foram realizados 4 experimentos para reconhecimento facial:

1. Experimento *zero-shot*: treinou-se o modelo *zero-shot* com o *dataset* Mega Face e avaliou seu desempenho nos *datasets* Família e Mega Face Teste.

2. Experimento *fine-tuning*: treinou-se um modelo com o *dataset* Família e comparou seu desempenho com o modelo *zero-shot* nos mesmos datasets de teste.

3. Experimento Mega Face + Família: treinou-se um outro modelo com os *datasets* Mega Face e Família concatenados e comparou seu desempenho com o modelo *zero-shot* com os testes nos mesmos datasets de teste.

4. Experimento Variação nas camadas: treinaram-se modelos variando diferentes camadas da rede.

Os *datasets* de validação foram utilizados para ajuste dos hiper parâmetros, número de épocas e seleção do otimizador, entre AdamW e SGD.

4.3.1 Experimento Zero-shot

Neste experimento um modelo *zero-shot* foi treinado com a base de dados do Mega Face, sendo a *loss* o ArcFace e utilizando Partial FC com amostragem = 1, ou seja, atualizando todos os centros de classe. Tamanho de lote = 128, *momentum* = 0.9, taxa de aprendizagem inicial = 0.1, otimizador SGD, 2 *workers* usados no carregamento de dados, *weight_ decay* = 0.0005 e 10 épocas de treinamento.

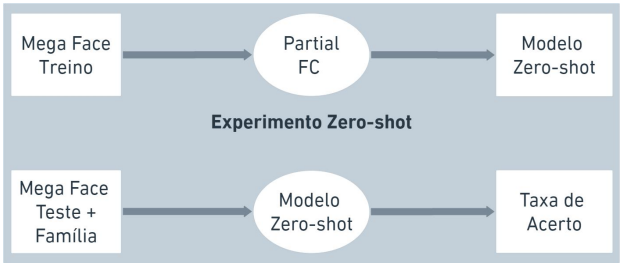

Figura 10: Fluxo de trabalho do experimento zero-shot.

A Figura 10 retrata o fluxo de trabalho do modelo *zero-shot*. Após o treino d modelo, utilizaram-se os *splits* de teste do MegaFace e Família para avaliar o modelo.

Tabela 3: Resultados de Inferência modelo *zero-shot*

Dataset	**Tempo de Inferência**	***Accuracy***
lfw	18.751	0.998
cfp_fp	21.898	0.965
cfp_ff	21.918	0.998
cplfw	19.270	0.913
agedb_30	18.734	0.977
calfw	18.840	0.960

Observa-se, pela Tabela 3 que o modelo *zero-shot* obteve resultados expressivos em termos de *accuracy* em diversos conjuntos de dados, como lfw (0.998), cfp_ff (0.998), e agedb_30 (0.977). Esses valores altos de *accuracy* indicam que o modelo apresentou um desempenho muito bom na classificação correta das imagens nesses conjuntos de dados. No entanto, é importante observar que o desempenho variou de acordo com o conjunto de dados, com algumas variações na *acuracy*, como no caso de cfp_fp (0.965) e cplfw (0.913).

A Tabela 4 apresenta os resultados do teste de identificação de pares positivos e negativos do modelo *zero-shot* em duas categorias: "Família Teste"e "Mega Face *(Facescrub)*". O tempo de inferência foi alto no Mega Face, pois são muitas imagens, indicando uma demanda computacional significativa. Em relação à *precision*, o modelo obteve uma *precision* menor para pares positivos na categoria "Família Teste", enquanto na categoria "Mega Face *(Facescrub)*"a *precision* foi mais alta. Para pares negativos, o modelo alcançou alta *precision* em ambas as categorias. A média da *precision* também foi relativamente alta para ambas as categorias.

Considerando ainda a Tabela 14, pode-se no-

Tabela 4: Teste de Identificação de Pares Positivos e Negativos do modelo *zero-shot*

Dataset	**Tempo de Inferência**	***Precision* Pares Positivos**	***Precision* Pares Negativos**	***Average Precision***
Família Teste	12.285	0.591	0.969	0.780
Mega Face *(Facescrub)*	778.392	0.954	0.997	0.977

tar, ainda, que as métricas se aproximam da *precision* média na identificação dos pares positivos e negativos, reforçando seu resultado.

4.3.2 Experimento Fine-tuning

Neste experimento, conforme Figura 11, o modelo *zero-shot* foi carregado como *backbone* e foi feito um *fine-tuning* sem congelar quaisquer camadas, treinando-o apenas com o *dataset* Família, com exceção, do tamanho do lote, reduzido para 8 e a quantidade de épocas aumentada pra 20, os outros parâmetros de treino continuaram os mesmos.

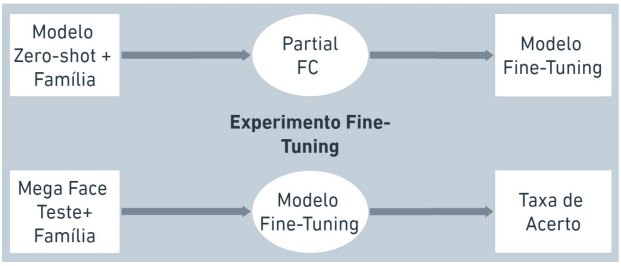

Figura 11: Fluxo de trabalho do experimento fine-Tuning.

Os resultados do primeiro teste do modelo *fine-tuning* estão apresentados na Tabela 5. Nessa tabela, são exibidos os resultados de inferência em diferentes conjuntos de dados, incluindo o tempo de inferência e a *accuracy* do modelo. Observa-se que o tempo de inferência varia entre os conjuntos de dados, com valores variando de 18.071 a 20.190. Quanto à *accuracy*, os valores obtidos estão na faixa de 0.504 a 0.680.

Tabela 5: Resultados de Inferência modelo *fine-tuning*

Dataset	**Tempo de Inferência**	***Accuracy***
lfw	18.459	0.680
cfp_fp	20.190	0.593
cfp_ff	19.788	0.675
cplfw	18.071	0.533
agedb_30	18.074	0.504
calfw	18.068	0.582

Na Tabela 6, que mostra os resultados do teste de identificação de pares positivos e negativos do modelo *fine-tuning*, observa-se que o tempo de inferência também varia entre as categorias Família Teste e Mega Face *(Facescrub)*, com valores de 11.210 e 692.120, respectivamente. Em relação à *precision*, o modelo apresenta uma *precision* mais alta para pares positivos na categoria "Família Teste"(0.700), enquanto a *precision* para pares negativos é maior na categoria Mega Face *(Facescrub)* (0.306). A *average precision* também é inferior ao modelo *zero-shot*, com valores de 0.643 para Família Teste e 0.602 para Mega Face *(Facescrub)*.

Com base nos resultados apresentados na Tabela 5, pode-se concluir que o *fine-tuning* teve um impacto negativo no modelo em comparação com o modelo *zero-shot*. Na Tabela 6, observa-se que, apesar da melhora na identificação de pares positivos no conjunto de dados Família Teste, os demais resultados foram inferiores ao modelo *zero-shot*, indicando uma abordagem menos eficiente.

4.3.3 Experimento Mega Face + Família

Nesta fase, como mostra a Figura 12, o modelo *zero-shot* foi carregado como *backbone* e foi feito um *fine-tuning* sem congelar nenhuma camada, novamente, porém, com a diferença agora de ter os *datasets* Mega Face e o *split* de treino de Família, concatenados, resultando em 657097 identidades e 4574439 imagens.

Tabela 6: Teste de Identificação de Pares Positivos e Negativos do modelo *fine-tuning*

Dataset	**Tempo de Inferência**	***Precision* Pares Positivos**	***Precision* Pares Negativos**	***Average Precision***
Família Teste	11.210	0.700	0.587	0.643
Mega Face *(Facescrub)*	692.120	0.898	0.306	0.602

Tabela 8: Teste de Identificação de Pares Positivos e Negativos do modelo Mega Face + Família

Dataset	**Tempo de Inferência**	***Precision* Pares Positivos**	***Precision* Pares Negativos**	***Average Precision***
Família Teste	12.221	0.600	0.926	0.763
Mega Face *(Facescrub)*	755.134	0.951	0.999	0.976

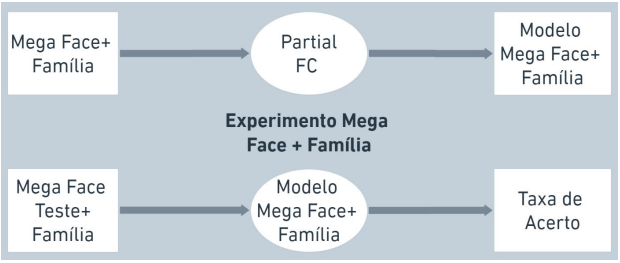

Figura 12: Fluxo de trabalho do experimento Mega Face + Família.

Os resultados do modelo Mega Face + Família estão apresentados na Tabela 7. Nessa tabela, são exibidos os resultados de inferência em diferentes conjuntos de dados, incluindo o tempo de inferência e a *accuracy* do modelo. Observa-se que o tempo de inferência varia entre os conjuntos de dados, com valores variando de 18.632 a 26.776. Quanto à *accuracy*, os valores obtidos estão na faixa de 0.909 a 0.997.

Tabela 7: Resultados de Inferência modelo Mega Face + Família

Dataset	**Tempo de Inferência**	***Accuracy***
lfw	26.776	0.996
cfp_fp	21.667	0.958
cfp_ff	22.645	0.997
cplfw	19.857	0.909
agedb_30	18.711	0.976
calfw	18.632	0.956

Na Tabela 8, que mostra os resultados do teste de identificação de pares positivos e negativos do modelo Mega Face + Família, observa-se que o tempo de inferência também varia entre as categorias Família Teste e Mega Face *(Facescrub)*, com valores de 12.221 e 755.134, respectivamente. Em relação à *precision*, o modelo apresenta uma *precision* um pouco maior para pares positivos na categoria "Família Teste"(0.600), enquanto a *precision* para pares negativos é mais alta na categoria Mega Face *(Facescrub)* (0.999). A *average precision* é relativamente alta para ambas as categorias, com valores de 0.763 para Família Teste e 0.976 para Mega Face *(Facescrub)*.

Ao comparar esses resultados com o modelo *zero-shot*, nota-se que o modelo Mega Face + Família apresenta números ligeiramente menores nos conjuntos de dados de avaliação. Na Tabela 8, observa-se uma pequena vantagem na identificação de pares positivos no conjunto de dados Família Teste e na identificação de pares negativos no conjunto de dados Mega Face *(Facescrub)*. No entanto, essa vantagem é compensada pela diferença inferior nas outras duas métricas, indicando que essa técnica pode não ser viável em termos gerais.

4.3.4 Experimento Variação das Camadas

A Figura 13 ilustra como foram feitos os procedimentos desse experimento, carregou-se o modelo *zero-shot* como *backbone* e vários *fine-tunings* foram feitos, mantendo ou modificando as últimas camadas da rede da seguinte forma:

1. L4: Treinando apenas a camada 4 e congelando as outras.

2. L3: Treinando apenas a camada 3 e congelando as outras.

3. L34: Treinando apenas as camadas 3 e 4 e congelando as outras.

4. L234: Treinando apenas as camadas 2, 3 e 4 e congelando as outras.

5. L4FC: Treinando apenas a camada 4 e congelando as outras e modificando a camada *fully-connected* e a camada *classifier*.

6. L3FC: Treinando apenas a camada 3, congelando as outras e modificando a camada *fully-connected* e a camada *classifier*.

7. L34FC: Treinando apenas as camadas 3, 4, congelando as outras e modificando a camada *fully-connected* e a camada *classifier*.

8. L234FC: Treinando apenas as camadas 2, 3 e 4, congelando as outras e modificando a camada *fully-connected* e a camada *classifier*.

9. L4ZS: Treinando apenas a camada 4 e congelando as outras, porém treinado com os *datasets* concatenados utilizados pelo Experimento Mega Face + Família.

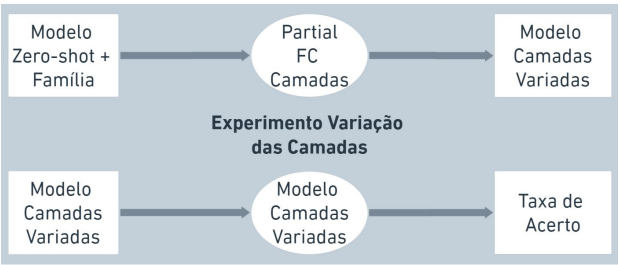

Figura 13: Fluxo de trabalho do experimento Variação das Camadas.

Esses modelos gerados acima foram treinados apenas com o *dataset* Família, com exceção do modelo L4ZS. E após os treinos, os modelos foram submetidos aos testes para comparação e avaliação da vialibidade.

Tabela 9: Resultados de Inferência modelo L4ZS

Dataset	**Tempo de Inferência**	***Accuracy***
lfw	19.292	0.996
cfp_fp	22.310	0.918
cfp_ff	21.917	0.997
cplfw	19.273	0.884
agedb_30	19.369	0.968
calfw	18.954	0.956

A Tabela 9 apresenta os resultados de inferência do modelo L4ZS em diferentes conjuntos de dados. Observa-se que o tempo de inferência varia entre 18.954 e 22.310, com a maior duração ocorrendo no conjunto de dados cfp_fp. Quanto à *accuracy*, os valores variam entre 0.884 e 0.997, sendo a *accuracy* mais baixa no conjunto de dados cplfw.

Na Tabela 10, são fornecidos os resultados do teste de identificação de pares positivos e negativos para o modelo L4ZS. No conjunto de dados Família Teste, o tempo de inferência é de 12.175. A *precision* para pares positivos é de 0.600, indicando que cerca de 60% dos pares de faces positivas são corretamente identificados. Para pares negativos, a *precision* é de 0.972, sugerindo que aproximadamente 97.2% dos pares de faces negativas são corretamente identificados. A *average precision* é de 0.786.

o modelo L4ZS apresenta resultados relativamente bons em termos de *accuracy* e *precision* na identificação de pares positivos e negativos, com exceção do conjunto de dados cfp_fp, que possui uma *accuracy* mais baixa. No entanto, é importante considerar que esse modelo possui um tempo de inferência consideravelmente maior, especialmente no conjunto de dados Mega Face *(Facescrub)*.

Tabela 10: Teste de Identificação de Pares Positivos e Negativos do modelo L4ZS

Dataset	**Tempo de Inferência**	***Precision* Pares Positivos**	***Precision* Pares Negativos**	***Average Precision***
Família Teste	12.175	0.600	0.972	0.786
Mega Face *(Facescrub)*	755.134	0.744	0.999	0.872

Tabela 12: Teste com Família Teste

Modelo	**Tempo de Inferência**	***Precision* Pares Positivos**	***Precision* Pares Negativos**	***Average Precision***
L4	12.270	0.718	0.812	0.765
L3	11.840	0.518	0.769	0.643
L34	12.080	0.791	0.572	0.681
L234	11.830	0.791	0.341	0.566
L4FC	12.330	0.754	0.677	0.715
L3FC	12.300	0.754	0.483	0.618
L34FC	12.590	0.481	0.867	0.674
L234FC	12.140	0.818	0.566	0.692

Tabela 11: Resultados de Inferência Diversos Fine-tunings no *dataset* LFW

Camadas Treinadas	**Tempo de Inferência**	***Accuracy***
L4	18.760	0.874
L3	18.710	0.725
L34	18.670	0.734
L234	18.780	0.695
L4FC	18.690	0.868
L3FC	18.670	0.695
L34FC	18.760	0.686
L234FC	18.820	0.695

A Tabela 11 resume os resultados de inferência de diferentes fine-tunings no conjunto de dados lfw. Os fine-tunings L4 e L4FC se destacam, alcançando altos números de 0.874 e 0.868, respectivamente, enquanto o fine-tuning L234FC apresenta a menor *accuracy*, com 0.695. Embora os tempos de inferência sejam semelhantes para todos os fine-tunings, esses resultados evidenciam a eficácia do L4 e L4FC na tarefa.

A Tabela 12 apresenta os resultados do teste com a Família Teste para os diferentes modelos. Observa-se que os modelos L4 e L4FC obtiveram tempos de inferência semelhantes, sendo 12.270 e 12.330, respectivamente. Em relação à *precision* dos pares positivos, o modelo L234FC obteve o valor mais alto, com 0.818, enquanto o modelo L3 apresentou a menor *precision*, com 0.518. Já para a *precision* dos pares negativos, o modelo L34FC obteve o maior valor, com 0.867, enquanto o modelo L234 teve o menor, com 0.341. Quanto à métrica de *average precision*, o modelo L234FC obteve o maior valor, com 0.692, e o modelo L3FC teve o menor, com 0.618.

NA Tabela 13 estão os resultados com o conjunto Mega Face (Facescrub). Observa-se que o modelo L3FC obteve o menor tempo de inferência, com 733.95, enquanto o modelo L234 teve o maior tempo, com 782.13. Em relação à *precision* dos pares positivos, o modelo L4FC obteve o valor mais alto, com 0.997, e o modelo L4 apresentou a menor *precision*, com 0.718. Já para a *precision* dos pares negativos, o modelo L34FC obteve o maior valor, com 0.455, enquanto os modelos L234 e L234FC tiveram as menores *precisions*, com 0.176 e 0.238, respectivamente. Quanto à métrica de *average precision*, o modelo L234FC obteve o maior valor, com 0.582, e o modelo L34 teve o menor, com 0.557.

A Tabela 14 apresenta as métricas dos modelos de reconhecimento facial submetidos ao KNN com o conjunto de dados Família Treino. Os melhores desempenhos foram observados nos modelos L4, *zero-shot*, L4FC e L3, com valores de *accuracy* de 0.947, 0.746, 0.933 e 0.827, respecti-

Tabela 13: Teste com Mega Face *(Facescrub)*

Modelo	**Tempo de Inferência**	***Precision* Pares Positivos**	***Precision* Pares Negativos**	***Average Precision***
L4	742.234	0.718	0.812	0.765
L3	755.141	0.889	0.230	0.559
L34	755.113	0.929	0.184	0.557
L234	782.136	0.955	0.176	0.565
L4FC	734.069	0.997	0.202	0.599
L3FC	733.958	0.971	0.155	0.563
L34FC	738.981	0.826	0.455	0.641
L234FC	760.187	0.925	0.238	0.582

Tabela 14: Métricas dos modelos de reconhecimento submetidos ao KNN com Família Treino

Modelo	**Accuracy**	**Precision**	**Recall**	**F1-Score**
zero-shot	0.746	0.891	0.750	0.775
fine-tuning	0.480	0.520	0.478	0.481
Mega Face + Família	0.787	0.841	0.785	0.785
L4ZS	0.760	0.848	0.763	0.762
L4	0.947	0.964	0.947	0.948
L3	0.827	0.848	0.829	0.815
L34	0.707	0.771	0.706	0.703
L234	0.587	0.652	0.588	0.575
L4FC	0.933	0.946	0.934	0.930
L3FC	0.827	0.863	0.824	0.822
L34FC	0.68	0.725	0.680	0.677
L234FC	0.587	0.627	0.592	0.590

vamente. Esses modelos também apresentaram altas *precision, recall* e *F1-Score*, indicando uma boa capacidade de classificação correta dos pares de imagens. Por outro lado, os modelos *fine-tuning*, L34, L234 e L34FC tiveram desempenhos inferiores, com valores mais baixos em todas as métricas. Esses resultados evidenciam a importância do treinamento específico dos modelos e destacam a eficácia dos modelos L4, *zero-shot*, L4FC e L3.

A Tabela 15 apresenta as métricas de desempenho dos modelos de reconhecimento facial avaliados com o conjunto de dados Família Teste utilizando o algoritmo KNN. Os modelos que obtiveram os melhores resultados foram o *fine-tuning*, Mega Face + Família e L4, com altos valores de *accuracy* de 0.9 para cada um. Além disso, esses modelos também demonstraram uma capacidade consistente de classificação correta, evidenciada pelos altos valores de *precision, recall* e *F1-Score*. Por outro lado, os modelos *zero-shot*, L4ZS, L3, L234, L4FC, L3FC e L234FC tiveram desempenhos inferiores, com valores mais baixos em todas as métricas. Esses resultados reforçam a eficácia dos modelos *fine-tuning*, Mega Face + Família e L4.

A Tabela 16 apresenta as métricas de desempenho dos modelos de reconhecimento facial avaliados com o conjunto de dados Mega Face (Facescrub) usando o algoritmo KNN. Observa-se que o modelo *zero-shot* obteve resultados perfeitos em todas as métricas, com *accuracy*, *precision*, *recall* e *F1-Score* igual a 1.0. Os modelos *fine-tuning*, L4, L3, L34, L234, L4FC, L3FC, L34FC e L234FC apresentaram resultados variados em relação às métricas. O modelo L4ZS obteve altos valores de *accuracy*, *precision*, *recall* e *F1-Score*, próximos de 1.0. Já os modelos *fine-tuning*, L3, L34, L234, L4FC, L3FC, L34FC e L234FC mostraram desempenhos um pouco inferiores, mas ainda alcançaram resultados respeitáveis nas métricas avaliadas.

Tabela 15: Métricas dos modelos de reconhecimento submetidos ao KNN com Família Teste

Modelo	Accuracy	Precision	Recall	F1-Score
zero-shot	0.700	0.600	0.625	0.560
fine-tuning	0.900	0.917	0.917	0.900
Mega Face + Família	0.900	0.917	0.937	0.915
L4ZS	0.700	0.600	0.625	0.560
L4	0.900	0.667	0.7500	0.700
L3	0.700	0.583	0.563	0.564
L34	0.800	0.604	0.686	0.638
L234	0.800	0.563	0.625	0.590
L4FC	0.800	0.625	0.686	0.631
L3FC	0.700	0.600	0.625	0.560
L34FC	0.900	0.917	0.938	0.914
L234FC	0.700	0.617	0.542	0.556

Tabela 16: Métricas dos modelos de reconhecimento submetidos ao KNN com Mega Face *(Facescrub)*

Modelo	Accuracy	Precision	Recall	F1-Score
zero-shot	1.000	1.000	1.000	1.000
fine-tuning	0.694	0.703	0.661	0.672
Mega Face + Família	1.000	1.000	1.000	1.000
L4ZS	0.976	0.982	0.981	0.981
L4	0.929	0.928	0.917	0.920
L3	0.776	0.799	0.738	0.729
L34	0.729	0.769	0.693	0.700
L234	0.776	0.796	0.742	0.752
L4FC	0.882	0.884	0.872	0.873
L3FC	0.776	0.783	0.747	0.754
L34FC	0.694	0.754	0.632	0.635
L234FC	0.718	0.718	0.682	0.688

As Tabelas 14, 15 e 16 retrataram o cenário dos experimentos, a consistência do modelo *zero-shot* e do Mega Face + Família estão presentes nas 3 tabelas, enquanto os outros modelos apresentam desvios nos números e inconstâncias nos resultados.

Somando aos demais resultados anteriores, pode-se confirmar o resultado final aplicado na API. E conclui-se que fazer o fine-tuning em modelos *zero-shot* pode não ser uma abordagem válida em alguns casos, onde o conjunto de dados alvo é parecido com o original.

4.4 API

Na prática, confirmou-se que o modelo *zero-shot* e o modelo Mega Face + Família são os modelos que tiveram os melhores resultados assertivos na API. Os demais modelos apresentaram inconsistências na definição das classes.

As identidades que possuíam menos imagens foram as que mais tiveram equívocos por parte dos modelos. Na Figura 14 estão alguns exemplos de faces detectadas (Figura 8) e reconhecidas o *dataset* Família com o KNN.

Essas imagens submetidas no teste na API não foram utilizadas nos treinos dos modelos, o resultado foi satisfatório em reconhecer a maioria das identidades submetidas e o objetivo da API foi alcançado. Portanto, com base nos resultados obtidos, optou-se por utilizar o modelo *zero-shot* na versão final da API.

É importante ressaltar que, devido à abordagem proposta de ser útil em aplicações com pessoas não famosas e com poucas imagens de treinamento e teste, a aplicação foi desenvolvida uti-

Figura 14: Montagem das imagens com as pessoas identificadas

lizando o algoritmo KNN com pessoas da família do autor. Portanto, as imagens submetidas ao processo de reconhecimento devem estar presentes no dataset Família.

5 Conclusão

O método do Retina Face se mostrou consistente e performou bem, dentro das limitações citadas, de modo geral cumpriu seu propósito de maneira eficiente e contribuiu muito para o desenvolvimento da API.

Os resultados experimentais demonstraram que alguns modelos com *fine-tuning* obtiveram bons números em desempenho, porém, com grandes inconsistências, o que tornou o modelo ineficiente. Na prática, constatou-se que os modelos que melhor se destacaram e funcionaram efetivamente na API foram os modelos *zero-shot* e Mega Face + Família, aquele sendo o modelo selecionado para a versão final da API, devido aos custos de treinamento do modelo Mega Face + Família.

Ademais, constatou-se que o desempenho do modelo após o *fine-tuning* pode ser considerado inferior ao do modelo *zero-shot* em diversos cenários. Portanto, ao considerar a aplicação de *fine-tuning* em modelos *zero-shot*, é essencial realizar uma avaliação minuciosa da natureza do conjunto de dados em questão e das metas específicas do projeto. Caso o conjunto de dados apresente variações significativas em relação ao conjunto de treinamento original, o *fine-tuning* pode ser uma estratégia apropriada para aprimorar a adaptabilidade do modelo. Entretanto, se o conjunto de dados for semelhante ao conjunto de treinamento original e o modelo *zero-shot* já apresentar um desempenho satisfatório, pode não ser necessário realizar o processo de *fine-tuning*.

Essa conclusão ressalta, ainda, a importância de avaliar os resultados não apenas com base nos números dos experimentos, mas também considerando a aplicabilidade prática da solução. O sucesso da API em reconhecer as identidades, valida a escolha dos modelos e abordagens adotados, demonstrando a eficácia do projeto como um todo.

Todos os objetivos do projeto foram alcançados, permitindo que a API desempenhe suas funcionalidades de forma eficiente e satisfatória.

Agradecimentos

Gostaria de agradecer a todas as pessoas que colaboraram com este projeto, desde o apoio fundamental da minha companheira Carla, noites e fim de semanas perdidos, investidos nesse projeto. Aos meus pais pelo suporte de conseguir chegar até este momento da vida e ao meu orientador professor Takashi, por todo o conhecimento passado, paciência e esforço nesse processo.

Referências

[1] Deng, J., Guo, J., Zafeiriou, S. (2019). "Arcface: Additive angular margin loss for deep face recognition."In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4690-4699).

[2] Deng, J., Guo, J., Xue, N. (2020). "Retinaface: Single-stage dense face localisation in the wild."In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5203-5212).

[3] Szeliski, R. (2010). "Computer vision: algorithms and applications."Springer Science e Business Media.

[4] Zhang, K., Zhang, Z., Li, Z., Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Processing Letters, 23(10), 1499-1503.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016

[6] R. Girshick. Fast r-cnn. In ICCV, 2015. 1, 3

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In ECCV, 2016.

[8] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S3fd: Single shot scale-invariant face detector. In ICCV, 2017.

[9] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wider face: A face detection benchmark. In CVPR, 2016.

[10] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition." in BMVC, 2015.

[11] J. Deng, Y. Zhou, and S. Zafeiriou, "Marginal loss for deep face recognition," in CVPR Workshop, 2017.

[12] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in CVPR, 2016.

[13] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep., 2007.

[14] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in CVPR, 2011

[15] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in WACV, 2016.

[16] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying crosspose face recognition in unconstrained environments," Tech. Rep., 2018.

[17] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: The first manually collected in-the-wild age database," in CVPR Workshop, 2017.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980, 2014.

[19] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and ´S. Belongie. Feature pyramid networks for object detection. In CVPR, 2017

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal ´loss for dense object detection. In ICCV, 2017

[21] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis. Ssh: Single stage headless face detector. In ICCV, 2017.

[22] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in CVPR, 2017.

[23] H. Wang, Y. Wang, Z. Zhou, X. Ji, Z. Li, D. Gong, J. Zhou, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in CVPR, 2018.

[24] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range loss for deep face recognition with long-tail," in ICCV, 2017.

[25] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in ECCV, 2016.

[26] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," arXiv:1701.06548, 2017.

[27] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in FG, 2018.

[28] X. An, J. Deng, J. Guo, Z. Feng, X. Zhu, J. Yang, and T. Liu, "Killing two birds with one stone: Efficient and robust training of face recognition cnns by partial FC," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. IEEE, 2022, pp. 4032–4041.

[29] Hang Du, Hailin Shi, Yuchi Liu, Jun Wang, Zhen Lei, Dan Zeng, and Tao Mei. Semi-siamese training for shallow face learning. In ECCV, 2020.

[30] Xiang An, Xuhan Zhu, Yuan Gao, Yang Xiao, Yongle Zhao, Ziyong Feng, Lan Wu, Bin Qin, Ming Zhang, Debing Zhang et al., "Partial fc: Training 10 million identities on a single machine", Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1445-1449, 2021.

[31] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild,"in Proceedings of International Conference on Computer Vision (ICCV), Dec. 2015.

[32] Chen, D., Cao, X., Wen, F., Sun, J. (2014). Blessing of Dimensionality: High-dimensional Feature and Its Efficient Compression for Face Verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3025-3032).

[33] Zhang, Tianjiao, et al. "Face detection: A survey."Journal of Computer Science and Technology 35.2 (2020): 187-210.

[34] Müller, Andreas C., and Sarah Guido. Introduction to machine learning with Python: a guide for data scientists. O'Reilly Media, Inc., 2016.

[35] SABINA POKHREL,How has Face Recognition Advanced since the 1960s?, 2020. Disponível em: https://xailient.com/blog/how-has-face-recognition-advanced-since-the-1960s/. Acesso em 05 de junho de 2023.

[36] Roth, K., Milbich, T., Sinha, S., Gupta, P., Ommer, B., & Cohen, J. P. (2020, November). Revisiting training strategies and generalization performance in deep metric learning. In International Conference on Machine Learning (pp. 8242-8252). PMLR.

[37] Li, Z. ; Hoiem, D. ; Learning Without Forgetting, B. Leibe et al. (Eds.): ECCV 2016, Part IV, LNCS 9908, Springer International Publishing AG 2016, pp. 614 - 629, DOI: 10.1007/978-3- 319-46493-0 37

[38] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe. Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441, 2014.

[39] Deng, Jiankang. RetinaFace: Single-stage Dense Face Localisation in the Wild. Papers With Code, 2019 Disponível em https://paperswithcode.com/paper/190500641. Acesso em: 05 de junho de 2023.

[40] Mukherjee, Suvadity. The Annotated ResNet-50. Towards Data Science, 2022 Disponível em https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758. Acesso em 26 de junho de 2023.

[41] Deng, Jiankang. ArcFace: Additive Angular Margin Loss for Deep Face Recognition, 2019. Disponível em https://paperswithcode.com/paper/arcface-additive-angular-margin-loss-for-deep. Acesso em 05 de junho de 2023.

[42] Kanstrén, Teemu. A Look at Precision, Recall, and F1-Score, 2020. Disponível em https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec. Acesso em 26 de junho de 2023.