Avaliação Experimental de Métodos de Quantização em Reconhecimento Facial *

Carlos Monteiro e **Edson Takashi Matsubara**
FACOM/UFMS
Campo Grande-MS, Brasil
carloshenriquemonteirom@gmail.com

***Abstract*—Este trabalho realiza uma avaliação experimental dos impactos causados pela quantização de modelos de geração de embeddings aplicados ao reconhecimento facial. São comparados dois modelos — FaceNet (baseado em Inception-ResNet) e TransFace (baseado em Vision Transformer) — sob diferentes representações de precisão (FP32 e INT8) e plataformas de inferência (Torch e ONNX). Os experimentos foram conduzidos nos datasets LFW, VGGFace2 e CelebA, com avaliação de acurácia Rank-1 e uso de distância cosseno como métrica de similaridade. Os resultados indicam que modelos mais estabelecidos, como o FaceNet, são mais robustos à quantização, mantendo a acurácia mesmo em formatos reduzidos, enquanto o TransFace sofre degradação perceptível ao ser quantizado. Além disso, a quantização dos vetores resultou em redução de até 80% no espaço de armazenamento sem comprometer significativamente o desempenho. Estes achados destacam a viabilidade de quantização como estratégia para otimização de modelos em ambientes com restrições de recursos.**

Keywords*—*Facenet, Transface, Quantização, LFW, VGGFace2, CelebA, Torch, ONNX

I. Introdução

Reconhecimento facial utilizando algoritmos de aprendizado de métrica estão presentes em diversas aplicações do cotidiano. Aeroportos para autenticar check-in e embarque [2], para venda de ingressos e entrada em estádios de futebol [3], estabelecimentos comerciais para segurança [13] e outros lugares [11] [19] são alguns exemplos. Considerando o cenário brasileiro, em que aeroportos já começam a utilizar ferramentas de reconhecimento, se tal aplicação for implementada em um aeroporto como o Galeão, em que é estimado passagem de 30 mil pessoas por dia [9], estimando o uso de 120MB por dia, para check in e embarque, este valor corresponde a 43.8GB por ano, porém tal valor é para apenas um aeroporto, esse número pode aumentar muito, dado que, ferramentas de IA para reconhecimento de pessoas tiveram crescimento em seu uso, e é estimado que o número de aplicações que as utilize continue a aumentar. Essa tendência de crescimento do uso de sistemas de reconhecimento facial pode gerar alguns desafios que precisam ser abordados, o primeiro problema, em sistemas que guardam registros das faces que foram processadas por modelos de geração de embeddings, como sistemas de segurança mostrados em [13], é o grande volume de dados que precisa ser armazenado para ter registro dos indivíduos que frequentaram um espaço, o segundo problema a ser observado é a precisão da capacidade de um modelo de gerar vetores de características que permitem o reconhecimento de uma pessoa, principalmente em casos que necessitam de uma alta taxa de acerto, como é o esperado de um sistema de autenticação de check-in em aeroportos [2]. O problema de armazenar embeddings das pessoas que passaram por programas de reconhecimento facial pode ser mitigado ao mudar o tipo de dado utilizado para guardar os valores dos vetores de características de face. O formato de dado original gerado pelas redes é float com 32 bits, ao mudar para int com 8 bits, espera-se reduzir de maneira considerável o espaço em disco utilizado. Quanto ao segundo problema apresentado, a solução a ser verificada é a aplicação de modelos que utilizam de métodos mais novos, como o Transface que utiliza visual transformers em sua arquitetura. Para este problema também será aplicada a quantização e conversão nos modelos de teste.

A proposta deste trabalho consiste em avaliar a redução de desempenho em algoritmos de aprendizado de métrica ao aplicar métodos de quantização. Para avaliar este problema, foi utilizado o domínio de reconhecimento facial.

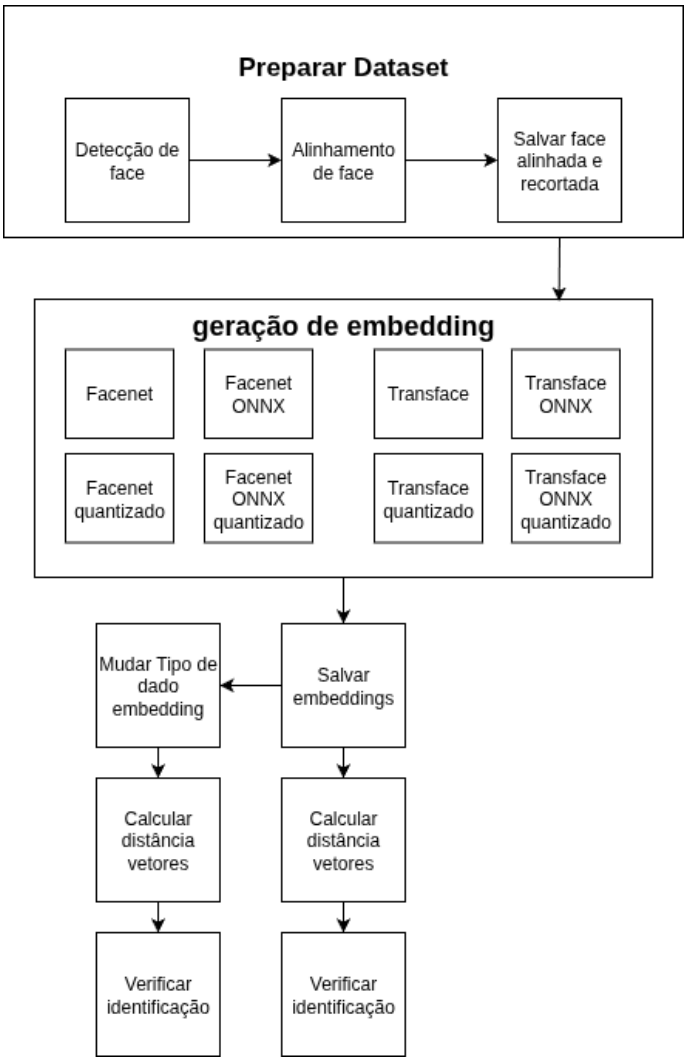
A principal contribuição é a avaliação destes resultados que indicam impactos distintos em plataformas diferentes e também nos algoritmos avaliados que são descritos na conclusão deste trabalho.

*

II. Metodologia Proposta

Os testes foram realizados seguindo o fluxo de execução apresentado na Figura I, que mostra a sequência de operações realizadas para a preparação dos dados a serem utilizados nos testes e a verificação da acurácia do modelo em reconhecer indivíduos.

Figura I: Metodologia de avaliação



1. Preparação do dataset
2. Conversão do modelo
3. Quantização(Em testes com modelo quantizado)
4. Geração de embeddings
5. Salvar embeddings
6. Cálculo de distância
7. Verificar acurácia Rank-1

Preparar dataset: Consiste em um pré-processamento das imagens de cada dataset no qual é feito detecção de face e alinhamento da mesma e então a salvar como um novo arquivo, para garantir que os modelos tenham as mesmas condições iniciais das características de cada imagem.

Conversão: Os modelos originalmente disponíveis para o motor de inferência Torch foram, em primeiro lugar convertidos para ONNX, a fim de ter variabilidade na forma como a plataforma de testes trata o modelo, e por ser um formato que permite conversão a outros motores de inferência.

Quantização: Para a avaliação do modelo quantizado, esse passo foi adicionado a estrutura de execução para que os testes fossem feitos também com versões quantizadas dos modelos.

Geração de embeddings: Com o modelo a ser testado pronto para uso, cada imagem presente no dataset foi então processada pelo modelo, que gera como resultado um vetor de características da face utilizada como entrada.

Salvar embeddings: Após os modelos gerarem os embeddings das faces do dataset, esses vetores de características foram então salvos em arquivos para serem posteriormente analisados.

Cálculo de distância: Para cada vetor presente no arquivo de embeddings, foi calculada a distância entre cada par de vetores e identificado aquele com menor valor, assim como feito para o método de avaliação Rank-1, então foi registrado qual o arquivo de origem para os vetores par de menor distância.

Verificar identificação: De acordo com cada dataset, foi verificado se as faces correspondentes ao vetores de menor distância correspondem à mesma pessoa e calculado a taxa de acerto para cada dataset.

III. Modelos

Para a realização dos testes, foram utilizadas duas versões das redes apresentadas. Para a rede Facenet [14], foi utilizada uma implementação em pytorch disponibilizada em um repositório github timesler/facenet-pytorch [18] pré-treinada sobre o dataset VGGFace2, enquanto para o modelo Transface foi utilizada como base a implementação de DanJun6737/TransFace [6] também disponibilizada no github.

A. Facenet

A rede Facenet [14] aplicada aos testes corresponde a uma implementação de uma arquitetura backbone Inception-ResNet [17], que é uma combinação de características de redes inception com redes residuais, nos testes realizados, foi utilizada a variante NN3, que possui arquitetura idêntica a NN2, porém com entrada 160x160 de uma imagem alinhada de face para gerar um vetor de dimensão 512 representando os embeddings da face. Informações da rede podem ser vistas na Tabela VI,que mostra as camadas da rede, a profundidade das camadas, filtros, número de parâmetros e operações de ponto flutuante.

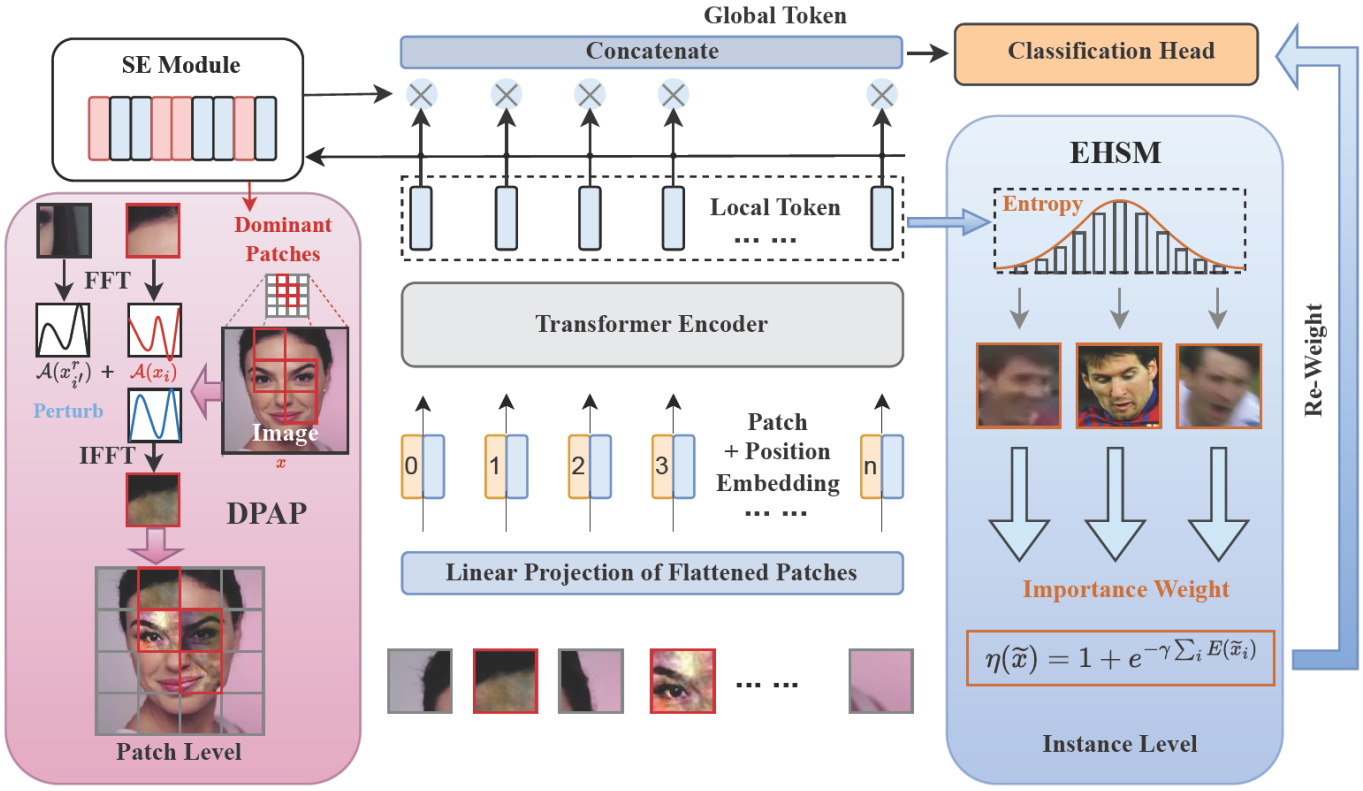
B. TransFace

Modelo baseado na arquitetura de Vision Transformer(ViT) [8], apresentando poucas mudanças em sua estrutura e utilização de um backbone Transformer Encoder seguido de um módulo de Squeeze-and-Excitation [10], tem como entrada uma imagem de face alinhada com tamanho 112x112 e gera como saída um vetor de tamanho 512, algumas das características que podem ser vistas na Figura II, que mostra os principais módulos da arquitetura como Transformer Encoder, DPAP e Classification Head, para os testes realizados foi utilizada a versão S do modelo que se caracteriza por ter 12 camadas de profundidade e 6 cabeças de atenção, assim como os pesos pré-treinados no dataset Glint360K [1].

IV. Datasets

Os datasets empregados nos experimentos foram o LFW [16], VGGFace2 [4] e CelebA [12]. Cada dataset passou por um processo de detecção de face e alinhamento de face para estabilizar as características que seriam coletadas por cada modelo. O dataset LFW [16] possui cerca de 13K imagens distribuídas em 5750 pessoas, sendo o único dataset que possui pessoas que possuem apenas uma imagem de face. Sendo assim há imagens que não possuem par, sua estrutura segue o formato de pasta com nome da pessoa e as imagens da pessoa em seu interior, essa mesma lógica de nome de pasta e imagens se aplica ao dataset VGGFace2, porém ele difere em quantidade e dispersão de imagens, tendo 170K imagens distribuídas em 480 pessoas, sendo o dataset que possui maior relação imagens por pessoa. Para o ultimo dataset, o CelebA possui estrutura de arquivos distinta dos outros, para esse cada arquivo de imagem tem um nome único que pode ser localizado em um arquivo que relaciona nome de imagem e identificação da pessoa, para a os conjuntos de dados mencionados, a Tabela I condensa as informações mencionadas na coluna **Identidades**, que é a quantidade de pessoas distintas no conjunto, **Imagens**, sendo a quantidade de imagens presentes no dataset e **Img/Id**, que é a razão entre quantidade de imagens e quantidade de identidades no dataset.

Figura II: Transface architecture. Imagem de TransFace: Calibrating Transformer Training for Face Recognition from a Data-Centric Perspective [5]



A. *Preparação*

Para preparar os datasets, foi utilizado o modelo de face detection e keypoints Retinaface [7], com score mínimo de 0.2 para detecção e, em caso de mais de uma face detectada foi considerada a face de maior score, em seguida, a face foi alinhada horizontalmente antes de ser salva com o mesmo nome e estrutura de arquivo de cada dataset.

Tabela I: Datasets

Dataset	Identidades	Imagens	Img/Id
LFW	5750	13K	2.26
VGGFace2	480	170K	354.16
CelebA	10177	202K	19.84

V. Métodos

A seguir estão presentes os métodos utilizados para converter, quantizar os modelos, comparar os embeddings e mudar a representação de embeddings.

A. *Plataforma e bibliotecas*

A plataforma que que os testes foram desenvolvidos é formada pelos seguintes itens de hardware e software, no Apêndice A, estão listadas as bibliotecas python que foram utilizas e suas versões, como *pandas=2.2.3*, *numpy=1.26.4* e *scipy=1.15.3*.

- CPU AMD Ryzen 2600X
- GPU NVIDIA RTX 4070
- RAM 32GB

- SO Ubuntu 22.04
- python 3.13.2
- Nvidia Driver 550.163.01
- CUDA 11.5

B. Conversão

Uma das principais etapas do processo foi a conversão do modelo incialmente em torch para ONNX, sendo feita antes de quantizar os modelos, para tal foi utilizada a biblioteca *torch.onnx.export*, que utiliza como parâmetros o modelo carregado, uma entrada no formato esperado do modelo, o nome do arquivo de saída, nome da entrada e saída a serem utilizadas e *opeset_ version*, no caso utilizada a 13, deixando a função de conversão da seguinte forma

```
    torch.onnx.export(
        model,
        dummy_input,
        "Transface.onnx",
        input_names=["input"],
        output_names=["embedding"],
        dynamic_axes={
            "input": {0: "batch"},
            "embedding": {0: "batch"}
        },
        opset_version=13,
        verbose=True
    )
```

C. Quantização

A quantização dos modelos foi aplicada de duas formas distintas, para os modelos em formato onnx foi aplicada quantização estática, enquanto para os modelos em pytorch foi aplicada quantização dinâmica. Estes modos diferem em seu funcionamento e método de uso, a quantização estática é feita uma vez antes de utilizar o modelo, em que fica salvo o arquivo do modelo em versão quantizada, enquanto a quantização dinâmica é realizada a cada execução do modelo.

1. ONNX quantization

Para quantizar o modelo em ONNX, os seguintes métodos da biblioteca *onnxruntime.quantization* foram utilizados.

- *quantize_ static*
- *CalibrationDataReader*
- *QuantType*

os quais foram utilizados da seguinte forma, função de quantização do modelo, nome de entrada do modelo, nome de saída do modelo, dataloader para imagens de calibração, no caso utilizado o dataset vgg e o tipo de quantização a ser aplicado, no caso Int8.

```
quantize_static(
    "TransfaceSmall-512.onnx",
    "TransfaceSmall-512-quantized.onnx",
    calibration_data_reader=calibration_data,
    quant_format=QuantType.QInt8,
    activation_type=QuantType.QInt8,
    weight_type=QuantType.QInt8,
)
```

2. Torch quantization

A quantização dinâmica foi aplicada a cada execução do modelo utilizando a função *quantize_ dynamic* da biblioteca *torch.quantization*, sendo utilizada da seguinte forma, em que é passado para a função o modelo original, as layers em que a quantização será aplicada, no caso as layers de operação Linear e LSTM e o data type da quantização, neste caso, INT8.

```
model = tq.quantize_dynamic(
                model,
                {torch.nn.Linear,torch.nn.LSTM},
                dtype=torch.qint8
            )
```

D. Distância

Para verificar a distância foi utilizada a função de cosine distance, da biblioteca spatial do scipy [15], representada pela Equação (1) em que é calculado a distância entre dois vetores **A** e **B**, com resultado 0 para vetores idênticos. Ao utilizar esse método, é possível verificar de maneira objetiva quais pares de embeddings gerados têm maior semelhança.

$$\text{cosine\_distance} = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

```
from scipy.spatial import distance
cosDist = distance.cosine(array_A, array_B)
```

E. Cálculo de acurácia

Para verificar a capacidade de reconhecimento dos modelos, foi utilizado o método de avaliação Rank-1. Este método consiste na geração de embeddings para todas as imagens presentes no dataset a ser avaliados. Cada vetor de características foi então comparado utilizando o método de cálculo de distância. Aquele que resultasse na menor distância seria então marcado como face mais semelhante para então ser verificado se tal par corresponde à mesma pessoa. A taxa de acerto foi então obtida pela divisão da quantidade de faces identificadas como sendo da mesma pessoa pelo total de faces no dataset, como exemplificado na Equação (2).

$$\text{acuracia} = \frac{IdentificacoesCorretas}{FacesNoDataset} \tag{2}$$

F. Representação

A mudança na representação de como os dados são representados em arquivo consiste na mudança do tipo de dado empregado para representar as faces das quais as características foram coletadas. Para a mudança no formato dos dados, foi utilizada uma função simples de cast, para cada índice do vetor de características, no qual o valor foi multiplicado por 127 para ficar dentro do range de valores para int 8.

```
quantized_list = []
for x in embedding_list:
    q = int(round(x * 127))
    quantized_list.append(np.int8(q))
```

VI. Resultados

A avaliação experimental tem como foco responder às seguintes perguntas:

RQ1 : Qual é a redução em termos de taxa de acerto ao realizar a quantização?

RQ2 : Existe diferença entre usar a quantização do Torch e a quantização do ONNX?

RQ3 : Considerando Transface e o Facenet, que são algoritmos populares de reconhecimento facial, qual deles é mais sensível ao método de quantização?

As seções seguintes estão estruturadas para responder a estas perguntas.

A. Torch x ONNX

A conversão do modelo foi o primeiro passo dos testes realizados, dessa forma, os dados sobre conversão cobrem somente os resultados referentes ao valores sem quantização dos modelos. Ao analisar em pares as linhas *Torch* e *ONNX*, da Tabela II e da Tabela III, pode-se verificar que os resultados apresentados para o Facenet presentes na Tabela II, para o dataset LFW ao comparar as linhas 1 e 2, percebe-se aumento de 4% de acurácia em ambos os formatos de dado, valor que se repete para o conjunto VGGFace2 nas linhas 3 e 4, também com 4% de aumento de acurácia ao converter o modelo, enquanto para o dataset CelebA, presente nas linhas 5 e 6 da Tabela II, houve um aumento de 8% ao converter o modelo de Torch para ONNX. Para o modelo Transface, as informações presentes na Tabela III, mostram que ao converter o modelo, nos conjuntos de dados testado a acurácia foi reduzida, para o dataset LFW, presente nas linhas 1 e 2, a redução na acurácia foi de 1%, enquanto para os datasets VGGFace2 e CelebA a redução foi de 2%, como pode ser visto nas linhas 3 a 6 da Tabela III.

B. Quantização

A quantização dos modelos foi feita com os modelos originais e com os modelos após a conversão para ONNX, então os impactos da quantização são somados a quaisquer alterações provenientes da conversão do modelo. Como resultado para a quantização dos modelos testados, pode-se verificar as colunas **FP32** e **INT8**, que mostram a acurácia para cada combinação de modelo e engine da Tabela II e da TabelaIII, que mostram que para o modelo Facenet, ao verificar a Tabela II, percebe-se que há uma redução máxima de 0.17% no conjunto LFW quando em Torch, e aumento de 0.07% quando testado sobre o mesmo dataset, porém com motor de inferência ONNX, de modo que tais diferenças de valores não representa uma mudança significativa na acurácia do modelo. Porém, ao verificar o modelo Transface, presente na Tabela III, observa-se um comportamento diferente, a quantização gera mudanças que podem ser consideradas perceptíveis à acurácia do modelo. Na linha 2, houve redução de 3% na taxa de acerto, a mesma redução pode ser vista na linha 6, com o dataset CelebA que também perdeu 3% de precisão, enquanto para o dataset VGGFace2, na linha 4, a redução da taxa de acerto do modelo foi de 4%. Estes resultados se apresentam todos para o motor de inferência ONNX, para as execuções em Torch, a variação máxima foi de 0.1%, para mais ou para menos. O processo de quantização também mostra outro aspecto, a redução do tamanho do modelo, que pode ser visto na Figura III, que mostra o tamanho de cada modelo antes e após a quantização, evidenciando que a quantização principalmente quando feita em ONNX, consegue reduzir bastante o tamanho do modelo, no Facenet reduxiu de 94 MB para 23.9 MB e no Transface reduz de 349 MB para 89.3 MB.

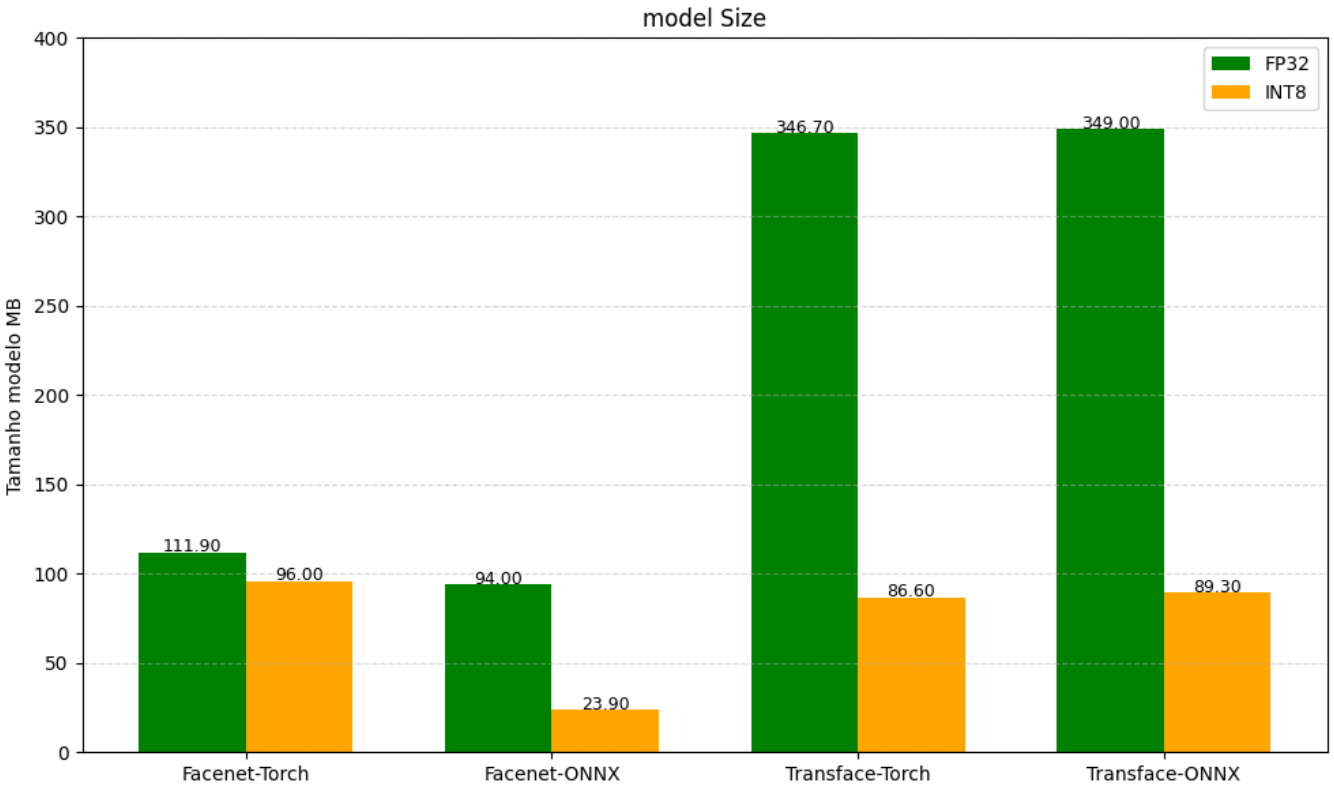
Tabela II: Resultados testes Facenet

	dataset	**engine**	**FP32**	**INT8**
1	LFW	Torch	61.41%	61.27%
2	LFW	ONNX	65.47%	65.54%
3	VGGFace2	Torch	92.62%	92.61%
4	VGGFace2	ONNX	96.66%	96.65%
5	CelebA	Torch	80.06%	80.03%
6	CelebA	ONNX	88.65%	88.59%

Tabela III: Resultados testes Transface

	dataset	**engine**	**FP32**	**INT8**
1	LFW	Torch	68.49%	68.48%
2	LFW	ONNX	67.15%	64.3%
3	VGGFace2	Torch	99.26%	99.27%
4	VGGFace2	ONNX	97.65%	93.62%
5	CelebA	Torch	95.81%	95.81%
6	CelebA	ONNX	93.19%	90.07%

Figura III: Tamanho dos modelos



C. *Facenet x Transface*

Ao comparar os modelos, podemos separá-los por dataset e analisar os resultados de forma individual, para os resultados do dataset LFW apresentados na Figura V, tem-se que para a execução em Torch em FP32 e INT8, o Transface apresenta acurácia superior ao Facenet, cerca de 7%, enquanto ao observar em ONNX, tal diferença em FP32 é reduzida para 2% ainda com o Transface sendo melhor, porém para INT8 o Facenet passa a ser melhor, com acurácia 1% superior. No conjunto de testes do dataset VGGFace2, resultados semelhantes ao LFW podem ser encontrados para Torch FP32 e INT8, com acurácia do Transface sendo 7% superior ao Facenet, enquanto ao verificar os testes com motor de inferência ONNX, em FP32 o Transface se apresenta superior em 1%, enquanto em INT8 a diferença aumenta para o Facenet, com 3% de vantagem, como mostra a Figura VI. Para o conjunto de faces do dataset CelebA, o Transface apresentou os melhores resultados em relação ao Facenet, como pode-se verificar na Figura VII, com 15% de diferença ao testar com Torch em FP32 e INT8. Neste caso para o motor de inferência ONNX, o Transface se manteve com acurácia maior que o Facenet em FP32 e INT8, sendo superior em 4.5% e 1.5% respectivamente.

D. *Computação espaço de armazenamento*

Na representação dos dados pode-se observar uma redução considerável no volume de disco utilizado para armazenar cada registro de embedding como pode ser visto nos valores em MB da Tabela IV, representando uma redução do uso de espaço de disco foi próximo a 19% do tamanho original, exceto pelo dataset VGGFace2 em que o volume final após a compressão foi de 24.7% do valor original, o que pode ser verificado pelo gráfico de comparação percentual do uso de disco na Figura IV, sem ter quedas na capacidade de reconhecimento dos modelos ao utilizar valores em INT, como pode ser visto na Tabela V, em que mostra o acerto com o dado em formato Float e Int para cada modelo e dataset.

Tabela IV: Tamanho de arquivos de embeding

Dataset	**FP32(MB)**	**INT8(MB)**
LFW	136	26
VGGFace2	1369	338
CelebA	2073	401

Tabela V: Acurácia valores em INT

Motor	Modelo	LFW	VGGFace2	CelebA
Torch	FacenetFP32	61.3%	92.6%	80.0%
Torch	FacenetINT8	61.3%	92.6%	80.0%
ONNX	FacenetFP32	65.4%	96.6%	88.5%
ONNX	FacenetINT8	65.4%	96.6%	88.5%
Torch	TransfaceFP32	68.4%	99.2%	95.8%
Torch	TransfaceINT8	68.4%	99.2%	95.8%
ONNX	TransfaceFP32	67.1%	97.6%	93.1%
ONNX	TransfaceINT8	64.2%	93.3%	90.0%

VII. Discussão

A partir dos resultados apresentados, é possível verificar a capacidade de reconhecimento dos modelos avaliada experimentalmente com os resultados descritos nos artigos de referência. Para o modelo Facenet, foi obtida acurácia máxima de 92.62% em torch e de 96.66% em ONNX, ambos em FP32 para o dataset VGGFace2, o que supera a acurácia apresentada originalmente no artigo. Para o modelo transface o artigo apresenta acurácia máxima de 99.85%, enquanto nos testes realizados foi obtido acerto máximo de 99.27% e 95.81% para os datasets VGGFace2 e CelebA, ambos em torch. Com os resultados apresentados, podemos verificar que, para melhorar a acurácia no processo de identificação de faces, a melhor abordagem se mostra a utilização de um modelo que utilize métodos mais refinados, porém a conversão e quantização de tais modelos não se mostra tão eficientes quanto aplicada a modelos mais antigos, que apresentam aumento na taxa de acerto, o que pode estar mais relacionado a um melhor entendimento do funcionamento e características das layers de redes mais estabelecidas. Dos resultados também é possível verificar que a abordagem de modificar a forma com que os valores são representados se mostra muito eficiente em reduzir o espaço de disco utilizado para guardar os vetores de características, de modo que os resultados de reconhecimento não são afetados. No cenário do aeroporto do Galeão, os 43.8 GB estimados, se tornariam 8.73 GB após mudança no formato dos dados. Estes resultados são esperados pela redução no número de bits utilizados para representar cada valor, de 32 bits para 8 bits, e a saída da rede ser limitada de -1 a 1, passa a ser limitada de -127 a 127.

Figura IV: Tamanhos de arquivos

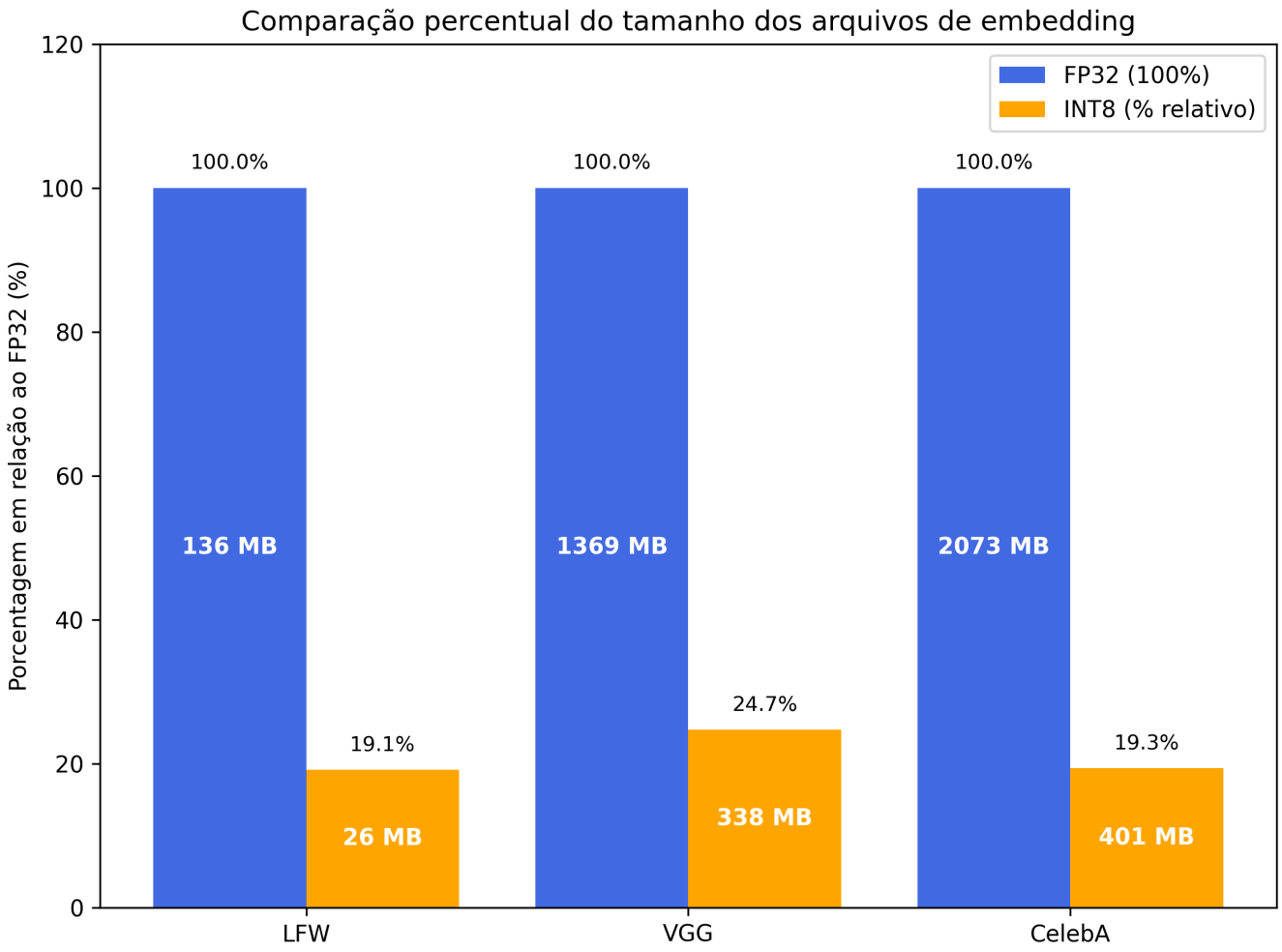


Figura V: Comparativo de acerto LFW

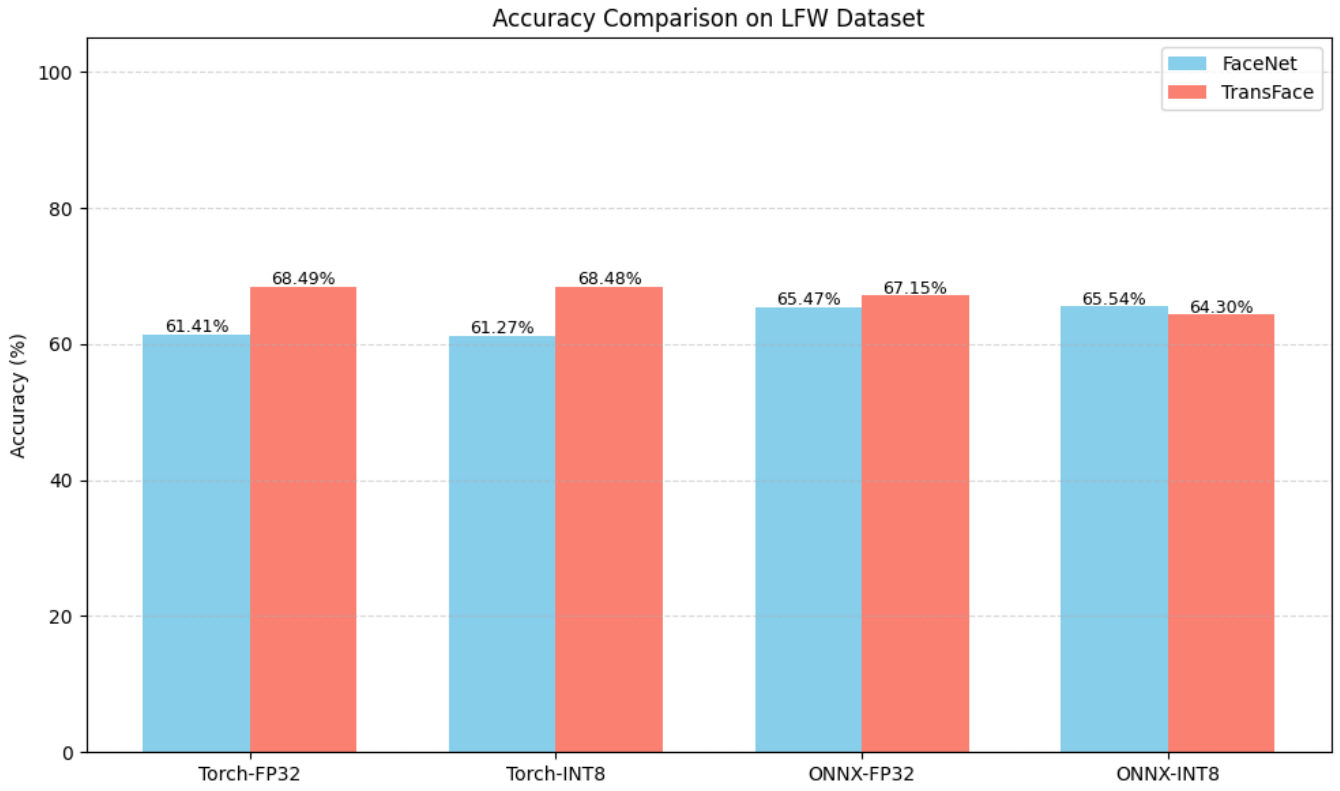


Figura VI: Comparativo de acerto VGGFace2

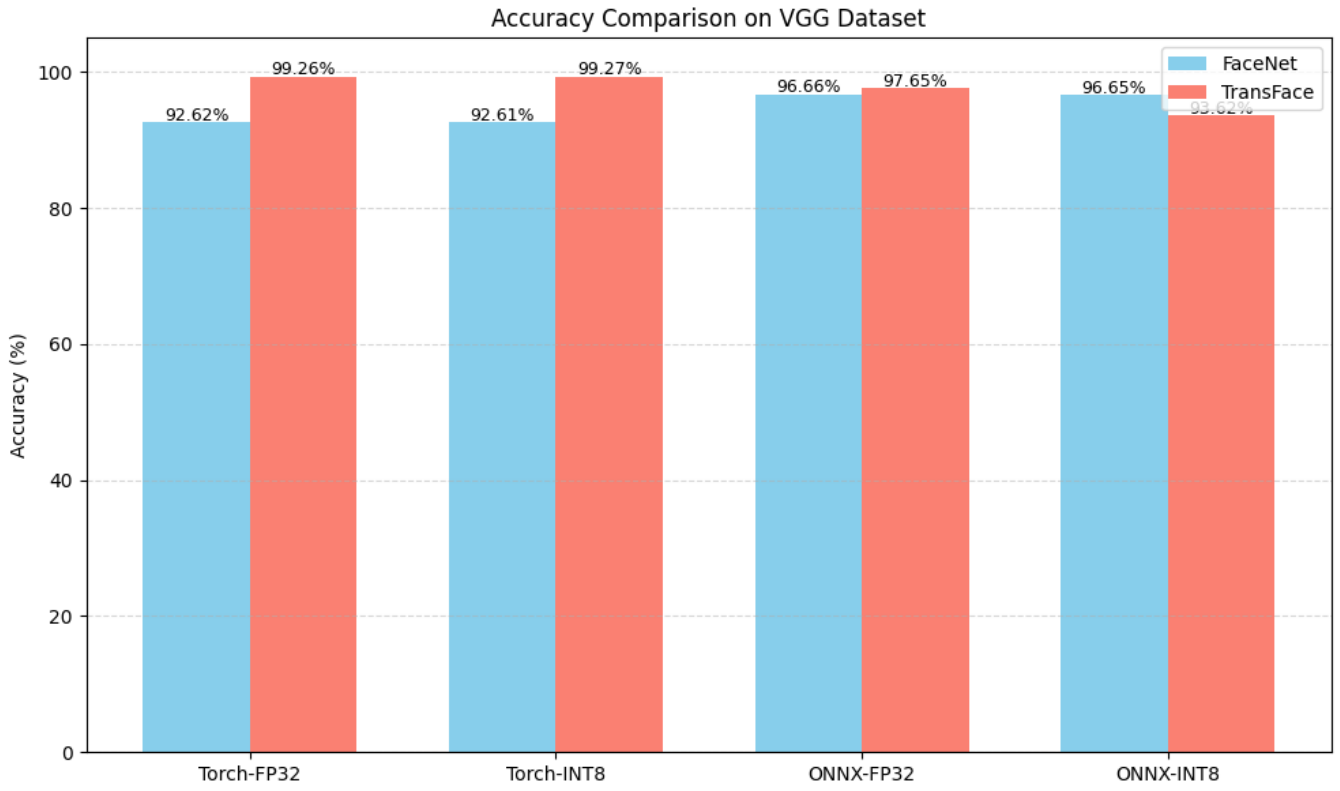
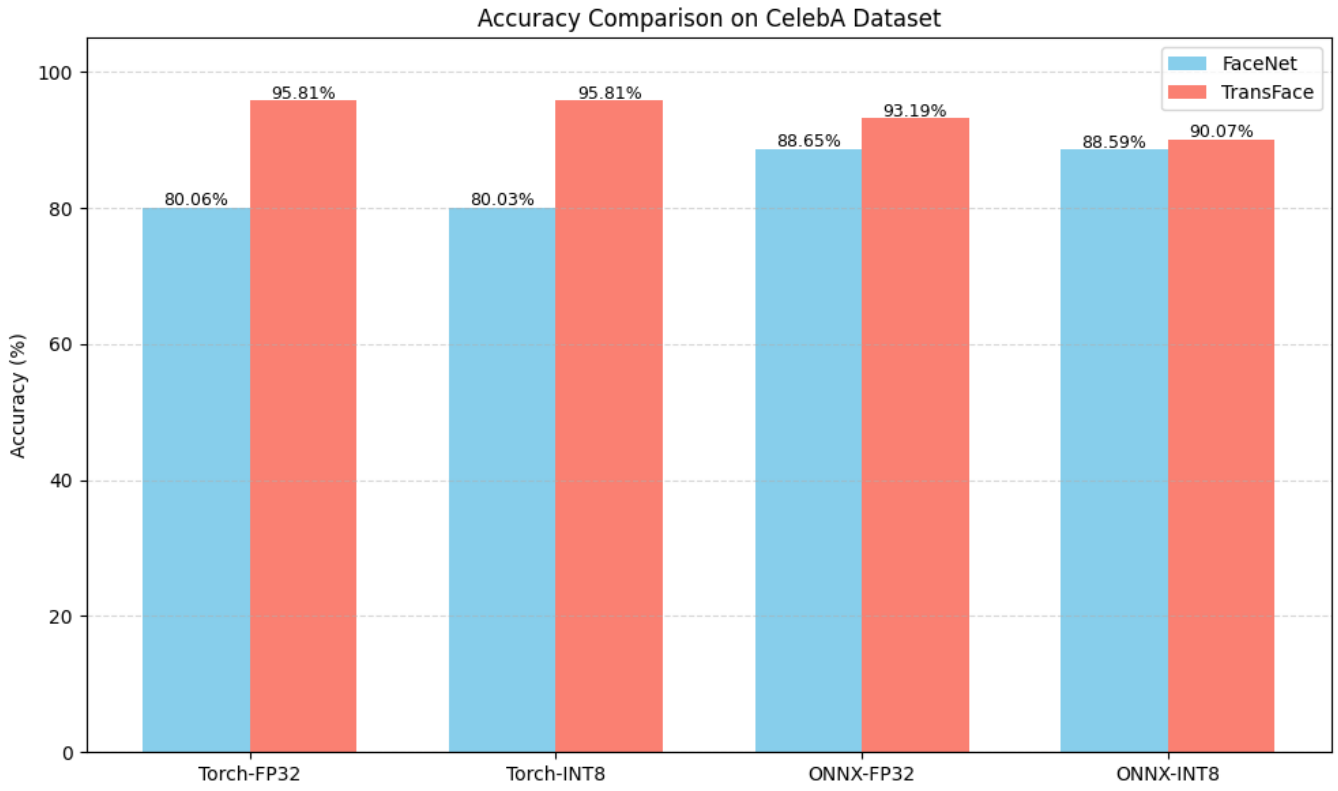


Figura VII: Comparativo de acerto CelebA



VIII. Conclusão

Este trabalho teve como objetivo avaliar modelos de geração de vetores de características de face para reconhecimento de pessoas, por comparação de modelos com tecnologias distintas, medir impacto de quantização e verificar uso de disco ao modificar a forma como os valores são representados. Com os testes realizados, foi possível verificar que:

- Ao utilizar o modelo mais novo, os resultados de reconhecimento são melhores;
- Quantizar o Facenet tem pouco impacto na acurácia;
- Quantizar o Transface degrada acurácia;
- O formato dos dados reduz uso de disco.

Estes resultados indicam que modelos mais estabelecidos são mais robustos a quantização e que é possível armazenar embeddings utilizando menos disco. Entre as limitações, destaca-se o uso de apenas duas estratégias de quantização, uma de conversão e a avaliação em hardware x86; estudos futuros podem explorar quantização dinâmica em GPU Nvidia/TensorRT ou ARM. Em síntese, a quantização se mostra melhor quando aplicada a modelos bem estabelecidos e que modificar o tipo de dado do vetor reduz em até 80% o uso de disco.

Tabela VI: Arquitetura Facenet NN2

type	**output size**	**depth**	#1 × 1	#3 × 3 **reduce**	#3 × 3	#5 × 5 **reduce**	#5 × 5	**pool proj (p)**	**params**	**FLOPS**
conv1 (7x7x3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Referências

[1] Xiang An, Xuhan Zhu, Yang Xiao, Lan Wu, Ming Zhang, Yuan Gao, Bin Qin, Debing Zhang, and Ying Fu. Partial fc: Training 10 million identities on a single machine, 2021.

[2] ANPD. biometria e reconhecimento facial, 2024.

[3] ANPD. Anpd fiscaliza uso de sistema de reconhecimento facial na venda de ingressos e na entrada de estádios por 23 clubes de futebol, 2025.

[4] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age, 2018.

[5] Jun Dan, Yang Liu, Haoyu Xie, Jiankang Deng, Haoran Xie, Xuansong Xie, and Baigui Sun. Transface: Calibrating transformer training for face recognition from a data-centric perspective, 2023.

[6] DanJun6737. Transface. `https://github.com/DanJun6737/TransFace`, 2023.

[7] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5203–5212, 2020.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[9] Fabio Grellet. Aeroporto santos dumont vai fechar devido ao g-20 no rio; o que acontecerá com os voos?, 2024.

[10] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[11] Global Growth Insights. Face recognition readers market size, share, growth, and industry analysis, by types (bracket installation, wall mounting), by applications covered (hotels, office buildings, schools, shopping malls, communities, others), regional insights and forecast to 2033, 2025.

[12] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[13] Laura Onita. The high-tech fight against shoplifters, 2025.

[14] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823. IEEE, June 2015.

[15] scipy. scipy. `https://github.com/scipy/scipy`, 2008.

[16] Yash Srivastava, Vaishnav Murali, and Shiv Ram Dubey. A performance comparison of loss functions for deep face recognition, 2019.

[17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[18] timesler. facenet-pytorch. `https://github.com/timesler/facenet-pytorch`, 2019.

[19] Shivani Zoting. Facial recognition market size, share, and trends 2025 to 2034, 2025.

A Bibliotecas

- absl-py=2.3.0
- certifi=2025.4.26
- charset-normalizer=3.4.2
- coloredlogs=15.0.1
- contourpy=1.3.2
- cycler=0.12.1
- dlib=19.24.9
- easydict=1.13
- filelock=3.18.0
- flatbuffers=25.2.10
- fonttools=4.58.0
- fsspec=2025.5.1
- graphviz=0.8.4
- grpcio=1.71.0
- hf-xet=1.1.2
- huggingface-hub=0.32.2
- humanfriendly=10.0
- idna=3.10
- imageio=2.37.0
- imutils=0.5.4
- Jinja2=3.1.6
- joblib=1.5.1
- kiwisolver=1.4.8
- lazy_loader=0.4
- Markdown=3.8
- MarkupSafe=3.0.2
- matplotlib=3.10.3
- mpmath=1.3.0
- mxnet=1.9.1
- networkx=3.4.2
- numpy=1.26.4
- nvidia-cublas-cu12=12.6.4.1
- nvidia-cuda-cupti-cu12=12.6.80
- nvidia-cuda-nvrtc-cu12=12.6.77
- nvidia-cuda-runtime-cu12=12.6.77
- nvidia-cudnn-cu12=9.5.1.17
- nvidia-cufft-cu12=11.3.0.4
- nvidia-cufile-cu12=1.11.1.6

- nvidia-curand-cu12=10.3.7.77
- nvidia-cusolver-cu12=11.7.1.2
- nvidia-cusparse-cu12=12.5.4.2
- nvidia-cusparselt-cu12=0.6.3
- nvidia-nccl-cu12=2.26.2
- nvidia-nvjitlink-cu12=12.6.85
- nvidia-nvtx-cu12=12.6.77
- onnx=1.18.0
- onnxruntime=1.22.0
- opencv-contrib-python=4.11.0.86
- opencv-python=4.11.0.86
- packaging=25.0
- pandas=2.2.3
- pillow=11.2.1
- protobuf=6.31.0
- pyparsing=3.2.3
- python-dateutil=2.9.0.post0
- pytz=2025.2
- PyYAML=6.0.2
- requests=2.32.3
- safetensors=0.5.3
- scikit-image=0.25.2
- scikit-learn=1.6.1
- scipy=1.15.3
- setuptools=78.1.1
- six=1.17.0
- sympy=1.14.0
- tensorboard=2.19.0
- tensorboard-data-server=0.7.2
- threadpoolctl=3.6.0
- tifffile=2025.5.26
- timm=1.0.15
- torch=2.7.0
- torchaudio=2.7.0
- torchvision=0.22.0
- tqdm=4.67.1
- triton=3.3.0
- typing_extensions=4.13.2
- tzdata=2025.2
- urllib3=2.4.0
- Werkzeug=3.1.3
- wheel=0.45.1