Experimental Evaluation of Quantization Methods in Facial Recognition *

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

***Abstract*—This work presents an experimental evaluation of the effects of quantizing embedding generation models when applied to face recognition. Two models - Facenet (based on Inception-ResNet) and Transface (based on Vision Transformer) - are compared under different precision formats (FP32 and INT8) and inference backends (Torch and ONNX). Experiments were carried out on the LFW, VGGFace2 and CelebA datasets, assessing Rank-1 accuracy and employing cosine distance as the metric similarity metric. The results show that more established models, such as Facenet, are more robust to quantization, maintaining accuracy even in reduced precision formats, where Transface suffers noticeable degradation when quantized. Moreover, quantizing the embeddings vectors yielded up to an 80% reduction in storage requirements without significantly impacting performance. These findings underscore the feasibility of quantization as a strategy of optimizing models in resource constrained environments.**

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

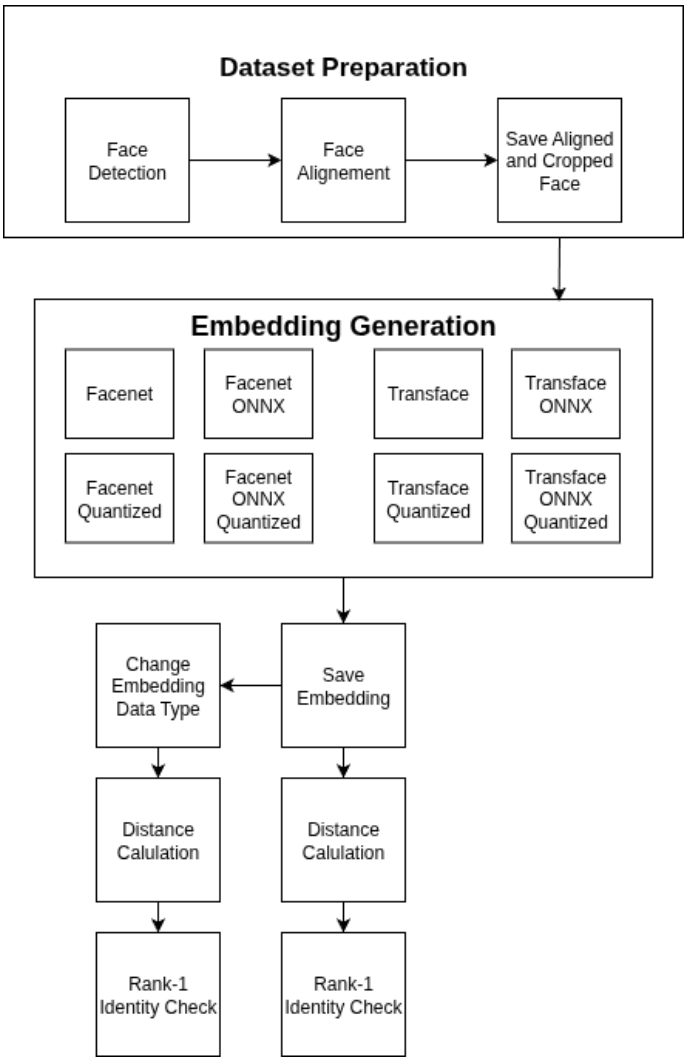
I. Introduction

Facial recognition using metric-learning algorithms is present in a variety of everyday applications. Examples include airports for authenticating check-in and boarding procedures [1], ticket sales and stadium entry systems for football matches [2], commercial establishments for security purposes [11], among others. The growth in AI tools for person recognition is expected to continue [16] [4] [9], raising two main challenges. First, systems that store records of processed face embeddings - such as security applications - face a large volume of data to be maintained for each individual, who enters a space. Second, there is the challenge of ensuring high accuracy in generating feature vectors that reliably identify a person, especially in critical scenarios like airport check-in authentication, which demand very high success rate.

The storage challenge can be mitigated by changing the data type used to save the face-feature vectors. The original networks generate 32-bit floating-point representations; by converting these to 8-bit integers, a substantial reduction in disk usage is anticipated. To address the accuracy challenge, newer model architectures—such as Transface, which employs Vision Transformers—were evaluated alongside quantization and model conversion techniques.

The aim of this work is to assess performance degradation in metric-learning algorithms when quantization methods are applied within the domain of face recognition. The primary contribution is an experimental evaluation showing distinct impacts across different inference platforms and model architectures, as detailed in the conclusions of this study.

Figura I: Evaluation methodology



II. Proposed Methodology

The experiments were carried out following the execution flow shown in I, which outlines the sequence of operations for preparing the data and verifying the model's recognition accuracy.

1. Dataset Preparation
2. Model Conversion.
3. Quantization (for experiments with the quantized model)
4. Embedding Generation
5. Embedding Saving
6. Distance Calculation
7. Rank-1 Accuracy Verification

Dataset Preparation: Each dataset undergoes a preprocessing step in which faces are detected and aligned on every image. The aligned faces are then saved as new files to ensure that all models receive inputs under the same initial conditions.

Model Conversion: The original models, which are available for the Torch inference engine, are first exported to the ONNX format. This adds variability in how different inference platformshandle the model and enable further

*

cross-platform testing.

Quantization: For the quantized model experiments, a quantization step is inserted into the workflow o that versions of each model in INT8 format can be tested alongside the original FP32 version.

Embedding Generation: With each variant of the model ready for inference, every image in the prepared dataset is processed to produce a fixed length feature vector(embedding) representating the face.

Embedding Saving: All generated embedding vectors are written to disk, creating files that will be used for batch analysis in subsequent steps.

Distance Calculation: For each embedding stored in disk, the cosine distance to all other embedding is computed. The smallest distance indicates the closest matching face.

Rank-1 Accuracy Verification: Finally, for each dataset, it is checked whether the nearest-neighbor embedding belongs to the same individual. The Rank-1 is calculated as the the proportion of correctly identified faces over the total number of faces in the dataset.

III. Models

For de experimental evaluation, two network architectures were employed. For the Facenet [12] model we used a PyTorch implementation available in the timesler/facenet-pytorch [15] GitHub repository, pre-trained on VGGFace2 dataset. For the Transface [5] model, we based our tests on the DanJun6737/Transface [6] implementation, also hosted on GitHub.

A. Facenet

The Facenet [12] network used in our tests adopts an Inception-ResNet [14] backbone, which merges inception modules with residual connections. We selected the NN3 variant for evaluation, which is architecturally identical to NN2 but accepts a 160x160 aligned-face input and produces a 512-dimensional embedding vector. Detailed layer configurations - including depth, filter sizes, parameter count, and floating-point operations - are presented in Table VI

B. Transface

The Transface [5] model is built upon the Vision Transformer(ViT) [7] architecture, with minor modifications that include a Transformer Encoder Backbone followed by a Squeeze-and-Excitation [8] module. It processes a 112x112 aligned-face image and outputs a 512-dimensional feature vector. For our experiments, we used the "Small"(S) variant, which comprises 12 transformer layers and 6 attention heads, and leverages weights pre-trained on the Glint360K datset.

IV. Datasets

The datasets used in the experiments were LFW [13], VGGFace2 [3], and CelebA [10]. Each dataset underwent face detection and alignment to stabilize the features collected by each model.

- **LFW:** Contains approximately 13 K images of 5750 individuals. It is the only dataset in which some identities have only a single image, resulting in unpaired entries. Its folder structure organizes into one directory per person.
- **VGGFace2:** follows the same per-person folder convention but differs in scale and distribution, comprising 170 K images across 480 individuals—yielding the highest images-per-person ratio.
- **CelebA:** uses a distinct layout: each image file has a unique name that is mapped to a person ID via an annotation file.

Table I consolidates each dataset's number of identities, total images, and average images per identity.

Tabela I: Datasets

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

V. Methods

The following methods were employed to convert and quantize the models, compare embeddings, and modify their representation.

A. Platform and Libraries

Experiments were run on a system with the following hardware and software specifications:

- CPU: AMD Ryzen 2600X
- GPU: NVIDIA RTX 4070
- RAM: 32GB
- OS: Ubuntu 22.04
- Python 3.13.2
- Nvidia Driver 550.163.01
- CUDA 11.5

All Python libraries and their versions are listed in the Appendix A (e.g., pandas 2.2.3, numpy 1.26.4, scipy 1.15.3)

B. Conversion

Prior to quantization, each PyTorch model was exported to ONNX using *torch.onnx.export*. The key parameters were:

- **Model:** The loaded PyTorch
- **dummy_input:** A tensor matching the model's expected input shape
- **"Transface.onnx":** Output model filename.
- **input_names=["input"]:** Network input name as input.
- **output_names=["embedding"]:** Network output name as embedding.
- **dynamic_axes** to allow variable batch size.
- **opset_version=13**

C. Quantization

Two quantization strategies were applied:

1. ONNX Static Quantization

Using the ONNX Runtime Quantization API(onnxruntime.quantization.quantize_static), each ONNX model was quantized to INT8. Calibration data was provided via a *CalibrationDataReader* (using VGGFace2 validation images), and both activations and weights were quantized as QInt8

2. PyTorch Dynamic Quantization

The original PyTorch models were quantized dynamically at inference time with *torch.quantization.quantize_ dynamic*, targeting all *torch.nn.Linear* and *torch.nn.LSTM* layers and casting their parameters to *torch.qint8*

D. Distance Computation

To measure similarity between embeddings, cosine distance was used:

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

This yields 0 for identical vectors. Implementation utilized *scipy.spatial.distance.cosine*.

E. Accuracy Calculation

Rank-1 accuracy was computed as follows: for each embedding in the dataset, identify the other embedding with the smallest cosine distance; if it belongs to the same person, count it as correct. Rank-1 accuracy is then as illustrated in Equation (2).

$$\text{Accuracy} = \frac{CorrectMatches}{FacesInDataset} \tag{2}$$

F. Embedding Representation

To reduce on-disk storage, each 512-dimensional embedding (originally FP32 in [–1,1]) was converted to INT8 by multiplying each component by 127, rounding to the nearest integer, and casting to np.int8

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

This simple cast preserves relative distances while reducing storage by up to 80%.

VI. Results

The experimental evaluation was designed to answer three research questions:

RQ1 : What is the reduction in recognition accuracy when quantization is applied?

RQ2 : Is there a difference between using Torch's dynamic quantization and ONNX's static quantization?

RQ3 : Between Transface and Facenet—which are popular face-recognition algorithms—which one is more sensitive to quantization?

A. Torch vs. ONNX Conversion

Before quantization, in Table II and Table III, we compared the baseline (FP32) accuracy of each model under the two inference engines.

- **Facenet:** Converting from Torch to ONNX boosts Rank-1 accuracy by 4 percentage points on LFW (Torch 61.41% to ONNX 65.47%) and VGGFace2 (92.62 to 96.66), and by 8 points on CelebA (80.06% to 88.65%), as seen in Table II.
- **Transface:** In contrast, Table III shows that, ONNX conversion slightly reduces accuracy: –1 point on LFW (68.49% to 67.15%) and –2 points on both VGGFace2 and CelebA (99.26% to 97.65%, 95.81% to 93.19%).

B. Impact of Quantization

Quantization effects were measured on both Torch (dynamic) and ONNX (static) engines, can be seen in Table II and Table III, explicitly in columns **FP32** and **INT8**:

- **Facenet**
  - LFW Torch: FP32 61.41% to INT8 61.27% (–0.14 points)
  - LFW ONNX: FP32 65.47% to INT8 65.54% (+0.07 points)

These minimal changes (<0.2 points) indicate that Facenet is highly robust to 8-bit quantization.

- **Transface**
  - LFW Torch: FP32 68.49% to INT8 68.48% (–0.01 points)

– ONNX: FP32 67.15% to INT8 64.30% (–2.85 points on LFW), and larger drops on VGGFace2 (97.65% to 93.62%, –4.03 points) and CelebA (93.19% to 90.07%, –3.12 points).

Tabela II: Facenet test results

	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: Transface test results

	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%

C. Facenet x Transface

When we place Facenet and Transface side by side across all experimental conditions, several important patterns emerge from analysis of Table II and Table III:

1. **Baseline Performance (FP32, Torch)**
   - On the LFW dataset, Transface achieves approximately 68.5% Rank-1 accuracy, outperforming Facenet's 61.4% by over 7 percentage points as shown in Figure III. This gap reflects Transface's stronger representation power under full-precision inputs and the lighter preprocessing requirements of its transformer backbone.
   - A similar advantage holds on VGGFace2 (99.3% vs. 92.6%) and CelebA (95.8% vs. 80.1%) as shown in Figure IV for VGGFace2 and Figure V for CelebA, demonstrating that Transface consistently extracts more discriminative embeddings when quantization is not applied.
2. **Impact of Conversion to ONNX (FP32, ONNX)**
   - Converting Facenet to ONNX yields a notable boost (e.g., LFW: 61.4% → 65.5%), narrowing the gap with Transface under ONNX (Transface LFW: 67.2%). Here, Facenet closes to within 2 percentage points, suggesting that its Inception-ResNet architecture benefits more from ONNX's graph optimizations and runtime kernels.
   - On larger datasets like VGGFace2 and CelebA, Facenet's conversion gains (4–8 points) reduce the performance differential, though Transface still leads by about 1–4 points in FP32 ONNX mode.
3. **Quantization Sensitivity (INT8)**
   - **Torch Dynamic Quantization:** Both models show minimal changes under Torch's on-the-fly quantization: Facenet drops by only 0.14 points on LFW; Transface is essentially unchanged (–0.01 points). This indicates that both networks tolerate weight quantization when activations remain in floating point.
   - **ONNX Static Quantization:** Here the divergence is stark. Facenet actually gains a slight 0.07 points on LFW (65.47% → 65.54%), while Transface suffers a 2.85 pt loss (67.15% → 64.30%). On VGGFace2 and CelebA, Transface's accuracy falls by 3–4 points, whereas Facenet remains within a 0.2 pt band of its FP32 ONNX results.
4. **Operational Takeaway**
   - Although Transface consistently delivers higher absolute accuracy in FP32, its performance under INT8 static quantization degrades noticeably. This suggests that transformer-based face encoders—while powerful—are more brittle when activations and weights are jointly quantized without further calibration or fine-tuning.
   - Facenet's convolutional-residual design proves far more robust: it not only maintains its accuracy but, in some cases, benefits slightly from ONNX's low-precision optimizations. For deployment scenarios where memory footprint and inference speed are critical, Facenet in ONNX INT8 mode offers the best balance of stability and efficiency.

In summary, if raw accuracy is the sole criterion and full-precision hardware is available, Transface is the superior choice. However, for resource-constrained environments requiring aggressive quantization, Facenet emerges as the more reliable and consistent performer.

D. On-Disk Storage Savings

Converting each 512-dimensional embedding from 32-bit floats to 8-bit integers yields substantial reductions in file size, as shown in Table IV.

Tabela IV: Embedding file sizes

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

Tabela V: Accuracy Values INT

Engine	Model	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%

- On LFW, storage drops from 136 MB to 26 MB—a savings of 110 MB ($\approx 81\%$).
- On VGGFace2, the largest dataset, embeddings shrink by 1,031 MB ($\approx 75\%$), going from 1.37 GB down to just 338 MB.
- On CelebA, files fall by 1,672 MB ($\approx 81\%$), from about 2.07 GB to 401 MB.

These reductions translate directly into lower storage costs and faster disk I/O during large-scale batch processing, without any detectable change in Rank-1 accuracy when using INT8 embeddings. In practical terms, a system storing embeddings for millions of faces could save terabytes of space simply by adopting this quantization strategy—making it highly attractive for cloud deployments or edge devices with limited memory.

E. Summary

RQ1 : Quantization causes negligible drops for Facenet (<0.2 points) but noticeable drops for Transface (up to 4 points).

RQ2 : Torch dynamic quantization is gentler than ONNX static quantization for Transface; for Facenet, both are effectively equivalent.

RQ3 : Facenet is more resilient to quantization than Transface, despite Transface's higher baseline accuracy.

VII. Discussion

From the results presented, we can assess the experimentally measured recognition performance against the figures reported in the reference articles. For the Facenet model, we achieved a maximum accuracy of 92.62% on Torch and 96.66% on ONNX—both in FP32 on the VGGFace2 dataset—which exceeds the accuracy originally reported in the Facenet paper. For Transface, the original article reports a peak accuracy of 99.85%; in our experiments we observed maximum accuracies of 99.27% on VGGFace2 and 95.81% on CelebA, both using Torch. These findings suggest that, to improve face-identification accuracy, the best approach is to employ a model based on the most advanced methods (e.g., Vision Transformers). However, converting and quantizing such newer models proves less efficient than applying the same techniques to more established architectures, which actually see slight increases in accuracy. This may be related to the deeper understanding of the behavior and characteristics of layers in well-studied networks.

The results also demonstrate that changing the way embedding values are represented reducing from 32-bit floats to 8-bit integers—very effectively lowers on-disk storage requirements without degrading recognition performance as shown in Figure II. This outcome is expected, since each value's bit-width is reduced from 32 bits to 8 bits, and the network's output range of –1 to 1 is simply rescaled to –127 to 127.

Figura II: Drive file size

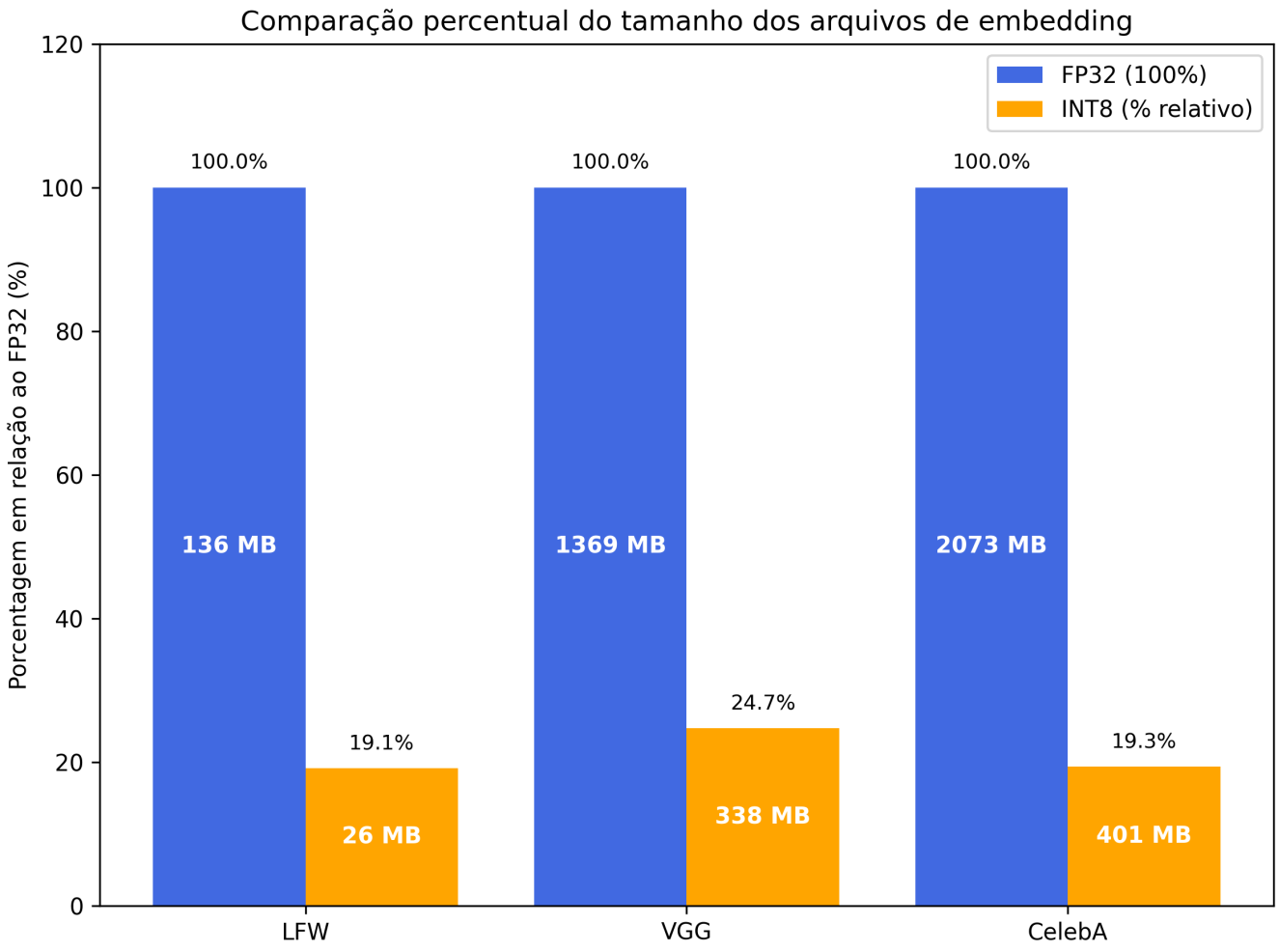


Figura III: LFW Success rate

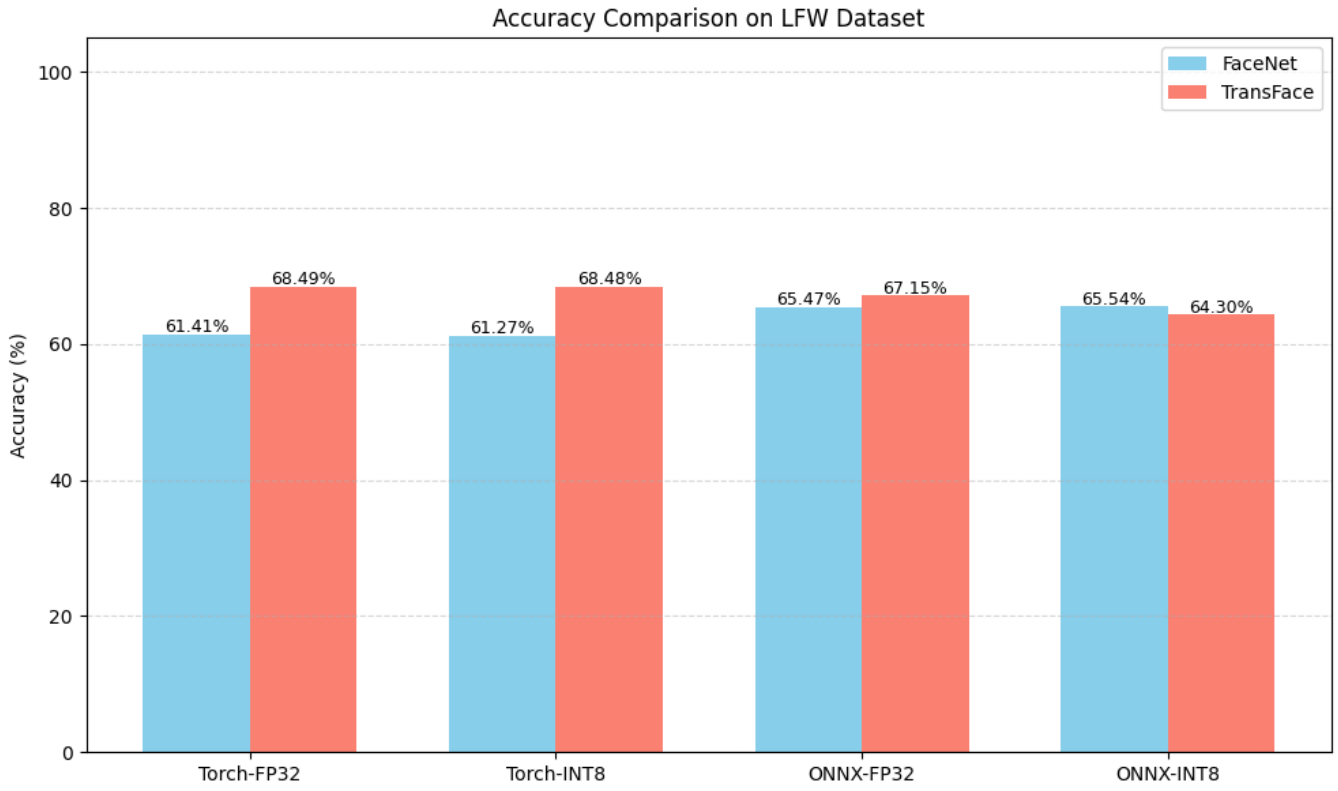


Figura IV: VGGFace2 Success rate

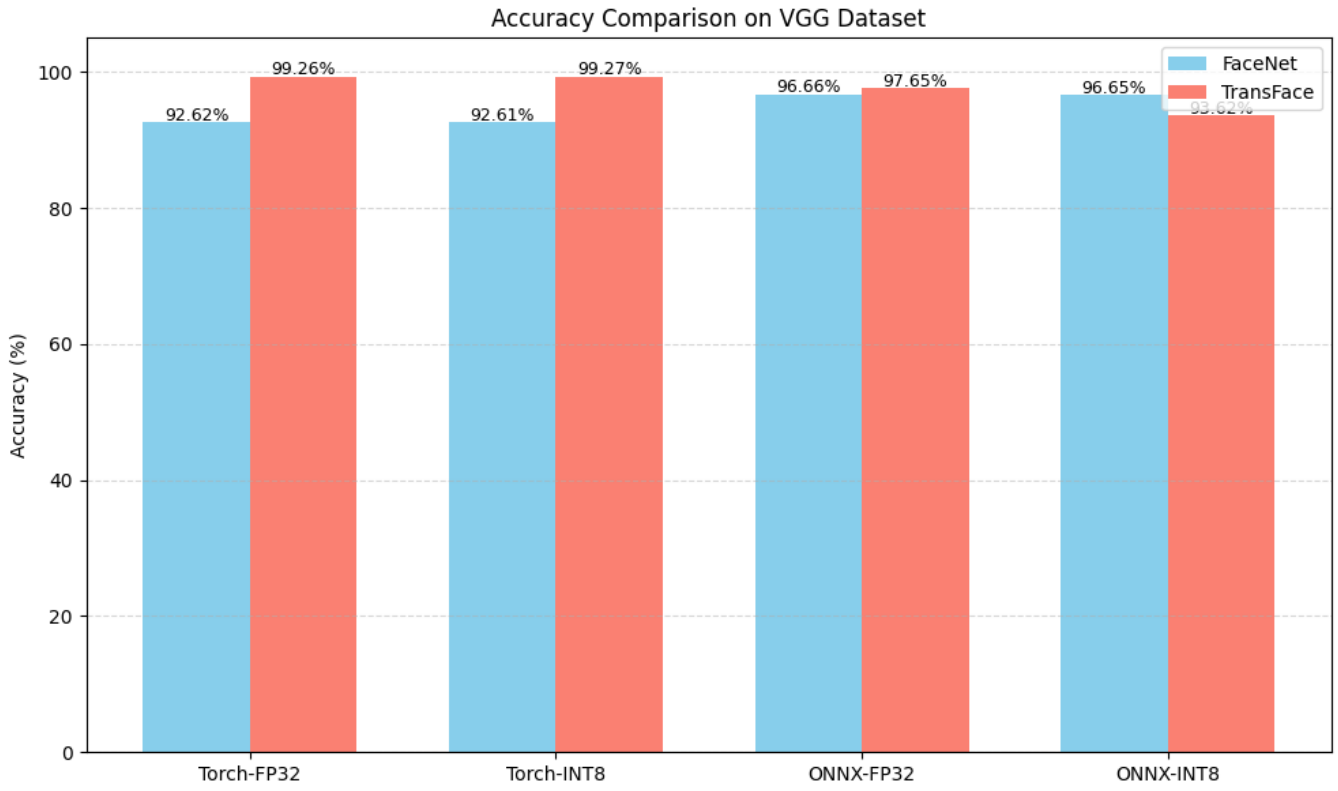
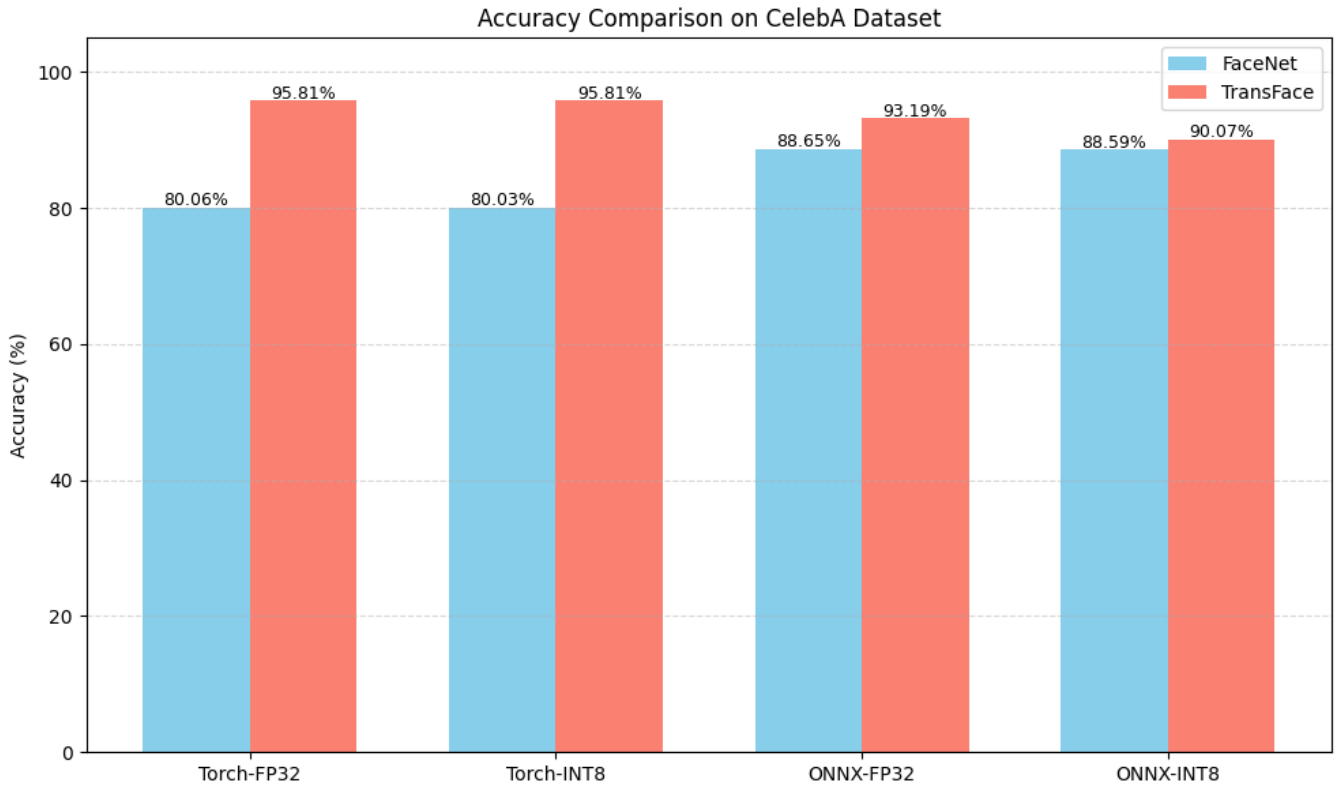


Figura V: CelebA Success rate



VIII. Conclusion

This work set out to evaluate face-feature-vector generation models for person recognition by comparing architectures with different technologies, measuring the impact of quantization, and assessing disk usage when altering the data representation. The experiments demonstrated that:

- Using the newer model yields better recognition results;
- Quantizing Facenet has little impact on accuracy;
- Quantizing Transface degrades accuracy;
- Changing the data format reduces disk usage.

These findings indicate that more established models are inherently more robust to quantization and that embeddings can be stored using significantly less disk space. Among the limitations are the application of only two quantization strategies, a single conversion approach, and evaluation solely on x86 hardware; future work could explore dynamic quantization on NVIDIA/TensorRT GPUs or ARM platforms. In summary, quantization proves most effective when applied to well-established models, and modifying the vector data type can reduce disk usage by up to 80%.

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] ANPD. biometria e reconhecimento facial, 2024.

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

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

[4] PW Consulting Automotive Machinery Research Center. Face recognition attendance machine market, 2025.

[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] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

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

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

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

[16] 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