Computer Vision with Deep Learning

Chapter 9: Face Recognition (2)

Eric Le

2023

Nội dung

1. Giới thiệu
2. GhostFaceNet
3. ArcFace Loss
4. FAISS
5. Tuning hệ thống Face Recognition

1. Giới thiệu

- **Ở bài trước, chúng ta biết được một hệ thống Face Recognition (FR) gồm nhiều module nhỏ:**
  - Face Detection & Alignment
  - Face Recognition
  - Nearest Neighbor Matching

1. Giới thiệu

- Việc nhận diện vị trí và extract gương mặt có thể được thực hiện dễ dàng bằng các model Object Detection (Yolo…)

- Bài học này tập trung chủ yếu vào việc huấn luyện model Face Recognition (FR).

1. Giới thiệu

- **Nhìn chung, mục tiêu của các model FR là biến mỗi hình ảnh gương mặt trở thành 1 vector embedding đặc trưng sao cho:**
  - Các gương mặt thuộc cùng 1 identity có độ tương đồng cao hoặc khoảng cách thấp (low intra-cluster distances).
  - Các gương mặt thuộc identity khác nhau có khoảng cách lớn (high inter-cluster distances).

1. Giới thiệu

❑ Các ưu điểm của việc sử dụng Deep Learning cho FR:

- Tăng cường độ chính xác trên các dataset phức tạp.
- Tự động extract các features cần thiết cho việc nhận dạng.

1. Giới thiệu

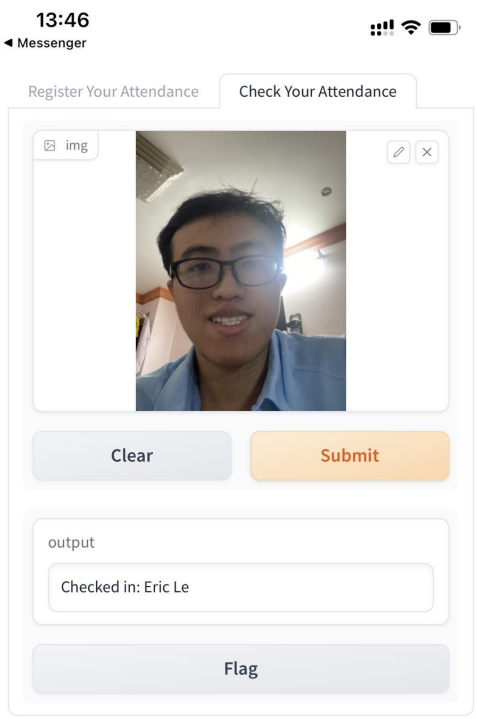
❑ Trong bài học này chúng ta sẽ tìm hiểu:

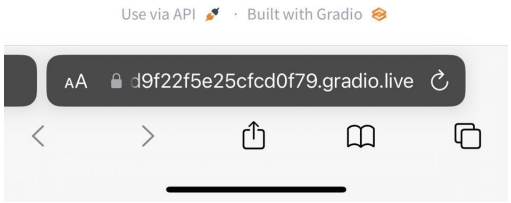
- Kiến trúc GhostFaceNet (2023).
- ArcFace Loss.
- Và 1 số tiến bộ trong lĩnh vực FR sử dụng Deep Learning.

1. Giới thiệu

Trong bài học này chúng ta sẽ:

- Xây dựng một ứng dụng FR hoạt động đa nền tảng
- Thử nghiệm ứng dụng trên bộ dataset VNCeleb





1. Giới thiệu

- Publish giải pháp trên nền tảng Azure Web App
- Thực hành Continuous Deployment with Git & Azure

Check Attendance Result

Total time: 3.2804742219999525 seconds

ID	Name	Distance	Images
[4729]	Diep	[0.6936719417572021]	
[4730]	Diep	[0.6936719417572021]	
[4731]	Diep	[0.6936719417572021]	

Nội dung

1. Giới thiệu
2. **GhostFaceNet**
3. ArcFace Loss
4. FAISS
5. Tuning hệ thống Face Recognition

2. GhostFaceNet

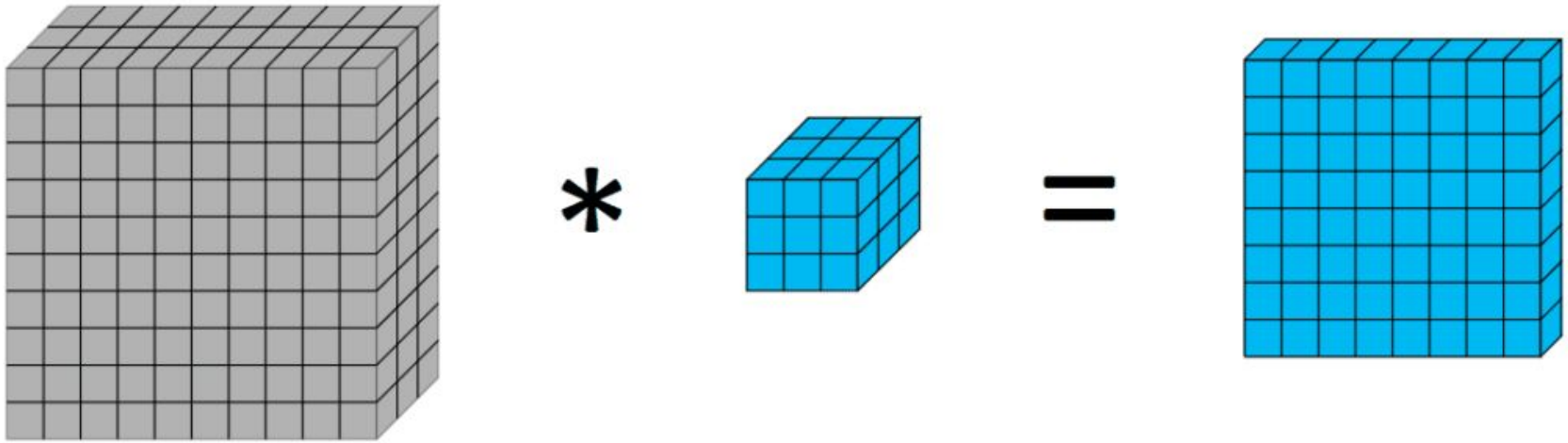
❑ Depthwise Convolution

- Trong phép tính Convolution thông thường, do mỗi ảnh được xem là 1 ma trận 3 chiều RGB, nên kernel sẽ duplicate 3 lần, kết quả convolution trên các channel sau đó sẽ được cộng lại để tạo thành 1 output feature map duy nhất
- Số lượng feature map = số lượng filter kernel.

2. GhostFaceNet

❑ Depthwise Convolution

- Số lượng feature map = số lượng filter kernel.



Ví dụ: Thực hiện convolution 3×3 kernel trên input 10x10x3 sẽ trả về output có kích thước 8x8x1.

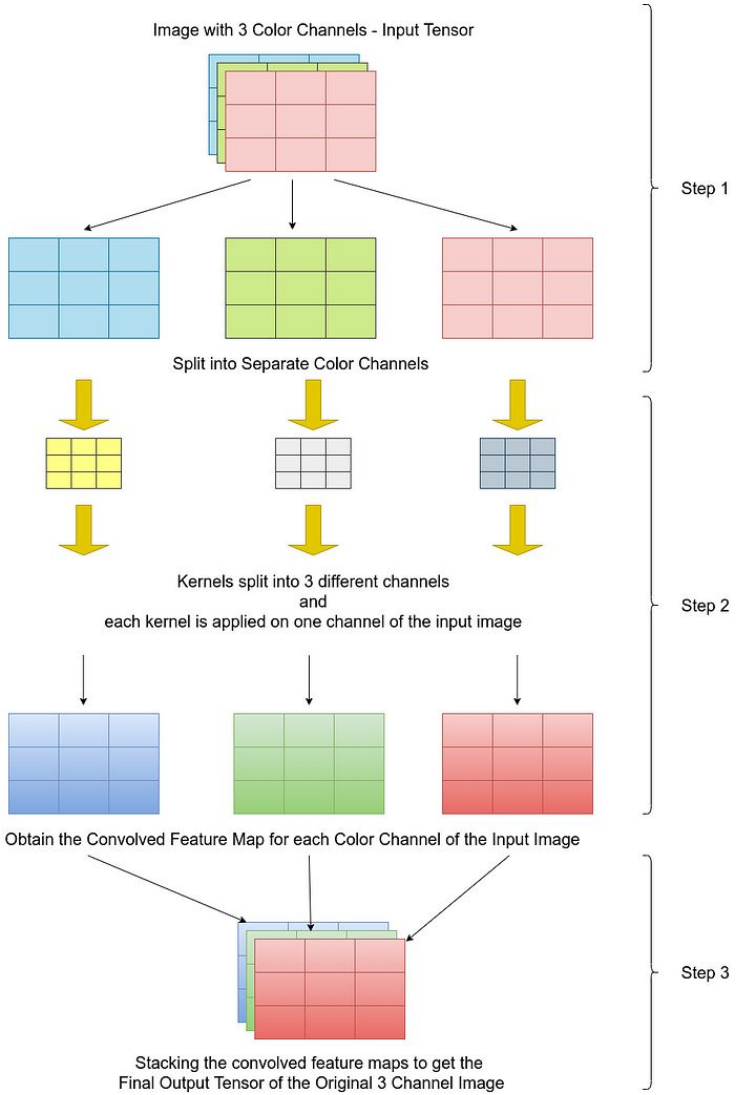
2. GhostFaceNet

Depthwise Convolution

- Depthwise convolution đề xuất 1 hướng tiếp cận khác, cho phép tăng số lượng channel trong output volume mà không cần tăng số lượng filter.
- Depthwise convolution thực hiện convolution trên từng channel input, sau đó tổng hợp các kết quả thay vì convolution từng phần trên toàn bộ channel input như convolution thông thường.

2. GhostFaceNet

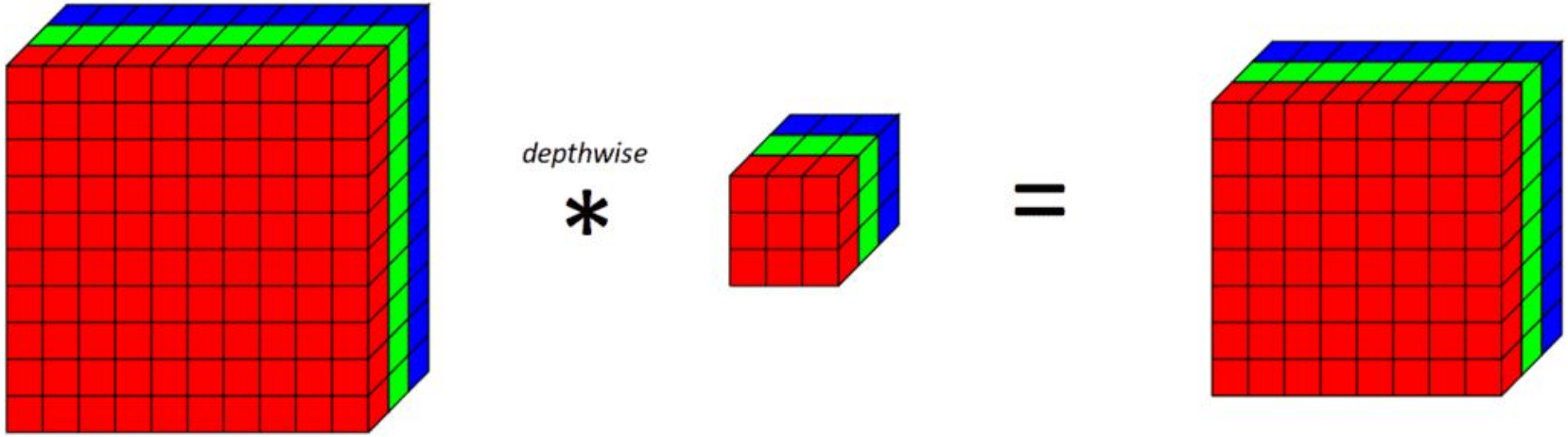
❑ Depthwise Convolution



2. GhostFaceNet

Depthwise Convolution

- Số lượng feature map = số lượng channel trong input.

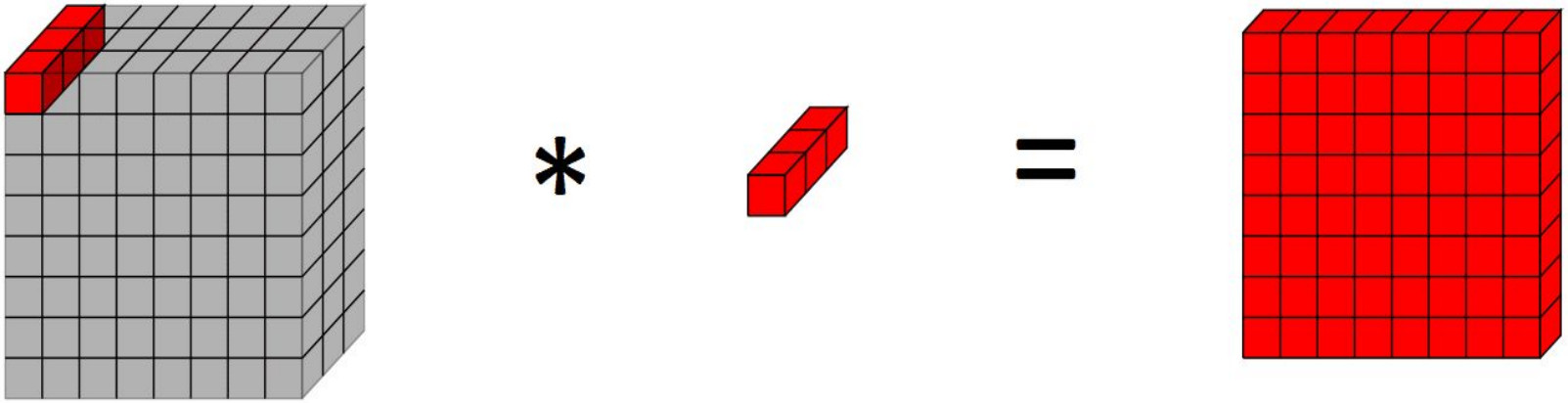


Ví dụ: Thực hiện convolution 3×3 kernel trên input 10x10x3 sẽ trả về output có kích thước 8x8x3.

2. GhostFaceNet

❑ Pointwise Convolution

- Kích thước feature map = Kích thước của input.

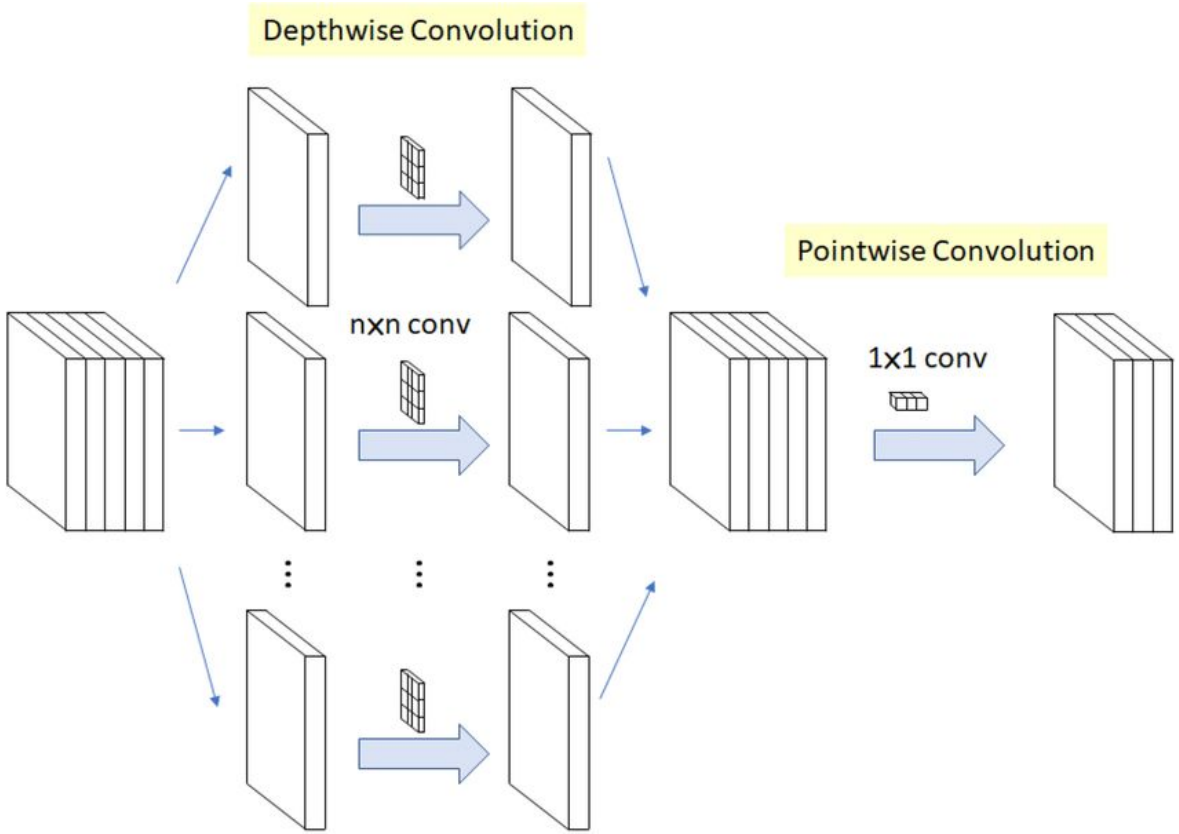


Ví dụ: Thực hiện pointwise convolution 3×3 kernel trên input 10x10x3 sẽ trả về output có kích thước 10x10x1.

2. GhostFaceNet

❑ Depthwise Separable Convolution

- Sự kết hợp giữa Depthwise và Pointwise convolution



2. GhostFaceNet

- Depthwise Separable Convolution cho phép giảm chi phí tính toán trong khi kích thước của output volume không đổi.
- Depthwise Separable Convolution là cơ sở để xây dựng nên các lightweight Architecture như MobileNet, GhostNet…

2. GhostFaceNet

Feature map redundancy

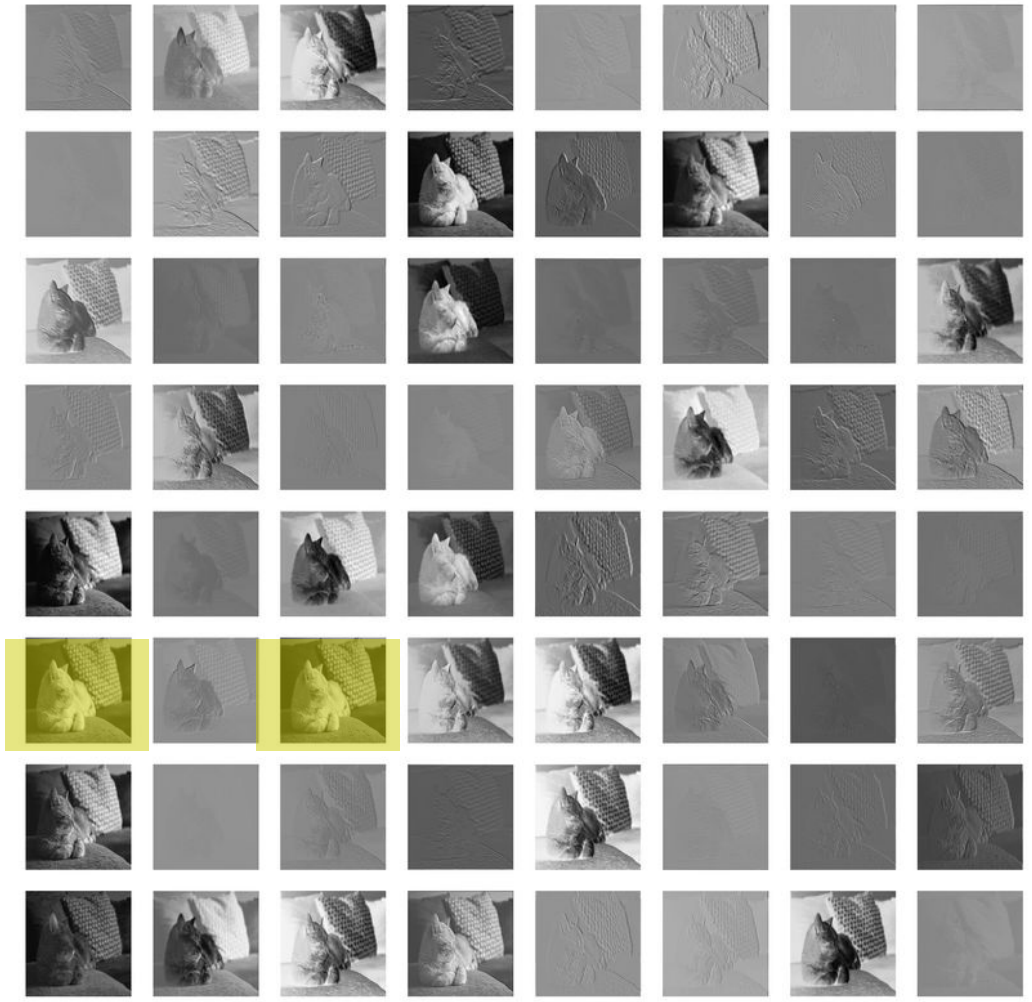
- 1 nhóm các nhà khoa học đã khảo sát mức độ liên quan giữa các feature map trong các lớp convolution layer bình thường.
- Kết quả visualization cho thấy, có rất nhiều feature map có toàn bộ đặc điểm từ rất giống nhau, cho đến trùng nhau hoàn toàn.
- Vấn đề này gọi là feature map redundancy (sự dư thừa các feature map).

2. GhostFaceNet

- Điều này dẫn đến 1 câu hỏi: Phải chăng chi phí tính toán cho các redundant feature maps này là không cần thiết?
- Liệu có cách nào để hạn chế các feature maps loại này trong khi vẫn giữ nguyên kích thước output và các feature map có ích khác không?
- Câu trả lời nằm trong 1 bài báo khoa học CVPR 2020: GhostNet: More Features from Cheap Operations.

2. GhostFaceNet

Các feature map được highlight hoàn toàn giống nhau



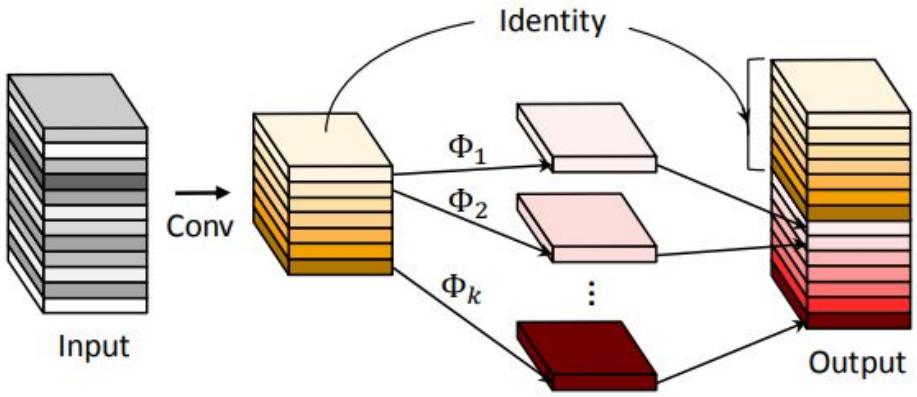
2. GhostFaceNet

❑ Ghost Module

- Giả sử output volume gồm C channel, để giảm thiểu feature map redundancy, GhostModule chỉ dùng convolution thông thường tạo ra x% trong số C channel.
- Các channel còn lại sẽ được tạo thông qua cheap linear operations, mà Depthwise hoặc Depthwise separable convolution là các lựa chọn hàng đầu.

2. GhostFaceNet

- Điều này cho phép giảm số lượng và chi phí tính toán, trong khi vẫn giữ nguyên performance của model



(b) The Ghost module.

2. GhostFaceNet

- Giả sử x là tensor input có kích thước (B, C, H, W) cho một module Ghost. Đầu ra của layer này được kỳ vọng là tensor x1 có kích thước (B, C1, H1, W1).

2. GhostFaceNet

- Một Ghost Module sẽ hoạt động như sau:
  - Bước 1 (Primary Convolution): Tính toán f(x) trên tensor đầu vào x để tạo ra một tensor có kích thước (B, $C_{1/2}$, H1, W1) trong đó f(x) là 1 convolution layer tiêu chuẩn cùng với batch normalization và ReLU. Tensor mới này có thể được ký hiệu là y_1.
  - Bước 2 (Secondary Convolution): Tính toán g(x) trên tensor y1 để tạo ra một tensor có kích thước (B, C1/2, H1, W1) trong đó g(x) là Depthwise Convolution cùng batch normalization với hàm ReLU. Tensor mới này có thể được ký hiệu là y2.
  - Bước 3 (Stack): Xếp chồng/ghép y_1 và y_2 để tạo thành tensor đầu ra kết quả x1.

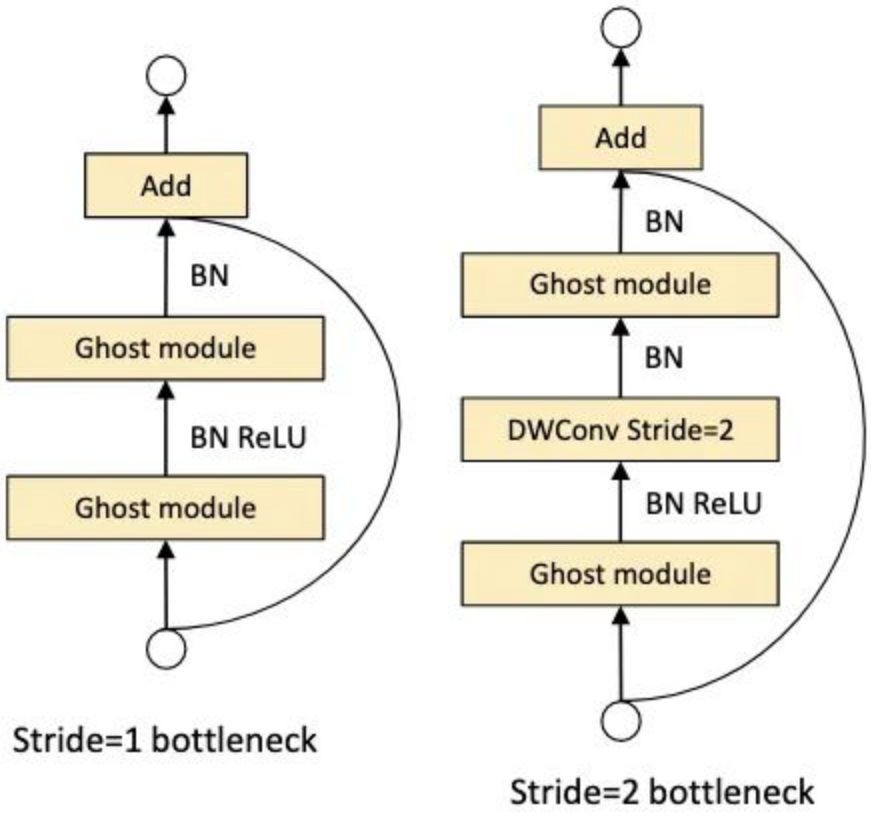
2. GhostFaceNet

❑ Ghost Bottleneck

- Các Ghost Module sẽ được xếp chồng lên nhau để tạo thành từng Ghost Bottleneck – đơn vị kiến tạo nên mạng GhostNet.
- Ghost Bottleneck cũng adapt các shortcut connection tương tự như mạng Resnet.

2. GhostFaceNet

Ghost Bottleneck



2. GhostFaceNet

GhostNet

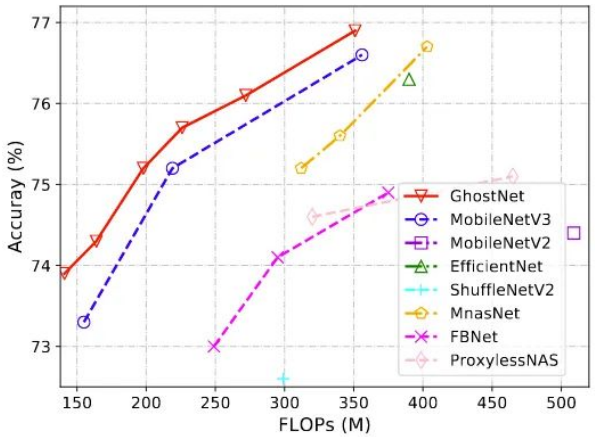


Figure 6. Top-1 accuracy *v.s.* FLOPs on ImageNet dataset.

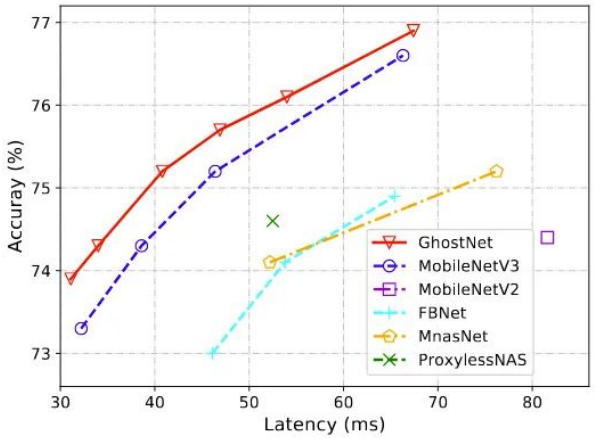


Figure 7. Top-1 accuracy *v.s.* latency on ImageNet dataset.

Table 1. Overall architecture of GhostNet. G-bneck denotes Ghost bottleneck. #exp means expansion size. #out means the number of output channels. SE denotes whether using SE module.

Input	Operator	#exp	#out	SE	Stride
$224^2 \times 3$	Conv2d 3×3	-	16	-	2
$112^2 \times 16$	G-bneck	16	16	-	1
$112^2 \times 16$	G-bneck	48	24	-	2
$56^2 \times 24$	G-bneck	72	24	-	1
$56^2 \times 24$	G-bneck	72	40	1	2
$28^2 \times 40$	G-bneck	120	40	1	1
$28^2 \times 40$	G-bneck	240	80	-	2
$14^2 \times 80$	G-bneck	200	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	480	112	1	1
$14^2 \times 112$	G-bneck	672	112	1	1
$14^2 \times 112$	G-bneck	672	160	1	2
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	Conv2d 1×1	-	960	-	1
$7^2 \times 960$	AvgPool 7×7	-	-	-	-
$1^2 \times 960$	Conv2d 1×1	-	1280	-	1
$1^2 \times 1280$	FC	-	1000	-	-

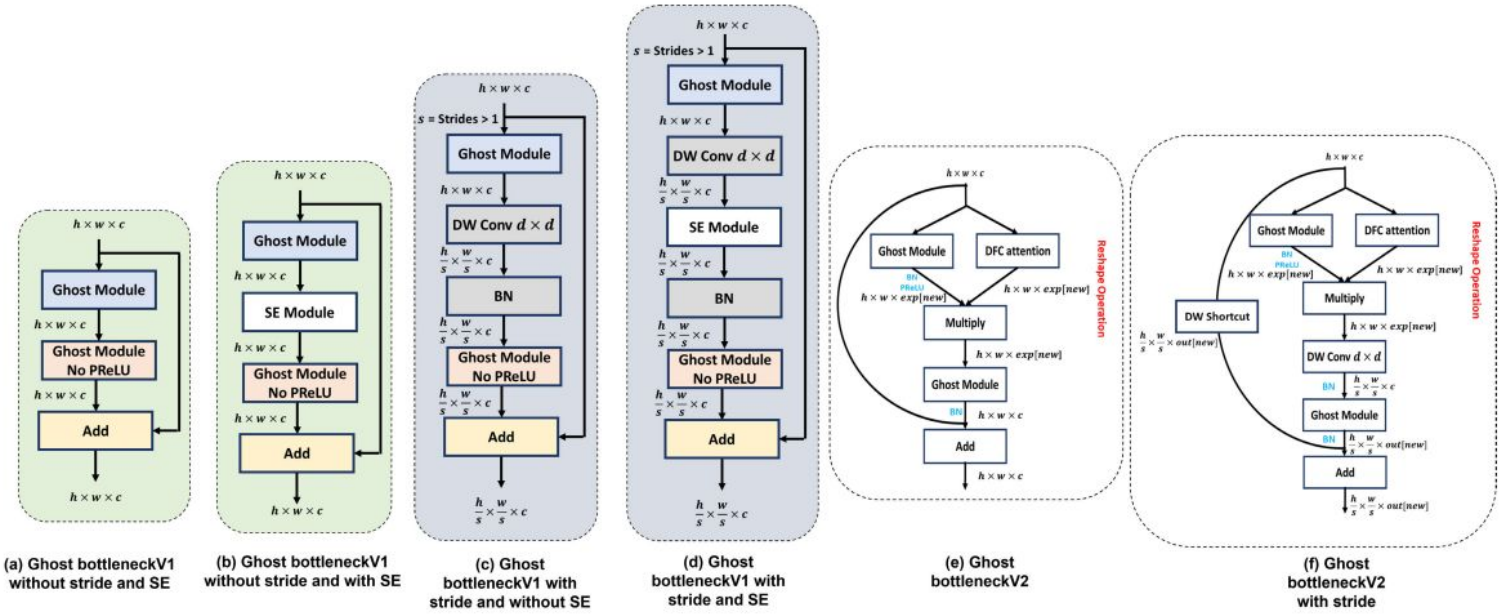
2. GhostFaceNet

GhostFaceNet

- Một kiến trúc Face Recognition dựa trên GhostNet
- Đây là 1 kiến trúc mới, lightweight và có hiệu năng khá tốt.
- Kiến trúc này được đề cập lần đầu trong GhostFaceNets: Lightweight Face Recognition Model From Cheap Operations (2023).

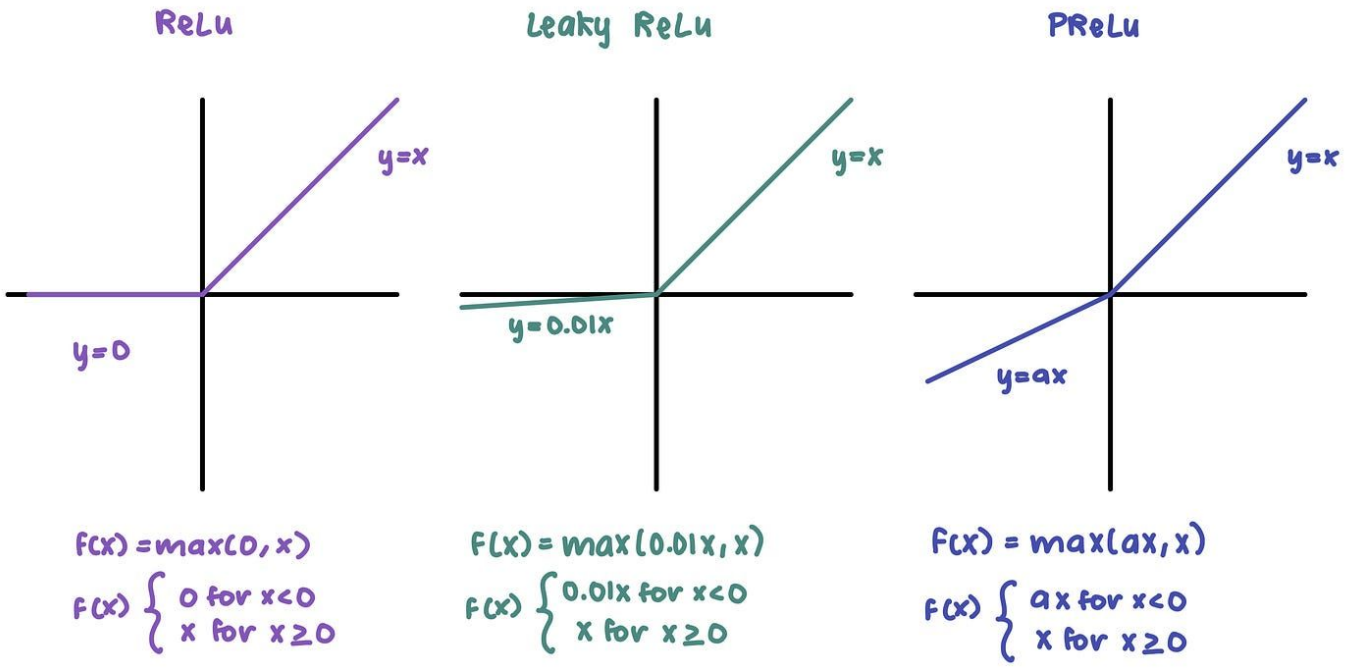
2. GhostFaceNet

- Ý tưởng: Tận dụng Ghost Module để giảm feature map redundancy.



2. GhostFaceNet

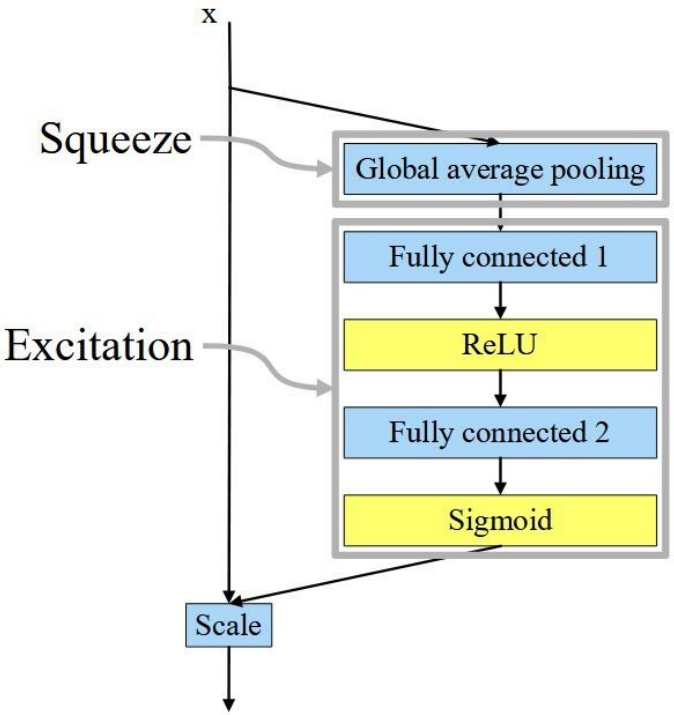
- Ý tưởng: Sử dụng PReLU activation function để tăng độ chính xác và tốc độ.



2. GhostFaceNet

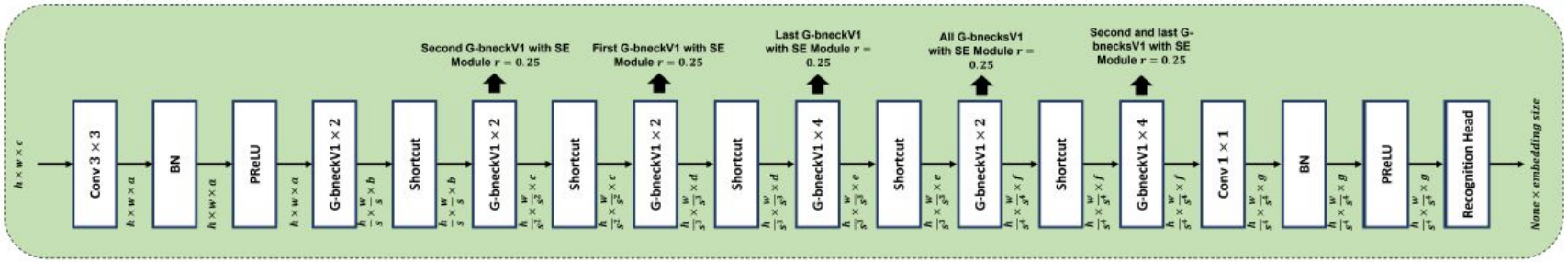
- Ý tưởng: Thay các FC layers bằng convolution layers trong Squeeze and Excitation module để tăng khả năng phân biệt.

2. GhostFaceNet



SQUEEZE AND EXCITATION BLOCK

2. GhostFaceNet



(a) GhostFaceNetV1−1

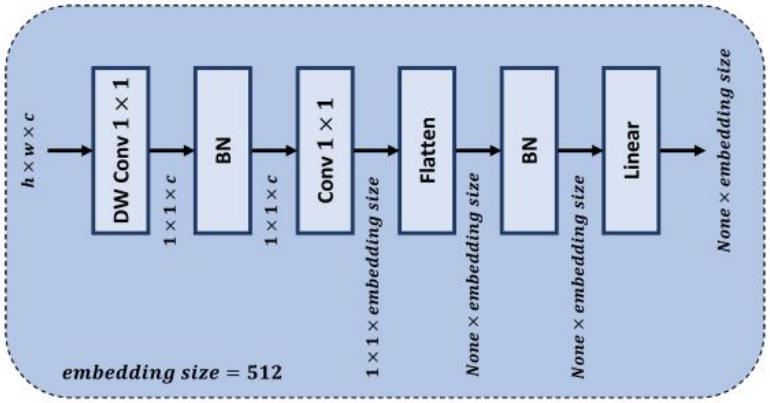


FIGURE 5. Modified GDC, Recognition Head.

2. GhostFaceNet

❑ Ưu điểm của GhostFaceNet:

- Gọn, nhẹ.
- Quá trình inference nhanh.
- Dung lượng model thấp (<20MB).
- Độ chính xác cao.
- Open source, support tốt.

2. GhostFaceNet

❑ Khuyết điểm của GhostFaceNet:

- Tốc độ thử nghiệm trong thực tế <5 FPs với 1 vài checkpoint (đã thử nghiệm trên google colab).

Nội dung

1. Giới thiệu
2. GhostFaceNet
3. **ArcFace Loss**
4. FAISS
5. Tuning hệ thống Face Recognition

3. ArcFace Loss

- **Một kỹ thuật được sử dụng trong GhostFaceNet và rất nhiều model FR khác: ArcFace Loss**
  - ArcFace Loss tăng cường khả năng học tập của model bằng việc tạo nên angular margin giữa các class (identity) khác nhau so với các phương pháp trước đây.

3. ArcFace Loss

ArcFace Loss

3. ArcFace Loss

ArcFace Loss xuất phát từ Softmax Loss (Cross Entropy + Softmax)

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

x_i denotes the deep feature of the i-th sample, belonging to the y_i-th class. W_j^T denotes the j-th column of the weight W and b_j is the bias term. The batch size and the class number are N and n, respectively.

3. ArcFace Loss

- Từ công thức trên, ArcFace tiến hành normalize các weights bằng L2 Norm sao cho ||w|| = 1. Đồng thời, cố định các bias = 0, từ đó chúng ta được:

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}}$$

3. ArcFace Loss

- Cuối cùng, ArcFace Loss đưa vào additive angular margin penalty m, để tăng cường sự khác biệt giữa các identity khác nhau và giảm sự khác biệt giữa các ảnh cùng thuộc 1 identity (compactness)

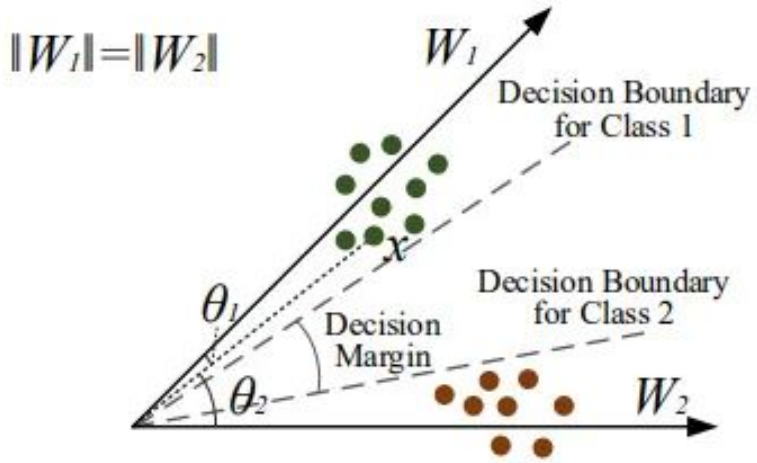
3. ArcFace Loss

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

where θ_j is the angle between the weight W_j and the feature x_i

s - feature scale, the hypersphere radius

m - angular margin penalty



Nội dung

1. Giới thiệu
2. GhostFaceNet
3. ArcFace Loss
4. **FAISS**
5. Tuning hệ thống Face Recognition

4. FAISS

- Một trong những yêu cầu cơ bản của bài toán FR: Tìm nearest neighbor cho 1 vector embedding.

- Một giải pháp đơn giản cho vấn đề này: Sử dụng kNNClassifier của Sklearn.

4. FAISS

❑ Tuy nhiên, việc này có 2 khuyết điểm:

- Khi số lượng vector embedding trong dataset càng lớn, thời gian thực hiện càng lâu (Sklearn không hỗ trợ GPU acceleration).
- Khi số lượng vector trong database trở nên rất lớn, có thể gây tràn bộ nhớ.

4. FAISS

FAISS (Facebook AI Similarity Search)

- Là một thư viện dùng để giải quyết vấn đề similarity search 1 cách có hiệu quả và hệ thống.
- Faiss được tối ưu hóa để có thể chạy trên cả CPU hoặc GPU nhằm tăng tốc độ xử lý, storing data và index trên RAM một cách tối ưu.

4. FAISS

- Một thử nghiệm nhỏ: Tìm vector tương đồng nhất với 1 query vector trong số 1000 vector có kích thước 512x1, thời gian trả về kết quả là:
  - `time to search neighbors: 0.0016546249389648438`

4. FAISS

FAISS (Facebook AI Similarity Search)

- Cài đặt trên CPU: pip install faiss-cpu
- Cài đặt trên GPU: pip install faiss-gpu

4. FAISS

```
# Define the size of index vectors
vector_dimension = 300

# Create the Flat L2 index
index = faiss.IndexFlatL2(vector_dimension)

# Add vectors to index. Embeddings can be NumPy arrays or torch tensors
index.add(embeddings)


# Save the index to disk
faiss.write_index(index, 'name_of_your_index')
```

4. FAISS

```
# Define the number of nearest neighbors to search for
top_n_neighbours = 4

# Read the index from disk
index = faiss.read_index('name_of_your_index')

# Search the top-k nearest neighbors of all the vectors in embedding
distances, neighbours = index.search(embeddings, k=top_n_neighbours)

print(neighbours)
print(distances)
```

Nội dung

1. Giới thiệu
2. GhostFaceNet
3. ArcFace Loss
4. FAISS
5. **Tuning hệ thống Face Recognition**

5. Tuning Face Recognition System

- **Một trong số các vấn đề nhức nhối trong bài toán Face Recognition là:**

1. Làm sao để giảm tỉ lệ False Positive Rate?
2. Làm sao để nhận diện các unknown hoặc stranger faces?
3. Xử lý thế nào khi ảnh đến từ nhiều nguồn khác nhau (chụp qua webcam, camera điện thoại…)

5. Tuning Face Recognition System

❑ Đối phó với False Positive Rate:

- False Positive cases xảy ra khi 2 gương mặt vốn không cùng 1 identity, lại được model xác nhận là của cùng 1 người. False Positive Rate (FPR) là tỉ lệ FP cases.
- Ở chiều ngược lại, False Negative Rate là tỉ lệ các trường hợp 2 gương mặt của cùng 1 người bị reject bởi model FR.

5. Tuning Face Recognition System

5. Tuning Face Recognition System

	Accept	Reject
Reject	Type I error (False accept)	Correct (True reject)
Accept	Correct (True accept)	Type II error (False reject)

5. Tuning Face Recognition System

- Thông thường, FPR nguy hiểm hơn FNR.
- Một phương pháp hữu hiệu có khả năng tăng cường accuracy của model là: Aggregation Enhanced Embedding Vector.

5. Tuning Face Recognition System

- Với phương pháp này, thay vì tìm similarity giữa query vector (qv) với toàn bộ vector trong database, ta tiến hành tìm similarity giữa *qv* & vector đại diện cho từng identity.

5. Tuning Face Recognition System

- Ví dụ: giả sử mỗi user submit 3 tấm ảnh gương mặt vào hệ thống, và có 1000 users, vậy thì thay vì tìm similarity giữa query vector và 3000 vectors, ta tìm similarity giữa query và 1000 vector đại diện. Mỗi vector đại diện được xây dựng dựa trên 3 tấm ảnh user submit.

5. Tuning Face Recognition System

- Ví dụ (tiếp theo): … Các vector đại diện có thể được xây dựng từ 1 phép toán hoặc hàm số tổng hợp (aggregation function) của các vector gương mặt ban đầu (3 vector cho mỗi user).
- Phổ biến nhất, chúng ta có thể dùng hàm Mean hoặc weighted average.

5. Tuning Face Recognition System

❑ Tại sao AEEV hoạt động tốt?

- Thứ nhất, phương pháp này hạn chế search space, giúp việc tìm nearest neighbor trở nên đơn giản, nhanh và nhất quán hơn.

5. Tuning Face Recognition System

- Thứ nhì, thông qua việc sử dụng các hàm aggregation như mean, vector đại diện sẽ có tính tổng quát và đa dạng cao hơn, nhờ vào việc các vector thành phần tương ứng với các hình khác nhau của cùng 1 user.

5. Tuning Face Recognition System

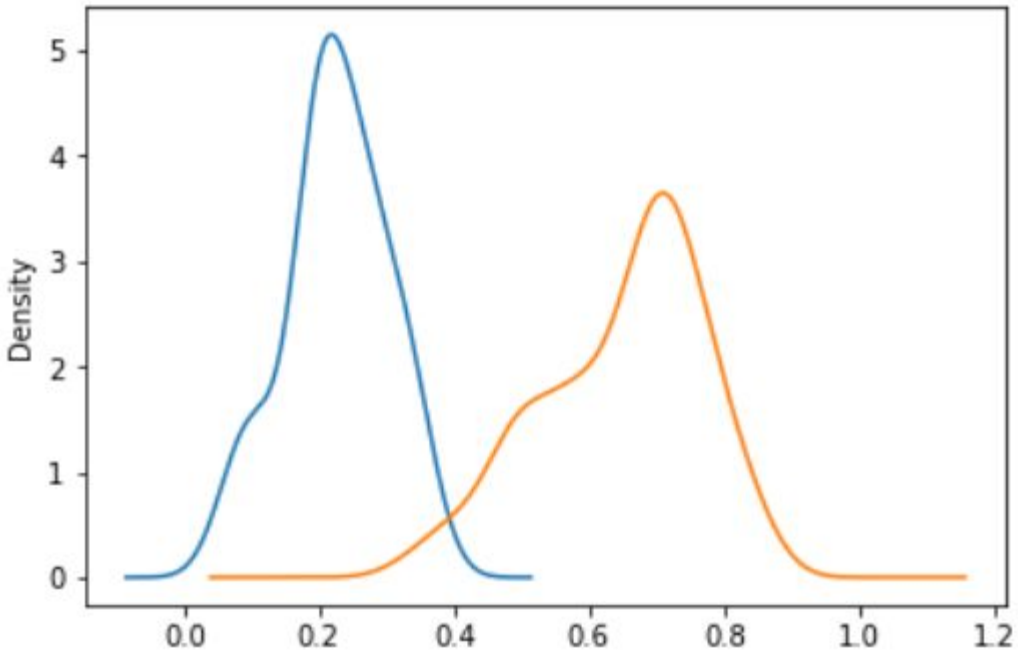
❑ Nhận diện Unknown Face

- Khi một gương mặt xa lạ xuất hiện trong hệ thống, chúng ta cần cảnh báo và không được thực hiện matching identity cho người lạ.
- Giải pháp: Chọn threshold phù hợp.

5. Tuning Face Recognition System

- Giải pháp: Chọn threshold phù hợp.
- Tiến hành thu thập các positive scores và các negative scores.
- Dùng đường cong ROC để tìm threshold tối đa hóa positive classification.
- Tham khảo: Fine Tuning The Threshold in Face Recognition - Sefik Ilkin Serengil (sefiks.com) và slide ROC-AUC đi kèm.

5. Tuning Face Recognition System



5. Tuning Face Recognition System

❑ Xử lý Multi Image Source

- Trong thực tế, ảnh user submit có thể đến từ nhiều nguồn khác nhau: Iphone, Ipad, other mobile phones, camera webcam…
- Điều này dẫn đến việc data khi train và khi test thực tế có sự khác nhau về chất lượng ảnh.

5. Tuning Face Recognition System

- Tuy nhiên, nếu nhìn nhận kỹ, thì sự khác biệt chủ yếu là do các ảnh đến từ nguồn khác nhau sẽ có resolution khác nhau.
- Ví dụ ảnh 2K sẽ khác với ảnh chụp từ camera webcam laptop (HD).
- Ngoài ra, tỉ lệ gương mặt trong hình ảnh cũng là 1 yếu tố ảnh hưởng đến chất lượng.

5. Tuning Face Recognition System

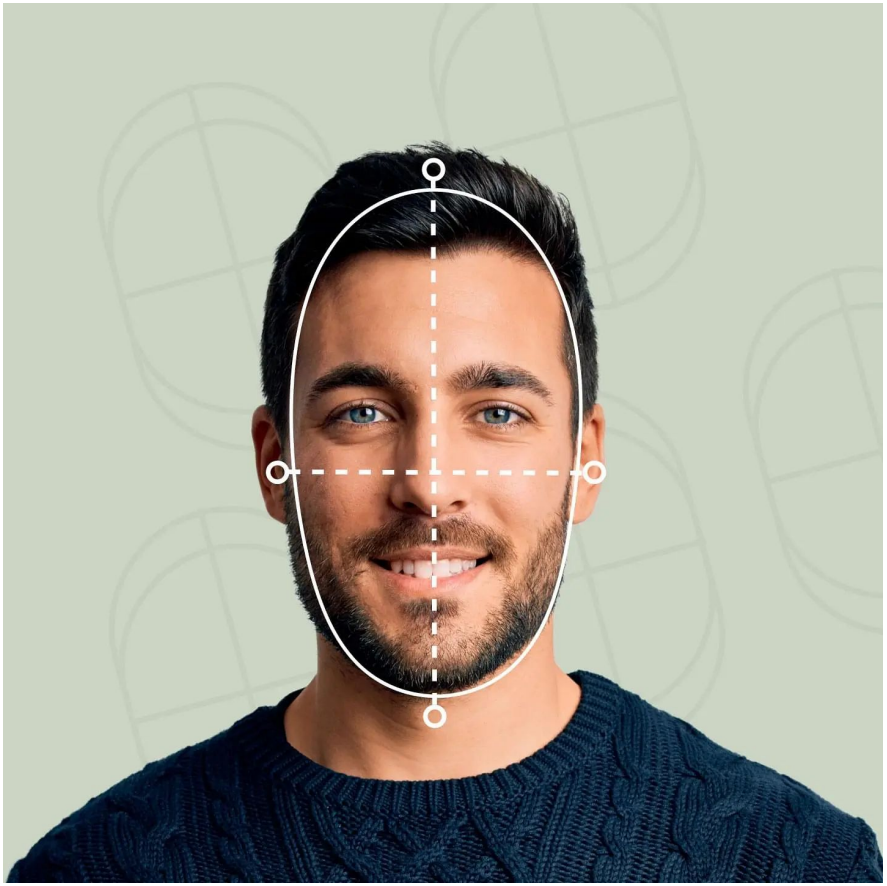
- Giải pháp: Resize các ảnh đầu vào về 1 resolution duy nhất – resolution phải đủ lớn để đảm bảo chất lượng hình ảnh, nhưng không được quá lớn.

5. Tuning Face Recognition System

- Ngoài ra, có thể hướng dẫn user cách lấy mẫu hình ảnh để đảm bảo tỉ lệ gương mặt trên diện tích ảnh, hoặc xây dựng model để kiểm tra yếu tố này.

5. Tuning Face Recognition System

❑ Xử lý Multi Image Source



Summary

- Face Recognition sử dụng các kỹ thuật ArcFaceLoss + Aggregation Vector Enhancement để tăng độ chính xác

Summary

Các bước trong luồng Face Recognition:

- Tạo vector đại diện bằng phương pháp Aggregation Vector Enhancement cho các identity đã có
- Đăng ký mới user: sử dụng các ảnh input tạo vector đại diện cho user mới

Summary

Các bước trong luồng Face Recognition:

- Check-in user: Tạo vector embedding từ ảnh input của user, tìm similarity giữa vector này với các vector đại diện sẵn có. Từ đó tìm ra identity.
- Có thể dùng phương pháp tune threshold.