

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA TOÁN - TIN HỌC  
—oo—



BÁO CÁO MÔN HỌC  
PHÂN TÍCH XỬ LÝ ẢNH

---

Face Image Retrieval  
Using Deep Learning Models

---

THÀNH VIÊN THỰC HIỆN BÁO CÁO

Lê Phú Trường

Đào Xuân Tân

Trần Lê Hữu Vinh

[22110245@student.hcmus.edu.vn](mailto:22110245@student.hcmus.edu.vn)   [22110195@student.hcmus.edu.vn](mailto:22110195@student.hcmus.edu.vn)   [22110263@student.hcmus.edu.vn](mailto:22110263@student.hcmus.edu.vn)

GIẢNG VIÊN HƯỚNG DẪN: ThS. Huỳnh Thanh Sơn

# Mục lục

<b>Giới thiệu về tài</b>	<b>3</b>
<b>1 Xử lý dữ liệu đầu vào</b>	<b>5</b>
1.1 Tiền xử lý dữ liệu . . . . .	5
1.2 Xử lý dữ liệu cho mô hình . . . . .	7
1.3 Chuẩn bị dữ liệu và các phép biến đổi (Transform) . . . . .	9
<b>2 Mô hình và các kỹ thuật liên quan</b>	<b>13</b>
2.1 Giới thiệu mô hình . . . . .	13
2.2 Phương pháp Fine-tuning . . . . .	19
2.2.1 Triplet-Loss . . . . .	19
2.2.2 Kỹ thuật tối ưu: Adam Optimizer [KB17] . . . . .	22
2.2.3 Thiết kế Embedding Layer . . . . .	24
2.2.4 Fine-tuning cho mô hình . . . . .	26
2.2.5 Phương pháp truy vấn: KD-Tree [DVL21] . . . . .	27
<b>3 Đánh giá mô hình</b>	<b>32</b>
3.1 Phương pháp đánh giá . . . . .	32
3.2 Đánh giá quá trình huấn luyện và kết quả . . . . .	33
3.2.1 Trước khi thực hiện Fine-Tuning . . . . .	33
3.2.2 Quá trình Fine-Tuning . . . . .	34
3.2.3 Sau khi thực hiện Fine-Tuning . . . . .	36
3.3 Áp dụng vào truy vấn ảnh . . . . .	38
<b>4 DEMO - Query Interface Design</b>	<b>42</b>
<b>Lời kết</b>	<b>45</b>
<b>Tài liệu tham khảo</b>	<b>48</b>

# Giới thiệu đề tài

Trong thời đại bùng nổ của dữ liệu hình ảnh, việc tìm kiếm và truy xuất hình ảnh một cách hiệu quả đã trở thành một nhu cầu thiết yếu trong nhiều lĩnh vực, từ nhận diện khuôn mặt, quản lý truyền thông, đến bảo mật thông tin. Hệ thống truy xuất hình ảnh dựa trên nội dung (CBIR - Content-Based Image Retrieval) là một giải pháp tiên tiến, cho phép người dùng tìm kiếm các hình ảnh tương tự dựa trên ảnh đầu vào thay vì các từ khóa hay siêu dữ liệu. Với sự phát triển mạnh mẽ của các mô hình học sâu (Deep Learning), việc ứng dụng các kỹ thuật này vào CBIR không chỉ nâng cao độ chính xác mà còn tối ưu hóa hiệu suất của hệ thống. Đề tài này tập trung vào việc xây dựng một hệ thống CBIR dựa trên các đặc trưng khuôn mặt, sử dụng bộ dữ liệu CelebA và các mô hình học sâu tiên tiến như ResNet-50 và MobileNet-V2. Bạn đọc có thể tham khảo repository chính thức của đề tài này thông qua đường dẫn <https://s.net.vn/iNFh>.

## 1 Mục tiêu của đề tài

- Xây dựng hệ thống truy xuất hình ảnh dựa trên nội dung (CBIR - Content-Based Image Retrieval), cho phép người dùng tìm kiếm các hình ảnh tương tự dựa trên ảnh đầu vào.
- Tối ưu hóa hiệu quả và độ chính xác của hệ thống bằng cách ứng dụng các mô hình học sâu (Deep Learning) và khai thác các đặc trưng khuôn mặt để nâng cao khả năng nhận diện và phân biệt.
- Cải thiện trải nghiệm người dùng bằng cách phát triển một giải pháp thực tiễn có thể áp dụng vào các lĩnh vực như nhận diện khuôn mặt, quản lý truyền thông, và bảo mật.

## 2 Giới thiệu bộ dữ liệu

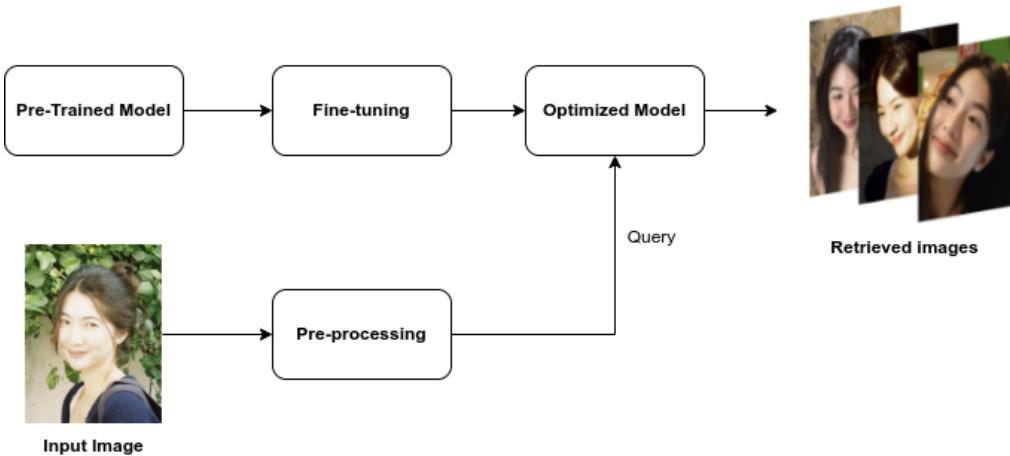
Trong nghiên cứu này, chúng tôi sử dụng bộ dữ liệu [CelebFaces Attributes Dataset \(CelebA\)](#), một tập dữ liệu lớn với hơn 200.000 hình ảnh khuôn mặt của các nhân vật nổi tiếng. CelebA được lựa chọn vì sự đa dạng về nội dung, số lượng lớn các hình ảnh, và các thuộc tính khuôn mặt chi tiết. Bộ dữ liệu này bao gồm:

- 202.599 hình ảnh khuôn mặt có độ phân giải **128x128** ở định dạng JPEG.
- 10.177 định danh (identities), tuy nhiên thông tin chi tiết không được công bố.
- 40 thuộc tính nhị phân (binary attributes) như màu tóc, có râu hay không, đeo kính hay không, v.v.
- 5 vị trí mốc (landmark locations) trên mỗi khuôn mặt.

Hình ảnh trong bộ dữ liệu bao gồm nhiều tư thế và các nền ảnh phức tạp, giúp đánh giá hiệu quả của mô hình trong các điều kiện thực tế. Trong phạm vi dự án này, nhóm chỉ tập trung khai thác các thuộc tính liên quan đến **identities**, **hình ảnh khuôn mặt**, và **thuộc tính khuôn mặt**.

### 3 Kiến trúc dự án

Hệ thống được xây dựng dựa trên kiến trúc tổng quan như Hình 1. Kiến trúc này bao gồm các bước chính:



Hình 1: *Kiến trúc tổng quan của hệ thống truy xuất hình ảnh.*

1. Xử lý dữ liệu và tiền xử lý ảnh đầu vào của người dùng, bao gồm các bước như cắt, chuẩn hóa và gán nhãn.
2. Huấn luyện và tinh chỉnh các mô hình học sâu, bao gồm ResNet-50 và MobileNet-V2, nhằm học các khía cạnh biểu diễn (embedding) phân biệt cho các khuôn mặt trên bộ dữ liệu CelebA.
3. Tích hợp KD-Tree để tăng hiệu quả truy vấn và giảm thời gian tính toán trong quá trình tìm kiếm.
4. Triển khai hệ thống truy xuất hình ảnh với giao diện đơn giản bằng framework Streamlit, giúp người dùng dễ dàng tìm kiếm và truy vấn hình ảnh.

# Chương 1

## Xử lý dữ liệu đầu vào

### 1.1 Tiềm xử lý dữ liệu

Bộ dữ liệu CelebA đã được xử lý crop mặt, tuy vậy, ảnh vẫn chưa thực sự sát với gương mặt. Vì vậy chúng ta sẽ thực hiện thêm 1 bước nữa để ảnh sát với mặt hơn, bằng cách sử dụng bộ lọc Haar [VJ01].

#### 1.1.1 Bộ lọc Haar (Haar Cascade Filter)

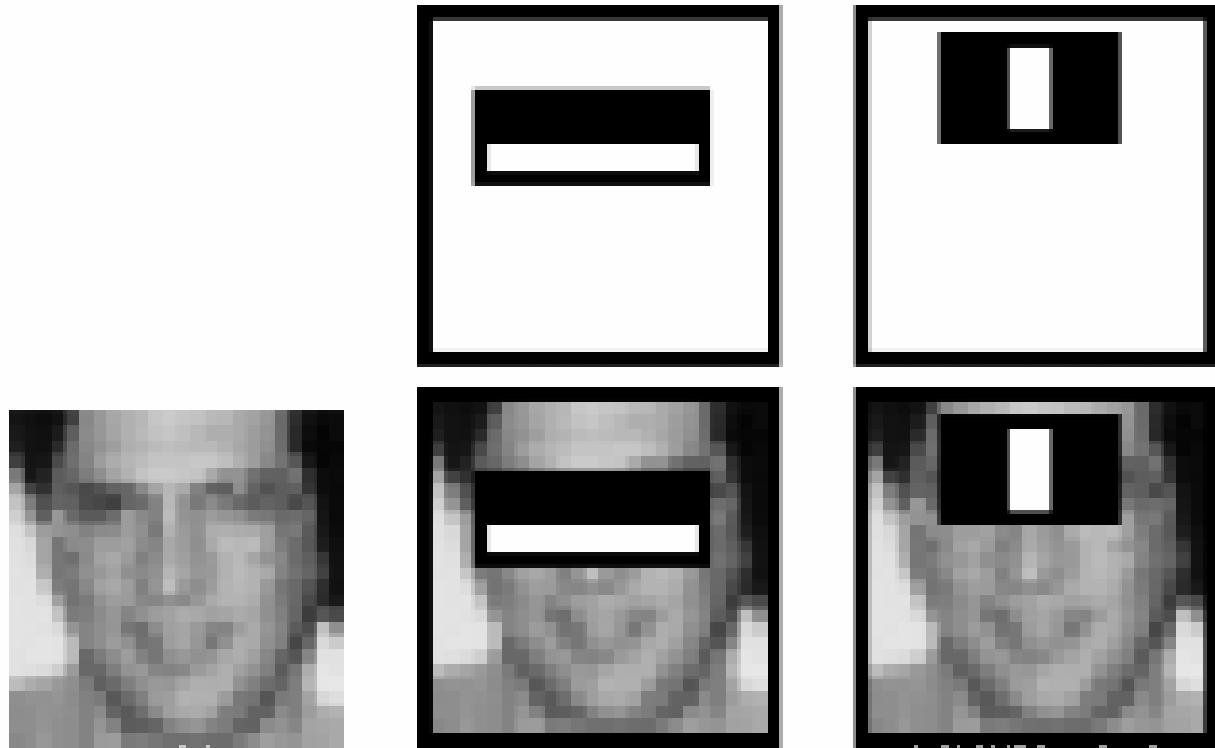
Bộ lọc Haar là một thuật toán học máy đã được huấn luyện từ trước, thường được sử dụng để nhận diện vùng chứa mặt trên 1 ảnh, bằng cách vẽ hộp bao quanh (bounding box) quanh mặt mà nó tìm được. Cách sử dụng nó tương tự với thuật toán vẽ bounding box của CNN, nhưng thay vì quét 1 thuật toán CNN trên từng vùng ảnh trên khắp ảnh đang xét, ta quét bằng 1 bộ lọc Haar.

#### Quá trình tiềm huấn luyện

Bản chất bộ lọc Haar là tổng hợp nhiều đặc trưng loại Haar, sau đó chạy theo kiểu thác nước (cascade). Đặc trưng loại Haar là một bộ lọc giống như bộ lọc của CNN. Điểm khác biệt nằm ở việc ở CNN, bộ lọc chiếm toàn bộ cửa sổ trượt, trong khi ở đặc trưng Haar, bộ lọc chỉ chiếm một phần cửa sổ trượt.

Hình 1.1 đưa ra 2 ví dụ của bộ lọc Haar. Bộ lọc thứ nhất tìm vùng phân biệt giữa mắt/lông mày với mũi, vì 2 vùng này có sự khác biệt đáng kể về mức độ sáng tối. Bộ lọc thứ hai tìm đường sống mũi, vì đường này sẽ sáng hơn so với 2 bên.

Tuy nhiên, số lượng bộ lọc Haar là rất nhiều, hơn **160.000** bộ lọc. Vì vậy, tác giả sử dụng AdaBoost để tiềm huấn luyện các bộ lọc này và chọn ra các bộ lọc quan trọng nhất. Sau bước này, ta chỉ cần khoảng **6000** bộ lọc để nhận diện một gương mặt.



Hình 1.1: *Sự khác biệt giữa bộ lọc CNN và Haar*

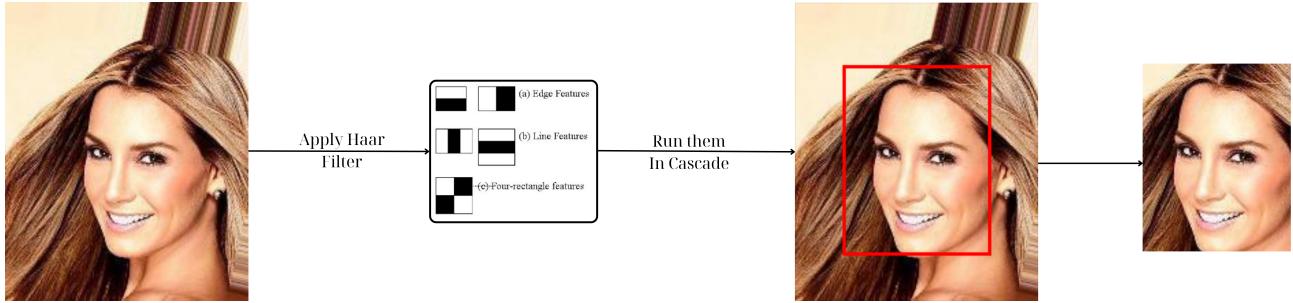
Tuy vậy, việc sử dụng 6000 bộ lọc trên 1 ảnh là rất tốn kém về mặt thời gian lẫn tài nguyên, nên thay vì chạy 6000 bộ lọc một lúc, ta chạy từng nhóm bộ lọc. Nếu một nhóm bộ lọc xác định là vùng này không chứa mặt, ta không xét tiếp các nhóm sau và dịch sang vùng kế tiếp.

### 1.1.2 Tiền xử lý với Bộ lọc Haar

Trong bước này, chúng ta tiền xử lý từng ảnh của bộ dữ liệu CelebA bằng bộ lọc Haar như sau:

1. Khởi tạo bộ lọc Haar đã được tiền huấn luyện, sử dụng file XML được cung cấp ở <https://s.net.vn/c4iV>.
2. Ta chạy ảnh qua bộ lọc Haar đã được tiền huấn luyện. Nếu ảnh quét ra 1 mặt, ta trả về vùng chỉ chứa mặt mà ta nhận diện được. Nếu ảnh quét ra nhiều hơn 1 hoặc không quét ra, ta sang bước sau.
3. Chia nửa bước quét (cả chiều rộng và chiều cao) và chạy ảnh qua bộ lọc Haar lại. Nếu ảnh quét ra 1 mặt, ta trả về cùng chứa mặt, nếu không, ta bỏ ảnh.

Bộ dữ liệu CelebA ban đầu có tất cả **202.599** tấm ảnh. Sau quá trình xử lý này, ta còn lại **196.791** tấm ảnh. Tham khảo thêm về cách hoạt động của Bộ lọc Haar ở Hình 1.2.



Hình 1.2: Ví dụ về việc xử lý bằng bộ lọc Haar.

## 1.2 Xử lý dữ liệu cho mô hình

Trong bước này, chúng ta tập trung vào việc chuẩn bị một tập dữ liệu có cấu trúc chặt chẽ, đủ tin cậy, và phù hợp cho quá trình huấn luyện với Triplet Loss. Trong phần này, chúng tôi muốn nhấn mạnh cách xây dựng bộ ba *Anchor*, *Positive*, *Negative* để mô hình học cách nhận biết khuôn mặt *đúng* (cùng một người) và *khác* (khác người đó). Dưới đây là ý tưởng tổng quát, giúp bạn đọc nắm bắt được phương pháp xử lý dữ liệu sơ lược:

### a) Nhắc lại mục đích của *Anchor*, *Positive*, và *Negative*:

- *Anchor* và *Positive* đều thuộc **cùng** một danh tính (ID), giúp mô hình hiểu rằng hai ảnh này cần được ánh xạ gần nhau trong không gian đặc trưng (embedding space).
- *Negative* là ảnh **khác** danh tính (khác ID), để mô hình học cách “đẩy” cặp *Anchor* - *Negative* ra xa trong không gian đặc trưng.

Cơ chế này giúp *phân cụm* ảnh của cùng một người và *tách biệt* ảnh của những người khác nhau, tạo tiền đề cho khả năng nhận diện khuôn mặt *chính xác*.

### b) Xác định danh tính và lọc dữ liệu:

Mỗi ảnh trong bộ dữ liệu CelebA đều đi kèm một *mã danh tính* (ID), xem thêm ở Hình 1.3. Tôi chỉ giữ lại những ảnh *thực sự tồn tại* (không bị hỏng hoặc thiếu) trong thư mục dữ liệu. Bằng cách loại bỏ các ảnh không hợp lệ, ta hạn chế tối đa tình trạng sai lệch trong quá trình huấn luyện.

### c) Tổ chức lại tập ảnh theo danh tính:

Sau giai đoạn lọc, tôi sẽ nhóm tất cả ảnh của mỗi ID lại với nhau để thuận tiện cho việc chọn *Anchor* và *Positive* phía dưới. Các ID có số lượng ảnh quá ít (thường là dưới 2) sẽ bị loại, vì không thể hình thành cặp *Anchor* - *Positive*.

### d) Chia tập Train/Test dựa trên ID:

Ta chọn một tỉ lệ (ví dụ 80%) danh tính để làm *train identities*, phần còn lại (20%) làm *test identities*. Cách



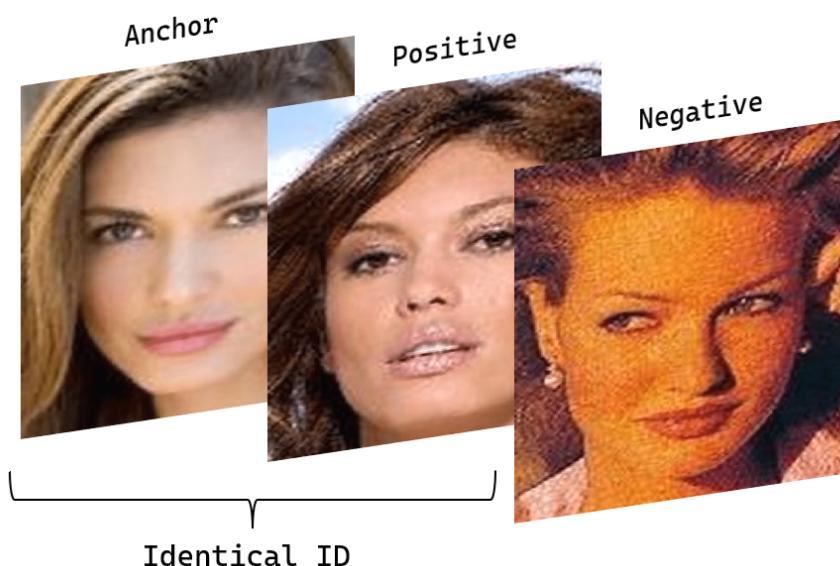
Hình 1.3: Hình ảnh minh họa chân dung đi kèm với ID (Identity) của 3 người.

chia này **ngăn** việc cùng một khuôn mặt (cùng ID) xuất hiện ở cả hai tập Train và Test, *tránh rò rỉ dữ liệu* và đảm bảo tính tổng quát khi đánh giá mô hình.

e) **Sinh bộ ba (Anchor, Positive, Negative) cho huấn luyện:**

- Từ mỗi ID trong tập Train, tôi sẽ chọn hai ảnh ngẫu nhiên làm *Anchor* và *Positive*. Như đã đề cập, hai ảnh này là cùng người, tạo tiền đề cho việc *đưa* hai mẫu gần nhau trong không gian đặc trưng.
- *Negative* được chọn từ một ID **khác** (một người khác), *đảm bảo* mô hình học cách *phân biệt* khuôn mặt khác người. Nếu không tìm được ảnh hợp lệ cho *Negative* (vì ID khác không có ảnh), tôi sẽ *loại* bộ ba đó khỏi dữ liệu huấn luyện.

Tham khảo tại Hình 1.4.

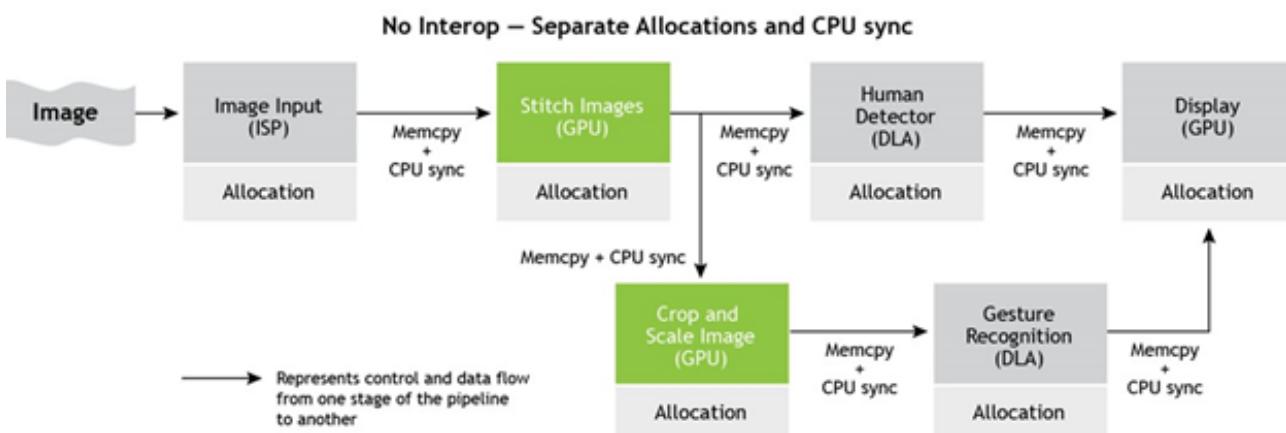


Hình 1.4: Minh họa một bộ ba Triplet Set trong tập huấn luyện.

f) Tạo bộ truy vấn (Query) và thư viện (Gallery) cho kiểm thử:

- Với tập Test, tôi sẽ chọn *một* ảnh từ mỗi ID làm Query.
- Những ảnh còn lại của cùng ID *được đưa* vào Gallery. Cách làm này cho phép ta kiểm tra khả năng của mô hình trong việc nhận dạng Query trong thư viện ảnh của cùng người đó. Ngoài ra, tôi cũng mở rộng *Gallery* với các ảnh từ những ID khác để đánh giá năng lực mô hình trong bài toán nhận diện hoặc tìm kiếm khuôn mặt *khác* với người muốn truy vấn.

g) **Tối ưu hiệu năng xử lý dữ liệu:** Do số lượng ảnh lớn, tôi sẽ sử dụng thêm kỹ thuật tính toán *đa luồng* (multi-threading) để phân chia công việc, từ đó tăng tốc độ quét và tạo các bộ *triplet*. Nhờ vậy, thời gian xử lý dữ liệu được rút ngắn đáng kể, hỗ trợ chúng ta tập trung vào các giai đoạn huấn luyện và đánh giá mô hình. Tham khảo thêm quy trình xử lý ở Hình 1.5.



Hình 1.5: Minh họa việc xử lý dữ liệu ảnh dựa trên đa luồng.

**Tóm lại**, bằng việc **kết hợp** các bước lọc ảnh, nhóm theo ID, chia Train/Test, rồi sinh bộ ba (Anchor, Positive, Negative), chúng ta có thể bảo đảm rằng mô hình *hiểu* và *học* được sự khác biệt giữa **các khuôn mặt cùng ID** và **khác ID**. Đây là tiền đề quan trọng để bước sang giai đoạn huấn luyện với *Triplet Loss*, giúp mô hình nắm bắt khái niệm tương đồng khuôn mặt và phân tách các khuôn mặt khác nhau một cách hiệu quả.

### 1.3 Chuẩn bị dữ liệu và các phép biến đổi (Transform)

Trong bước này, chúng tôi đề xuất một số phương pháp tiền xử lý và biến đổi ảnh (*transform*) nhằm giúp mô hình huấn luyện tốt hơn, đặc biệt trong bài toán *Triplet Loss*. **Các tham số** liên quan được lựa chọn như sau:

- **Kích thước ảnh (Image Size):** Dựa toàn bộ ảnh về độ phân giải  $218 \times 218$  (theo như cách xử lý ở Chương 1). Quy ước này đảm bảo *tính nhất quán* và phù hợp với nhiều mô hình CNN đã tiền huấn luyện trên ImageNet.

- **Chuẩn hoá trung bình ( $\mu$ ) và độ lệch chuẩn ( $\sigma$ ):**

$$\mu = [0.485, 0.456, 0.406], \quad \sigma = [0.229, 0.224, 0.225]$$

Với  $\mu$  là trung bình của các kênh màu R, G, B và  $\sigma$  là độ lệch chuẩn của các kênh màu R, G, B.

Các giá trị này giúp ảnh *tiệm cân* phân phối dữ liệu gốc trên bộ ImageNet, nâng cao hiệu quả *transfer learning*. Tham khảo thêm tại <https://shorturl.at/du67A>.

- **Tỉ lệ xoá ngẫu nhiên (Random Erasing):**

- *Xác suất thực hiện ( $p$ )* đặt là 0.5. Có nghĩa trung bình 50% số ảnh trong mỗi batch sẽ bị xoá một vùng nào đó.
- *Quy mô vùng xoá (scale)* khoảng (0.02, 0.33) và *tỉ lệ cạnh (ratio)* từ (0.3, 3.3). Ta sẽ nói thêm một chút về *ratio*: Tham số này thường được biểu diễn dưới dạng một khoảng giá trị  $[r_{min}, r_{max}]$ , nếu giá trị *ratio* gần 1.0, vùng bị xoá có hình dạng gần vuông; nếu giá trị *ratio* lệch xa 1.0, vùng bị xoá sẽ có hình dạng chữ nhật dài hoặc dẹt. Vậy nghĩa là ở khoảng (0.3, 3.3) thì vùng xoá có thể rất hẹp và cao (0.3), và có thể rất rộng và thấp (3.3).

Với cách làm này, mô hình học cách *thích nghi* khi khuôn mặt bị che khuất, tránh phụ thuộc quá mức vào một khu vực đặc trưng.

- **Loại bỏ ngẫu nhiên (Dropout) vùng ảnh:**

- *Xác suất ( $p$ )* thường được đặt là 0.3, nghĩa là có 30% cơ hội một vùng ảnh (theo lựa chọn của hàm Dropout) bị che.
- Kỹ thuật này *buộc* mô hình chú trọng toàn bộ gương mặt, thay vì tập trung quá nhiều vào một chi tiết.

Dựa trên các **tham số** cụ thể ở trên, chúng tôi xây dựng quy trình *transform* cho **tập huấn luyện** và **tập kiểm thử** như sau:

- a) **Thông nhất kích thước:** Mỗi ảnh được *Resize* về  $218 \times 218$ , **giữ** cùng tỉ lệ chiều rộng - chiều cao (nếu muốn), hoặc cắt giữa ảnh trước khi thay đổi kích thước.

b) **Biến ảnh sang dạng số hoá (Tensor) và chuẩn hoá giá trị:** Sau khi cắt, chúng tôi chuyển sang **tensor** và *normalize* bằng các giá trị trung bình  $\mu$  và độ lệch chuẩn  $\sigma$  như trên. Ý tưởng này giúp *giữ* cho dữ liệu đầu vào gần gũi với tập dữ liệu ban đầu mà mô hình được tiền huấn luyện, tận dụng hiệu quả chiến lược *transfer learning*.

c) **Tăng cường dữ liệu (Data Augmentation) cho huấn luyện:**

- *Xoá ngẫu nhiên một phần ảnh (Random Erasing* với  $p = 0.5$ ,  $scale = (0.02, 0.33)$ ,  $ratio = (0.3, 3.3)$ ) để mô phỏng tình huống khuôn mặt bị che khuất hoặc ảnh bị nhiễu.
- *Loại bỏ ngẫu nhiên một vùng ảnh (Dropout* với  $p = 0.3$ ) cũng là một cách khác để mô hình khái quát tốt hơn.

Nhờ các phép này, mỗi epoch huấn luyện sẽ tạo ra *phiên bản khác nhau* của ảnh, *kích thích* mô hình học tính *bất biến* trước nhiều điều kiện chụp. Tham khảo tại [Hình 1.6](#).



Hình 1.6: *Transformed Image* dựa trên các phép biến đổi ảnh.

d) **Giữ dữ liệu kiểm thử ở trạng thái “trung tính”:** Trong tập kiểm thử, chúng tôi *không* áp dụng các phép biến đổi ngẫu nhiên như *Random Erasing* hay *Dropout*. Thay vào đó, chỉ thực hiện **các bước cơ bản** như *Resize* và *Normalize* để kết quả *khách quan*, sát với môi trường triển khai thực tế.

e) **Các tham số cho việc cài đặt Dataset:**

- **Batch size:** Kích thước *batch* được lựa chọn khác nhau giữa tập huấn luyện và kiểm thử. Trong huấn luyện, *batch size* được đặt lớn để tận dụng tối đa dữ liệu và tăng tốc độ học. Ví dụ,  $N = 64$  hoặc  $128$  mẫu trong một *batch*. Trong kiểm thử, *batch size* được đặt nhỏ hơn (ví dụ,  $N = 16$  hoặc  $32$ ) nhằm giảm tải bộ nhớ và tăng tốc độ xử lý.
- **Chức năng ghép dữ liệu:** Chức năng này chịu trách nhiệm tổ chức dữ liệu trong một *batch* theo đúng định dạng yêu cầu. Đối với bài toán *Triplet Loss*, mỗi *batch* cần bao gồm các bộ ba (**a**, **p**, **n**), trong đó:

$$\mathbf{a} \in \mathbb{R}^{C \times H \times W}, \quad \mathbf{p} \in \mathbb{R}^{C \times H \times W}, \quad \mathbf{n} \in \mathbb{R}^{C \times H \times W}.$$

Chức năng này sẽ thực hiện các tác vụ chính sau:

- **Tổ chức dữ liệu:** Ghép các bộ ba ( $\mathbf{a}_i, \mathbf{p}_i, \mathbf{n}_i$ ) từ tập dữ liệu thành một *batch* với kích thước:

$$\text{Batch}_{\text{triplet}} = \left( \{\mathbf{a}_i\}_{i=1}^N, \{\mathbf{p}_i\}_{i=1}^N, \{\mathbf{n}_i\}_{i=1}^N \right),$$

trong đó  $N$  là kích thước *batch*.

- **Xử lý độ dài không đồng nhất:** Trong trường hợp dữ liệu không đồng nhất về kích thước (ví dụ, ảnh có độ phân giải khác nhau), chức năng này sẽ thực hiện căn chỉnh dữ liệu (bằng cách *padding*) để đảm bảo tất cả ảnh trong *batch* có cùng kích thước.

#### f) Ý nghĩa đối với Triplet Loss:

- Với *Anchor*, *Positive*, *Negative*, chúng tôi áp dụng chung **một** quy trình transform (nhưng vẫn có yếu tố *ngẫu nhiên* ở bước xoá, dropout) để duy trì sự công bằng giữa ba ảnh.
- Nhờ vậy, mô hình dần học được *cách phân biệt* các khuôn mặt khác danh tính, đồng thời *gắn gần* hơn với những khuôn mặt cùng danh tính.

**Tóm lại**, quá trình này được thực hiện như sau:

1. *Chuẩn hoá* kích thước ( $218 \times 218$ ) và *normalize* giá trị pixel với  $\mu, \sigma$  như ở trên, *đảm bảo khả năng tương thích* với mạng nơ-ron đã tiền huấn luyện.
2. *Các phép biến đổi ngẫu nhiên* (*Random Erasing*, *Dropout...*) chủ yếu được dùng trong huấn luyện nhằm *tạo đa dạng* dữ liệu và *chống quá khớp* (*overfitting*).
3. *Tạo ra các tập Datasets* với tham số được chỉ định, chuẩn bị cho việc tinh chỉnh mô hình.

Nhờ những bước chuẩn bị này, kết hợp với *chiến lược chia dữ liệu* và *tổ chức triplet* đã nêu ở mục 1.2, chúng tôi xây dựng được nền tảng vững chắc cho dữ liệu, sẵn sàng *tiến tới* giai đoạn huấn luyện mô hình với *Triplet Loss*.

# Chương 2

## Mô hình và các kỹ thuật liên quan

### 2.1 Giới thiệu mô hình

#### 2.1.1 ResNet [He+15]

1. **Motivations:** Mục tiêu chính của ResNet là giải quyết các vấn đề lớn trong các mạng neural sâu truyền thống, đặc biệt là khi độ sâu của mạng tăng lên. Những vấn đề này bao gồm:

- **Vanishing Gradient và Exploding Gradient:** Khi mạng nơ-ron trở nên sâu hơn (với số lượng lớp tăng đáng kể), giá trị gradient được lan truyền ngược qua nhiều lớp có thể:
  - *Rất nhỏ (Vanishing Gradient):* Làm giảm khả năng cập nhật trọng số, dẫn đến mạng không học được.
  - *Rất lớn (Exploding Gradient):* Làm trọng số cập nhật không ổn định, gây ra sự phân kỳ trong quá trình huấn luyện.

Đây là vấn đề phổ biến trong các mạng như VGGNet hoặc ZFNet khi độ sâu mạng vượt quá giới hạn nhất định.

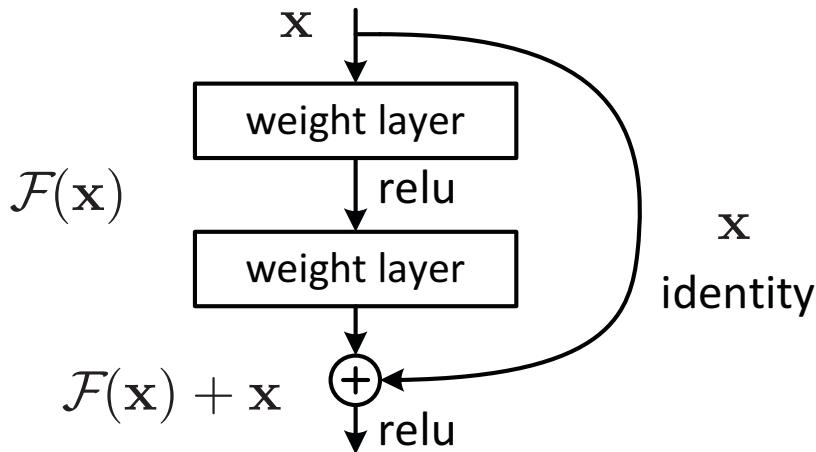
- **Degradation Problem (Vấn đề suy giảm hiệu suất):** Một mạng sâu hơn không phải lúc nào cũng cải thiện hiệu suất so với mạng nông hơn. Thậm chí, trong nhiều trường hợp, độ sâu mạng tăng lên lại dẫn đến hiệu suất suy giảm, ngay cả khi overfitting không xảy ra.

2. **Skip Connections:** Để khắc phục những vấn đề trên, ResNet giới thiệu một khái niệm đột phá: **Skip Connections** hay **Shortcut Connections**. Thay vì buộc các lớp trung gian phải học toàn bộ ánh xạ đầu vào ( $H(x)$ ), ResNet chuyển mục tiêu thành học một *hàm dư thừa (residual function)*  $F(x)$ , với mối quan hệ (xem thêm ở Hình 2.1):

$$H(x) = F(x) + x,$$

trong đó:

- $H(x)$ : Hàm ánh xạ mong muốn mà mạng cần học.
- $F(x)$ : Hàm dư thừa (*residual function*), được kỳ vọng giúp mô hình dễ học hơn do gradient có thể lan truyền trực tiếp qua các kết nối tắt (*skip connections*).
- $x$ : Đầu vào từ lớp trước.



Hình 2.1: *Residual learning: a building block.*

Điều này có hai lợi ích chính:

- **Truyền ngược gradient hiệu quả hơn:** Với kết nối tắt, gradient từ lớp đầu ra cuối cùng có thể lan truyền trực tiếp ngược về lớp đầu tiên mà không phải đi qua toàn bộ các lớp trung gian. Điều này giảm thiểu hiện tượng *vanishing gradient*.
- **Học ánh xạ đồng nhất dễ dàng hơn:** Nếu các lớp trung gian không đóng góp nhiều vào việc cải thiện ánh xạ, mô hình vẫn có thể học được  $H(x) \approx x$ , tức là chỉ cần sao chép đầu vào đến đầu ra mà không làm giảm hiệu suất.

3. **Lợi ích của Skip Connections:** Giả sử gradient lan truyền qua một lớp thông thường mà không có skip connections, công thức lan truyền sẽ là:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial H(x)} \cdot \frac{\partial H(x)}{\partial x},$$

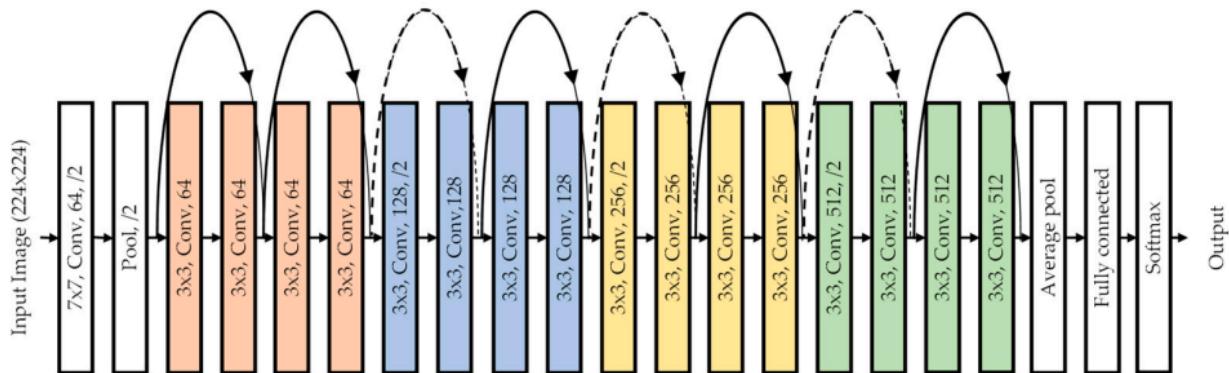
trong đó  $\frac{\partial H(x)}{\partial x}$  có thể nhỏ hơn 1, dẫn đến việc gradient giảm dần qua nhiều lớp.

Khi có skip connections, gradient được tính như sau:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial (F(x) + x)} = \frac{\partial \mathcal{L}}{\partial F(x)} \cdot \frac{\partial F(x)}{\partial x} + \frac{\partial \mathcal{L}}{\partial x}.$$

Thuật ngữ bổ sung  $\frac{\partial \mathcal{L}}{\partial x}$  giúp giữ lại gradient, giảm thiểu hiện tượng gradient biến mất.

Tham khảo cấu trúc của ResNet-18 tại Hình 2.2.



Hình 2.2: Cấu trúc ResNet-18. <https://s.net.vn/vxXV>.

#### 4. Tại sao ResNet phù hợp với bài toán?

- **Khả năng học đặc trưng sâu sắc:** Trong bài toán nhận diện khuôn mặt, cần trích xuất các đặc trưng phân biệt mạnh mẽ, như hình dạng, kết cấu và chi tiết vùng mặt. Với độ sâu lớn, ResNet cho phép học các đặc trưng đa cấp độ, từ đặc trưng cục bộ (local features) ở các lớp đầu đến đặc trưng toàn cục (global features) ở các lớp sâu hơn, giúp mô hình phân biệt hiệu quả giữa các khuôn mặt khác nhau.
- **Tương thích với Triplet Loss:** *Triplet Loss* yêu cầu một không gian biểu diễn (*embedding space*) mà trong đó:

$$d(f(\mathbf{a}), f(\mathbf{p})) \ll d(f(\mathbf{a}), f(\mathbf{n})),$$

với  $d(\cdot, \cdot)$  là khoảng cách giữa các vector biểu diễn. ResNet, nhờ *residual blocks*, đảm bảo gradient lan truyền ổn định qua mạng sâu, giúp học tốt ánh xạ không gian biểu diễn và duy trì cấu trúc khoảng cách cần thiết.

- **Khả năng tận dụng trọng số tiền huấn luyện (*Pretrained Weights*):** ResNet được huấn luyện trước trên ImageNet, một tập dữ liệu với hơn một triệu ảnh và 1000 danh mục, giúp mô hình học được các đặc trưng phổ quát (universal features). Trong bài toán hiện tại, việc sử dụng trọng số tiền huấn luyện giúp:
  - Tăng tốc độ hội tụ trong quá trình huấn luyện.
  - Cải thiện hiệu năng nhờ tận dụng đặc trưng đã học từ ImageNet, đặc biệt trong trường hợp kích thước tập huấn luyện không quá lớn.

- **Khả năng tổng quát hóa tốt:** Trong bài toán *Face Image Retrieval*, khả năng tổng quát hóa là yếu tố then chốt để nhận diện hoặc phân biệt các khuôn mặt mới (chưa xuất hiện trong tập huấn luyện). ResNet, nhờ *skip connections*, giúp giảm hiện tượng overfitting và cải thiện độ chính xác trên tập kiểm thử.

### 2.1.2 MobileNet [How+17]

1. **Motivations:** MobileNet được thiết kế với mục tiêu tối ưu hóa hiệu năng và giảm thiểu yêu cầu tính toán, đặc biệt trên các thiết bị di động và nhúng. Những vấn đề mà MobileNet giải quyết bao gồm:

- **Yêu cầu tài nguyên thấp:** Các mạng sâu truyền thống như ResNet hay VGG yêu cầu tài nguyên lớn (bộ nhớ và thiết bị), làm hạn chế khả năng triển khai trên các thiết bị di động.
- **Độ phức tạp tính toán cao:** Số lượng tham số và phép nhân ma trận trong các mô hình cũ thường quá lớn, dẫn đến thời gian suy luận chậm và khó triển khai trong thực tế.

2. **Depthwise Separable Convolution:** MobileNet giới thiệu một cải tiến quan trọng, *Depthwise Separable Convolution*, để giảm đáng kể độ phức tạp tính toán mà vẫn duy trì hiệu năng. Quá trình này tách một phép tích chập thông thường thành hai bước:

- (a) **Depthwise Convolution:** Trong bước này,  $M$  kênh đầu vào được xử lý độc lập bằng  $M$  kernel  $1 \times D_K \times D_K$  khác nhau, thay vì áp dụng  $N$  kernel  $M \times D_K \times D_K$  trên toàn bộ các kênh. Độ phức tạp tính toán của bước này là:

$$\mathcal{O}(D_K^2 \cdot M \cdot D_F^2),$$

so với

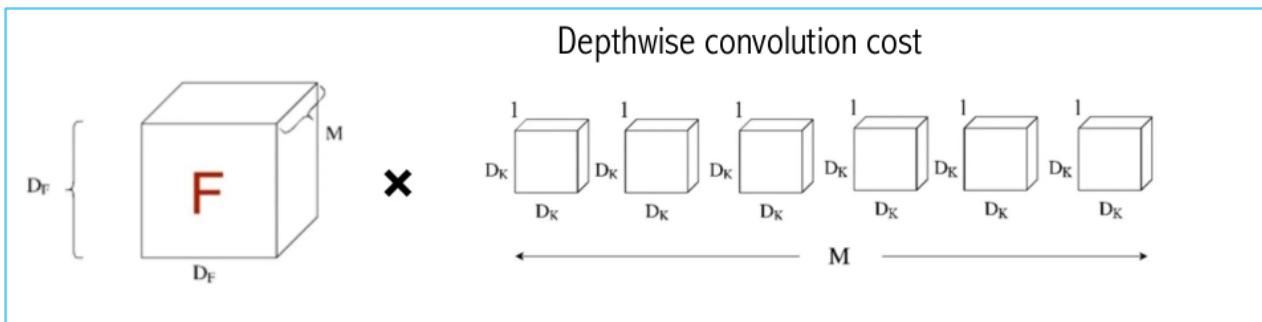
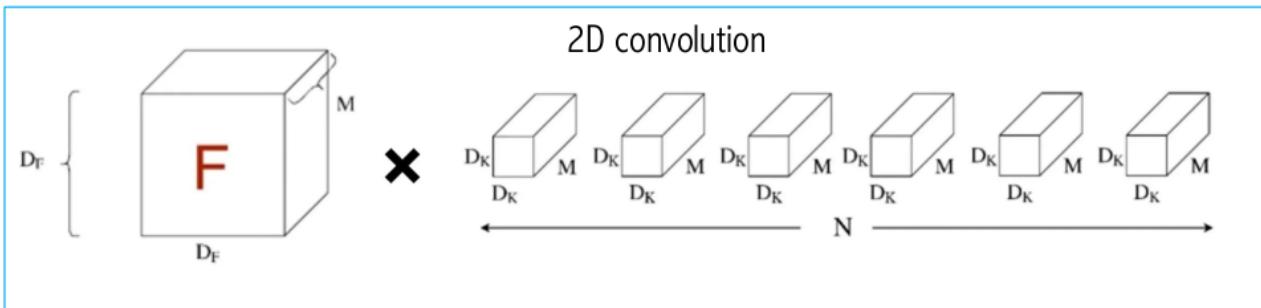
$$\mathcal{O}(D_K^2 \cdot M \cdot N \cdot D_F^2),$$

với  $D_F$  ứng với độ lớn của kênh. Xem thêm ở [Hình 2.3](#).

- (b) **Pointwise Convolution:** Sau khi trích xuất đặc trưng không gian từ từng kênh, bước này sử dụng  $N$  tích chập  $M \times 1 \times 1$  để kết hợp các kênh đầu vào, tạo thành các kênh đầu ra mong muốn (trong trường hợp này là  $N$ ). Độ phức tạp của bước này là:

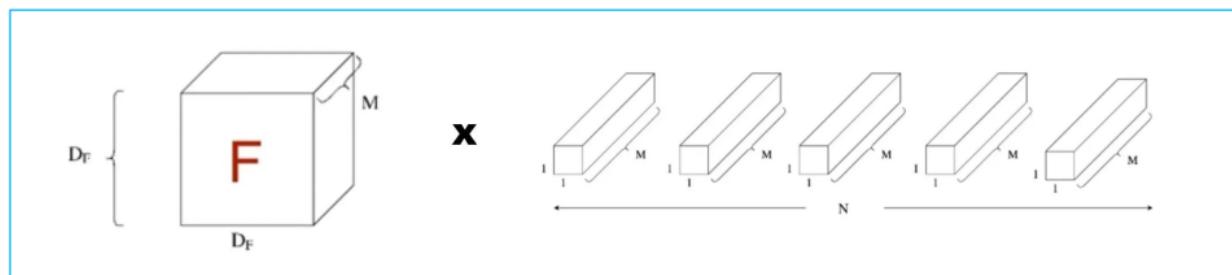
$$\mathcal{O}(M \cdot N \cdot D_F^2).$$

Tham khảo thêm ở [Hình 2.4](#).



Hình 2.3: *2D Convolution vs. Depthwise Convolution.*

#### Point-wise Convolution



Hình 2.4: *Pointwise Convolution.*

Như vậy, so với tích chập thông thường với độ phức tạp  $\mathcal{O}(D_K^2 \cdot M \cdot N \cdot D_F^2)$ , MobileNet đã giảm đáng kể xuống còn:

$$\mathcal{O}(D_K^2 \cdot M \cdot D_F^2 + M \cdot N \cdot D_F^2),$$

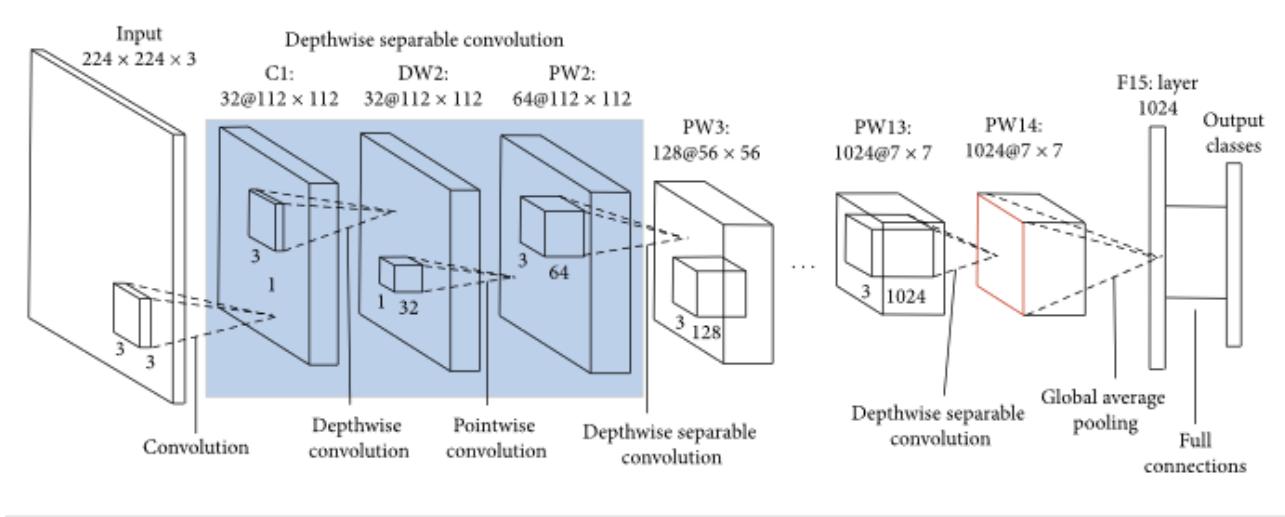
trong đó  $D_K$  là kích thước kernel,  $M$  là số kênh đầu vào, và  $N$  là số kênh đầu ra.

Tham khảo cấu trúc MobileNet ở [Hình 2.5](#).

### 3. Width Multiplier và Resolution Multiplier:

MobileNet sử dụng hai tham số quan trọng, *Width Multiplier* và *Resolution Multiplier*, để điều chỉnh kích thước mô hình và giảm chi phí tính toán một cách linh hoạt, tùy thuộc vào tài nguyên phần cứng và yêu cầu bài toán.

- **Width Multiplier ( $\alpha$ ):** Tham số này được sử dụng để giảm số lượng kênh trong các tầng của mô hình. Với mỗi giá trị  $0 < \alpha \leq 1$ , số lượng



Hình 2.5: Cấu trúc MobileNet. <https://s.net.vn/uHh0>.

kênh trong mỗi tầng được giảm tỉ lệ theo:

$$M' = \alpha M \text{ và } N' = \alpha N,$$

trong đó  $M$  là số kênh đầu vào gốc và  $M'$  là số kênh sau khi áp dụng *Width Multiplier*. Điều này dẫn đến việc giảm số lượng tham số và phép tính toán trong tích chập, giúp mô hình nhẹ hơn, nhưng có thể làm giảm hiệu năng khi  $\alpha$  quá nhỏ.

- **Resolution Multiplier ( $\rho$ ):** Tham số này được dùng để giảm độ phân giải không gian ( $D_F$ ) của ảnh đầu vào. Nếu ảnh gốc có kích thước  $H \times W$ , sau khi áp dụng  $\rho$ , kích thước được giảm xuống:

$$H' = \rho H, \quad W' = \rho W,$$

với  $0 < \rho \leq 1$ . Điều này làm giảm số lượng phép toán cần thiết trong mỗi lớp convolution, vì độ phức tạp tính toán phụ thuộc vào diện tích không gian  $D_F \times D_F$ .

Tóm lại, sau khi áp dụng các hệ số  $\alpha$  và  $\rho$ , độ phức tạp của mô hình MobileNet lúc này chỉ còn

$$\mathcal{O}(D_K^2 \cdot \alpha M \cdot \rho^2 D_F^2 + \alpha^2 \cdot M \cdot N \cdot \rho^2 D_F^2), \quad \alpha, \rho \in (0, 1].$$

Đây là giá trị khá hợp lý trong trường hợp xử lý dữ liệu hình ảnh lớn như CelebA và bị giới hạn thiết bị để huấn luyện như phần cứng, mô hình MobileNet sẽ làm giảm đáng kể lượng tính toán mà máy tính phải chịu và thời gian có được kết quả từ mô hình.

#### 4. Tại sao MobileNet phù hợp với bài toán?

- **Giảm chi phí tính toán:** MobileNet sử dụng *Depthwise Separable Convolution* để giảm độ phức tạp tính toán trong quá trình trích xuất đặc trưng. Điều này giúp giảm số phép toán cần thực hiện, đảm bảo khả năng suy luận nhanh mà không ảnh hưởng lớn đến hiệu suất, đặc biệt hữu ích trong các bài toán thời gian thực như nhận diện khuôn mặt.
- **Triển khai trên thiết bị hạn chế:** MobileNet được thiết kế để hoạt động trên các thiết bị nhúng hoặc di động với tài nguyên phần cứng giới hạn, như CPU hiệu năng thấp hoặc GPU nhỏ. Nhờ khả năng điều chỉnh cấu trúc bằng các tham số như *Width Multiplier* và *Resolution Multiplier*, MobileNet có thể được tối ưu hóa để đáp ứng các yêu cầu cụ thể mà vẫn đảm bảo hiệu năng.
- **Tối ưu hóa tài nguyên bộ nhớ:** MobileNet giảm số tham số trong mô hình, giúp tiết kiệm bộ nhớ, tăng tốc độ xử lý và giảm độ trễ, đặc biệt quan trọng trong các ứng dụng yêu cầu độ trễ thấp như tìm kiếm và nhận diện khuôn mặt.

## 2.2 Phương pháp Fine-tuning

### 2.2.1 Triplet-Loss

Triplet Loss là một trong những phương pháp học biểu diễn (*metric learning*) phổ biến, được giới thiệu trong lĩnh vực nhận diện khuôn mặt [SKP15]. Mục tiêu của nó là **học** ra một hàm ánh xạ (thường được biểu diễn bởi mạng deep neural) để chiếu mỗi ảnh vào *không gian đặc trưng* (*embedding space*) sao cho:

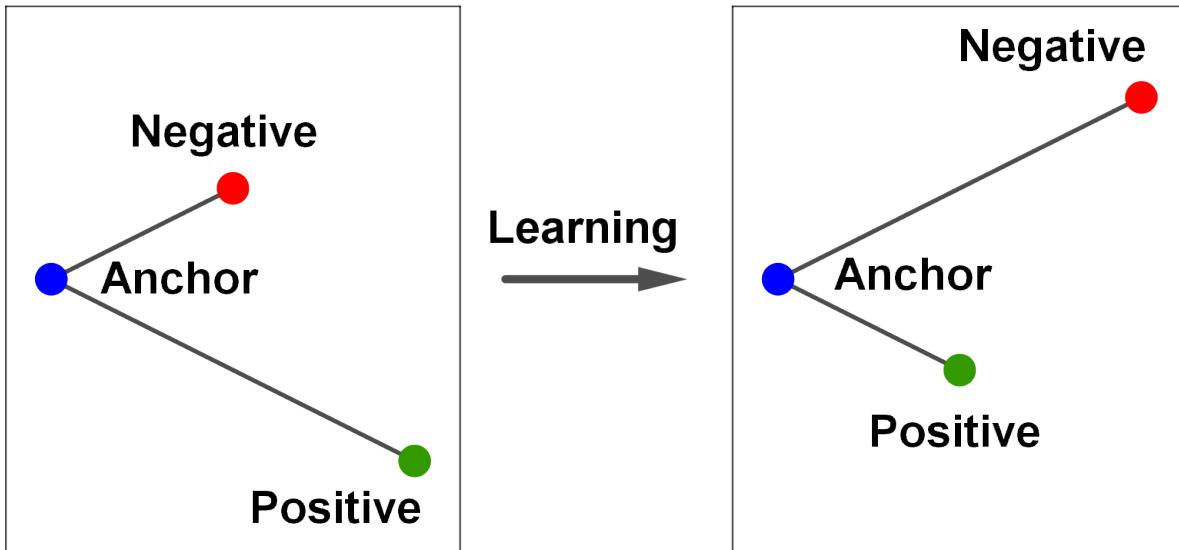
- Ảnh của cùng một danh tính (*anchor* và *positive*) có **khoảng cách** nhỏ (nằm gần nhau).
- Ảnh thuộc các danh tính khác nhau (*anchor* và *negative*) phải **nằm cách xa** nhau.

Xem thêm ở Hình 2.1.

### Ý tưởng cơ bản

Triplet Loss hoạt động với **bộ ba** (*triplet*), gồm:

1. **Anchor (a):** Ảnh đại diện (điểm neo) cho một danh tính nhất định.
2. **Positive (p):** Ảnh của *cùng* danh tính với Anchor.
3. **Negative (n):** Ảnh của *khác* danh tính so với Anchor.



Hình 2.6: *Hàm mất mát triplet giảm khoảng cách giữa điểm gốc (anchor) và điểm tích cực (positive), cả hai có cùng danh tính, đồng thời tăng khoảng cách giữa điểm gốc và điểm tiêu cực (negative) có danh tính khác.* [https://en.wikipedia.org/wiki/Triplet\\_loss](https://en.wikipedia.org/wiki/Triplet_loss).

Giả sử  $\mathbf{a}, \mathbf{p}, \mathbf{n}$  là đầu vào của mạng, và  $f(\mathbf{x})$  là hàm ánh xạ (embedding) mà mô hình **đang học**. Diển hình,  $\mathbf{a}, \mathbf{p}, \mathbf{n}$  có kích thước  $(C, H, W)$  (kênh màu, chiều cao, chiều rộng), còn  $f(\cdot)$  ánh xạ mỗi ảnh sang một vector  $\mathbf{z}$  trong không gian  $\mathbb{R}^d$ , với  $d$  là số chiều đặc trưng (embedding dimension) và thường nhỏ hơn số chiều của ảnh.

## Công thức Triplet Loss

Triplet Loss [HA15] được định nghĩa như sau:

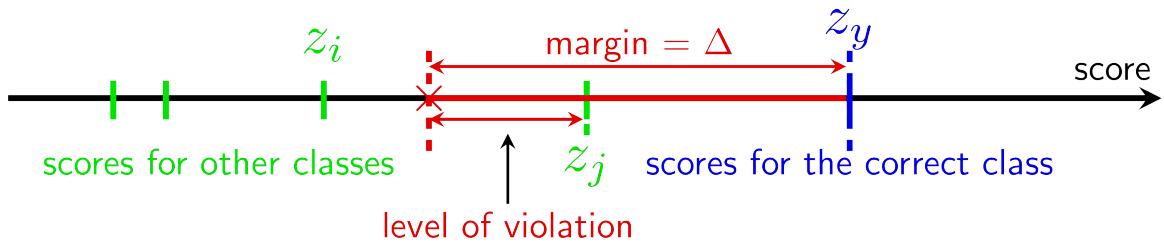
$$\mathcal{L}(\mathbf{a}, \mathbf{p}, \mathbf{n}) = \max\left(d(f(\mathbf{a}), f(\mathbf{p})) - d(f(\mathbf{a}), f(\mathbf{n})) + \alpha, 0\right) \quad (2.1)$$

trong đó:

- $d(\mathbf{u}, \mathbf{v})$  là **khoảng cách** giữa hai vector  $\mathbf{u}$  và  $\mathbf{v}$  trong không gian đặc trưng. Thông thường, người ta dùng khoảng cách Euclid hoặc  $l_2$ -norm:

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2.$$

- $\alpha$  là **margin** (biên), một hằng số dương nhằm duy trì *khoảng cách “an toàn”* giữa cặp  $\mathbf{a}, \mathbf{p}$  và cặp  $\mathbf{a}, \mathbf{n}$ . Ví dụ,  $\alpha$  có thể được đặt là 0.2, 0.3 hoặc 0.5, tùy thuộc vào kinh nghiệm hay thực nghiệm.



Hình 2.7: Xét ánh xạ  $f(\cdot)$  của các  $a, p, n$  là những embedding score với  $z_y$  là **Anchor**,  $z_i$  là **Negative** nhưng được đẩy ra khỏi vùng vi phạm (vùng mà nó được xem là **Positive**) và  $z_j$  cũng là **Negative** nhưng đang bị vi phạm, và  $\alpha$  là  $\Delta$ . Triplet Loss muốn score của correct point, được minh họa bởi điểm màu lam, cao hơn các scores khác, minh họa bởi các điểm màu lục, một khoảng cách an toàn  $\Delta$  là đoạn màu đỏ. Những scores khác nằm trong vùng an toàn (phía trái của dấu  $X$  màu đỏ) sẽ không gây ra mất mát gì, những scores nằm trong hoặc bên phải vùng màu đỏ đã vi phạm quy tắc và cần được xử phạt (hay cập nhật). Tham khảo thêm tại [Ti].

Mục đích của phương pháp là *đảm bảo*:

$$d(f(\mathbf{a}), f(\mathbf{p})) + \alpha < d(f(\mathbf{a}), f(\mathbf{n}))$$

tức là mẫu cùng danh tính (anchor và positive) phải *gần* nhau hơn so với mẫu khác danh tính (anchor và negative) ít nhất một khoảng  $\alpha$ . Khi bắt đầu thực hiện chưa được thoả, giá trị (2.1) sẽ dương, mô hình cần được cập nhật trọng số (qua backpropagation). Ngược lại, nếu đã thoả mãn (nghĩa là  $d(\mathbf{a}, \mathbf{p}) + \alpha > d(\mathbf{a}, \mathbf{n})$ ), mất mát ( $\mathcal{L}$ ) sẽ tiến về hoặc bằng 0 và mô hình không cần cập nhật tham số nữa. Xem thêm ví dụ ở Hình 2.7.

## Quy trình huấn luyện

Phần này sẽ được nói chi tiết hơn khi tới giai đoạn tinh chỉnh cho mô hình ở Mục 2.2, tuy nhiên tôi sẽ nói tóm gọn về quá trình huấn luyện để cho bạn đọc có thể nắm bắt khái quát:

1. **Chọn hoặc sinh bộ ba ( $\mathbf{a}, \mathbf{p}, \mathbf{n}$ )**. Dữ liệu được tổ chức thành các triplet:  $\mathbf{a}, \mathbf{p}$  cùng ID,  $\mathbf{n}$  khác ID. Việc *chọn* negative hợp lý (nhất là *hard negative*) có thể **đẩy nhanh** tốc độ hội tụ.
2. **Tính embedding** thông qua mạng  $f(\cdot)$ . Ta tiến hành lan truyền thuận (forward pass) để thu được  $f(\mathbf{a}), f(\mathbf{p}), f(\mathbf{n}) \in \mathbb{R}^d$ .
3. **Tính mất mát Triplet** theo (2.1) và thực hiện lan truyền ngược (back-propagation). Dựa vào  $\mathcal{L}$ , ta cập nhật trọng số mạng nơ-ron nhằm *giảm* giá trị hàm mất mát.

Cứ mỗi *epoch*, một loạt triplet mới được *xáo trộn* hoặc *tái sinh*, bảo đảm quá trình học *không* bị rơi vào tình trạng ghi nhớ (*overfit*) với một vài cặp mẫu duy nhất.

## Đánh giá và ứng dụng

- **Đánh giá:** Mô hình sau khi học xong sẽ ánh xạ ảnh khuôn mặt vào một *embedding* vector. Ta sẽ đánh giá mô hình dựa trên **tính chính xác** trong việc phân nhóm (clustering) hoặc *retrieval* (tìm kiếm khuôn mặt giống nhau) hoặc ảnh có đặc trưng trên khuôn mặt giống nhau. Ví dụ, ta so sánh khoảng cách  $\|f(\mathbf{a}) - f(\mathbf{p})\|$  và  $\|f(\mathbf{a}) - f(\mathbf{n})\|$  trên tập kiểm thử.
- **Ứng dụng:** Triplet Loss được sử dụng rộng rãi trong các bài toán *nhận diện khuôn mặt*, *xếp hạng ảnh* (*image ranking*), *nhận dạng chữ viết tay*, v.v... Trong nhận diện khuôn mặt, thiết kế Triplet Loss *giúp* mô hình học cách đưa mẫu của cùng một người vào *cùng “cụm”* trong không gian đặc trưng, *tách xa* các người khác.

## Nhận xét

Triplet Loss có một ưu điểm là **không** yêu cầu mô hình dự đoán trực tiếp “danh tính” của mỗi ảnh, mà chỉ cần **học** một *hàm khoảng cách phù hợp* (distance metric). Điều này giúp ta *linh động* hơn khi gặp những danh tính mới (chưa có trong tập huấn luyện). Khi mô hình đã học xong, ta *chỉ cần* tính *embedding* của ảnh mới và so sánh với *embedding* trong cơ sở dữ liệu, *không cần* huấn luyện lại.

### 2.2.2 Kỹ thuật tối ưu: Adam Optimizer [KB17]

Trong bài toán *Face Image Retrieval*, thuật toán **Adam (Adaptive Moment Estimation)** được sử dụng để tối ưu hóa trọng số của mô hình nhờ khả năng thích nghi và hiệu suất cao trong việc xử lý gradient. Adam kết hợp các ý tưởng từ *Momentum* (giúp tăng tốc độ hội tụ) và *RMSProp* (điều chỉnh tỉ lệ học dựa trên biến thiên của gradient) để tạo ra một thuật toán ổn định và hiệu quả. Các bước tính toán chính trong Adam bao gồm:

1. **Cập nhật moment bậc nhất và bậc hai:** Adam duy trì hai moment động:

- $m_t$  (moment bậc nhất): đại diện cho trung bình động (*exponential moving average*) của gradient.
- $v_t$  (moment bậc hai): đại diện cho trung bình động của bình phương gradient.

Công thức cập nhật:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2,$$

trong đó:

- $g_t = \nabla \mathcal{L}_t(w_t)$ : Gradient của hàm mất mát tại bước  $t$ .
- $\beta_1, \beta_2$ : Hệ số điều chỉnh ( $0 < \beta_1, \beta_2 < 1$ ), thường là  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ .

2. **Hiệu chỉnh thiên lệch:** Moment ban đầu có thiên lệch về không (do  $m_0 = 0, v_0 = 0$ ). Adam sử dụng hiệu chỉnh:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

3. **Cập nhật trọng số:** Sau khi hiệu chỉnh moment, trọng số  $w_t$  được cập nhật:

$$w_{t+1} = w_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

trong đó:

- $\eta$ : Tỉ lệ học (*learning rate*).
- $\epsilon$ : Giá trị rất nhỏ ( $10^{-8}$ ) để tránh chia cho 0.

## Tại sao Adam phù hợp cho bài toán?

- **Hội tụ nhanh và ổn định:** Adam tự động điều chỉnh tỉ lệ học cho từng tham số dựa trên moment bậc hai ( $v_t$ ), giúp hội tụ nhanh hơn và ổn định hơn trong các bài toán không gian biểu diễn, nơi gradient thường biến thiên không đồng đều.
- **Xử lý gradient nhỏ hoặc hiếm:** Trong bài toán *Triplet Loss*, các lớp *embedding* thường có gradient nhỏ. Vì vậy, Adam, nhờ cơ chế  $m_t$  và  $v_t$ , có thể xử lý tốt các gradient này, đảm bảo không bị mất thông tin quan trọng.
- **Khả năng thích nghi với dữ liệu lớn:** Với tập dữ liệu lớn như CelebA, gradient có thể dao động mạnh giữa các batch. Adam giảm thiểu sự dao động nhờ trung bình động, đảm bảo các cập nhật trọng số luôn ổn định.
- **Tương thích với Fine-tuning:** Trong giai đoạn fine-tuning, trọng số ban đầu của các *frozen layers* đã được học từ pre-training. Adam cho phép thực hiện các cập nhật nhỏ, chính xác mà không phá hủy các đặc trưng đã học, phù hợp với bài toán cần tinh chỉnh đặc trưng sâu như nhận diện khuôn mặt.

### 2.2.3 Thiết kế Embedding Layer

Việc xây dựng **Embedding Layer** trong mô hình truy hồi là một bước quan trọng để tối ưu hóa việc biểu diễn các đặc trưng của hình ảnh trong không gian thấp chiều. Dưới đây là lý do chi tiết về việc thiết kế và triển khai Embedding Layer:

#### Cấu trúc và mục tiêu của Embedding Layer

Embedding Layer được thiết kế để chuyển các đặc trưng đầu ra từ lớp trích xuất (backbone) thành một vector có kích thước cố định, phù hợp với bài toán truy xuất ảnh. Mục tiêu chính của Embedding Layer bao gồm:

- **Giảm chiều không gian:** Kích thước đầu ra của backbone, như MobileNetV2 hoặc ResNet-50, thường rất lớn, gây khó khăn cho các bước tính toán tiếp theo. Embedding Layer giúp giảm chiều không gian về kích thước nhỏ hơn, như 128 hoặc 256.
- **Tăng tính biểu diễn:** Bằng cách sử dụng các lớp Fully Connected (Linear Layer) với hàm kích hoạt phi tuyến (ReLU), mô hình có thể học được các biểu diễn đặc trưng tinh vi hơn.
- **Chuẩn hóa không gian embedding:** Không gian embedding giúp các vector có thể so sánh và tính toán khoảng cách trong các bài toán như tìm kiếm ảnh tương tự hoặc phân cụm.

#### Thiết kế Embedding Layer

Embedding Layer đóng vai trò quan trọng trong việc tối ưu hóa biểu diễn đặc trưng của hình ảnh và đảm bảo rằng các đầu ra của mô hình phù hợp với không gian tìm kiếm ảnh. Trong bài toán này, Embedding Layer được triển khai dưới dạng một chuỗi các lớp *Sequential* với thiết kế cụ thể như sau:

1. **Adaptive Average Pooling:** Lớp Adaptive Average Pooling được sử dụng để giảm kích thước không gian đầu ra của backbone xuống ( $1 \times 1$ ) và giữ nguyên số lượng kênh. Lớp này có vai trò tổng hợp thông tin toàn cục từ các đặc trưng không gian (spatial features) mà backbone sinh ra, giúp tạo ra biểu diễn toàn ảnh (global representation).

Công thức của Adaptive Average Pooling cho đầu vào là một tensor kích thước  $(N, C, H, W)$  được định nghĩa như sau:

$$y_{n,c} = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W x_{n,c,i,j},$$

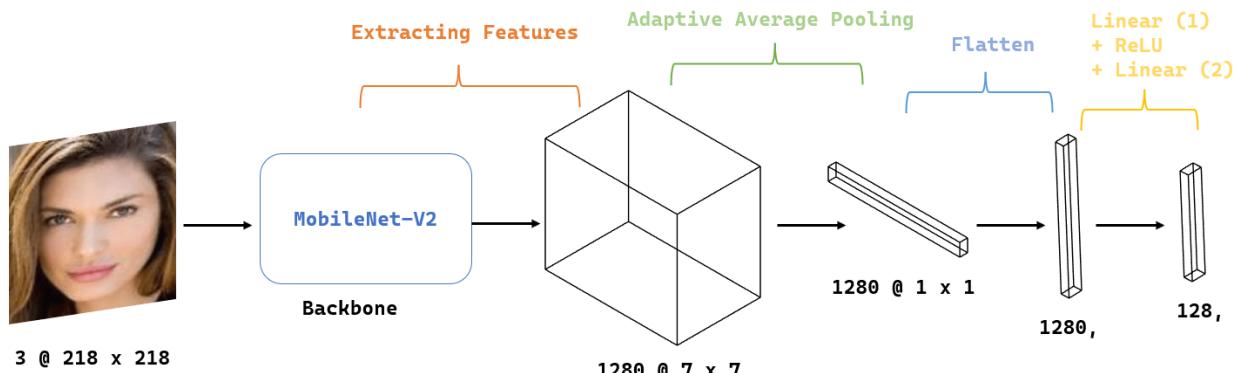
trong đó:

- $N$ : kích thước batch.
- $C$ : số lượng kênh đặc trưng.
- $H, W$ : chiều cao và chiều rộng của tensor đặc trưng đầu vào.
- $x_{n,c,i,j}$ : giá trị tại vị trí  $(i, j)$  trong kênh  $c$  của batch thứ  $n$ .
- $y_{n,c}$ : đầu ra của kênh  $c$  trong batch thứ  $n$  sau khi áp dụng Adaptive Average Pooling.

Kết quả đầu ra của lớp này có kích thước  $(N, C, 1, 1)$ , giảm toàn bộ thông tin không gian về một điểm duy nhất trên mỗi kênh.

2. **Flatten**: Lớp Flatten chuyển tensor từ kích thước  $(N, C, 1, 1)$  thành  $(N, C)$  để chuẩn bị cho các lớp Fully Connected phía sau. Đây là bước cần thiết để biến đổi đầu ra 4 chiều thành vector đặc trưng 2 chiều.
3. **Linear Layer (1)**: Lớp Fully Connected đầu tiên được sử dụng để giảm chiều từ số lượng đặc trưng đầu vào xuống kích thước không gian embedding. Đây là bước giảm chiều quan trọng nhằm tinh chỉnh và tối ưu hóa biểu diễn đặc trưng.
4. **ReLU**: Hàm kích hoạt phi tuyến ReLU (*Rectified Linear Unit*) được áp dụng để tăng khả năng biểu diễn của không gian embedding. Mục đích của ReLU không chỉ là loại bỏ các giá trị âm, mà còn tăng cường khả năng học của mô hình bằng cách tạo ra sự phi tuyến giữa các lớp.
5. **Linear Layer (2)**: Lớp Fully Connected thứ hai được sử dụng để tinh chỉnh thêm vector embedding và giữ nguyên kích thước đầu ra (embedding dimensions). Lớp này giúp mô hình học được các quan hệ phức tạp hơn trong không gian đặc trưng cuối cùng.

Tham khảo minh họa tại Hình 2.8.



Hình 2.8: Minh họa Embedding Block cho mô hình MobileNet-V2..

## 2.2.4 Fine-tuning cho mô hình

### ResNet-50

ResNet-50, một mô hình mạng nơ-ron sâu tiêu biểu với 50 lớp, được thiết kế để học các đặc trưng mạnh mẽ trên bộ dữ liệu ImageNet. Trong bài toán *Face Image Retrieval* dựa trên *Triplet Loss*, mô hình được điều chỉnh (*fine-tuned*) để tối ưu hóa không gian biểu diễn (*embedding space*), phục vụ cho việc phân biệt khuôn mặt. Hình 2.9 minh họa kiến trúc ResNet-50 và cách điều chỉnh để phù hợp với bài toán này. Dưới đây là chi tiết ý tưởng:

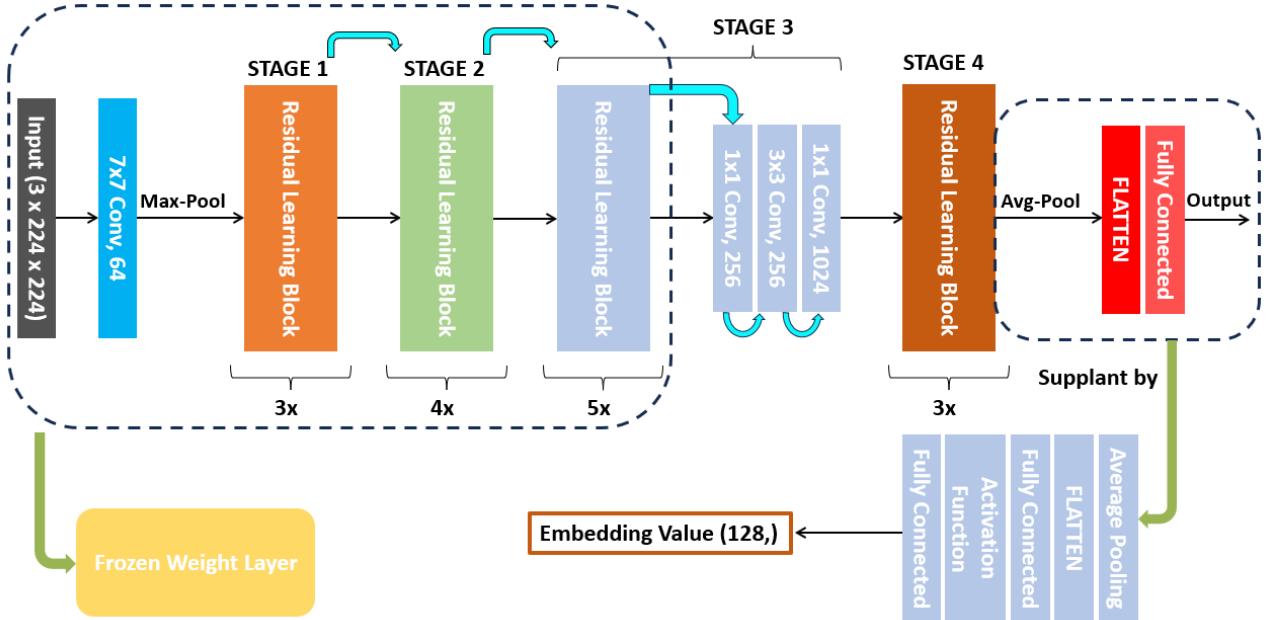
1. **Đóng băng (*Freezing*) các tầng dưới:** Các tầng ban đầu của ResNet-50, bao gồm các khối Residual Learning Block (Stage 1, 2, 3), giữ nguyên trọng số đã được huấn luyện trước (*pre-trained weights*). Điều này giúp tận dụng các đặc trưng phổ quát (universal features) đã học từ ImageNet, như đường viền, hình dạng, và kết cấu. Việc đóng băng cũng giảm số lượng tham số cần tối ưu, giúp tăng tốc độ huấn luyện.
2. **Điều chỉnh các tầng cao:**
  - Các khối Residual Learning Block ở Stage 4 và một khối ở Stage 3 được *unfreeze* để cập nhật trọng số (10 layers có trọng số). Điều này cho phép mô hình học các đặc trưng chuyên biệt hơn, liên quan đến khuôn mặt, từ tập dữ liệu CelebA.
  - Lớp cuối cùng (Fully Connected Layer) ban đầu được thay thế bằng một lớp trung gian (*embedding layer*) có đầu ra là vector biểu diễn (128-chiều), phù hợp với yêu cầu của *Triplet Loss*.

### MobileNet-V2

MobileNet-V2 là phiên bản cải tiến của MobileNet, được thiết kế với mục tiêu giảm độ phức tạp tính toán thông qua việc sử dụng *Depthwise Separable Convolution* và kiến trúc *Inverted Residual Block*. So với phiên bản trước, MobileNet-V2 cải thiện khả năng học các đặc trưng hiệu quả hơn, đặc biệt là trong các hệ thống có tài nguyên hạn chế. Tham khảo thêm ở Hình 2.10.

Trong bài toán *Face Image Retrieval* dựa trên *Triplet Loss*, phương pháp *fine-tuning* với MobileNet-V2 được thực hiện tương tự như với ResNet-50:

- **Đóng băng các tầng đầu:** Các tầng đầu tiên của MobileNet-V2 (bao gồm phần lớn các khối *Depthwise Separable Convolution*) được giữ nguyên trọng số (*frozen*), tận dụng các đặc trưng phổ quát đã học từ ImageNet.



Hình 2.9: Quy trình fine-tuning ResNet-50 cho bài toán Face Image Retrieval.

- **Thay thế lớp cuối:** Lớp *classifier* ban đầu được thay thế bằng một *embedding layer* với đầu ra là vector biểu diễn 128-chiều tương tự như ở Phần 2.2.4, phục vụ cho mục tiêu tối ưu hóa không gian biểu diễn của *Triplet Loss*.
- **Huấn luyện các tầng cao:** Mô hình mở khóa (*unfreeze*) 10 lớp cuối cùng, bao gồm các *bottleneck blocks* và lớp kết nối cuối. Điều này giúp học các đặc trưng chuyên biệt hơn, phù hợp với dữ liệu CelebA.

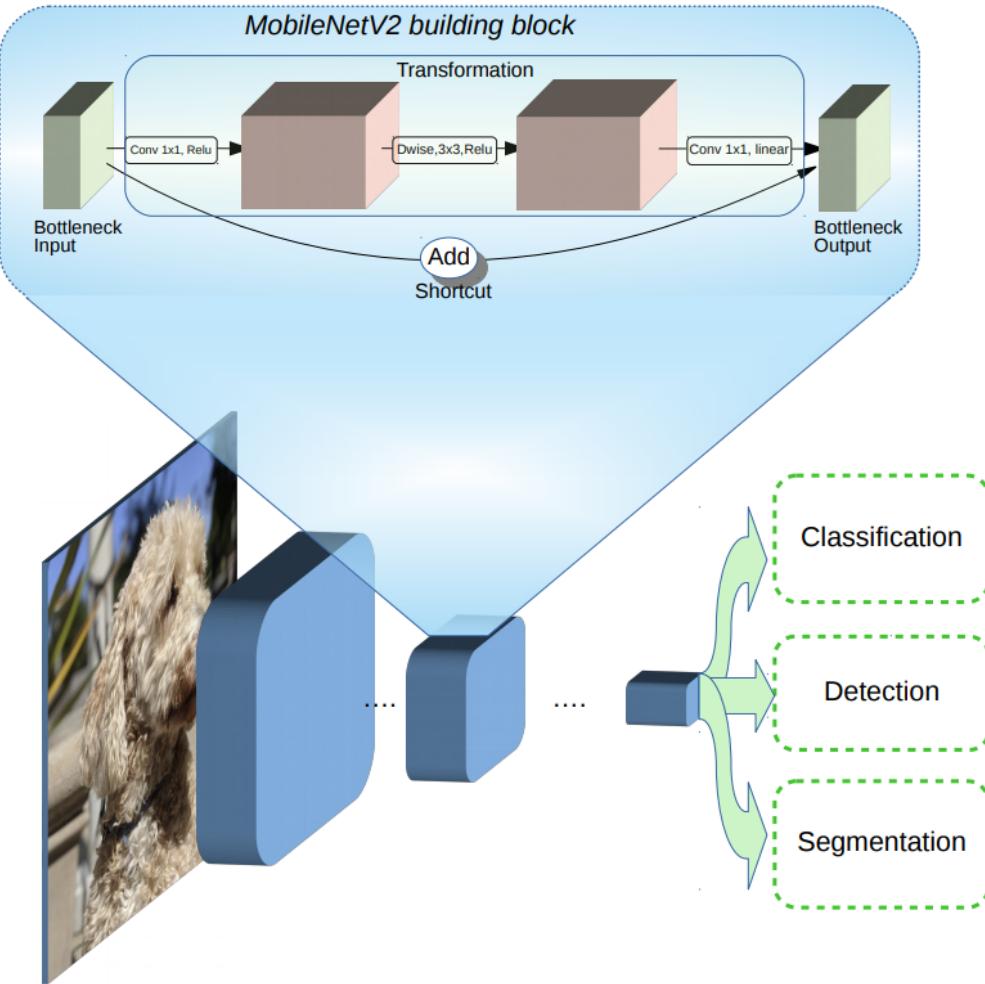
## 2.2.5 Phương pháp truy vấn: KD-Tree [DVL21]

**KD-Tree (K-dimensional Tree)** là một cấu trúc dữ liệu phân hoạch nhị phân trong không gian nhiều chiều, được thiết kế để tổ chức và truy vấn hiệu quả các điểm trong tập dữ liệu lớn.

### Công thức và Chi tiết

1. **Xây dựng KD-Tree:** Cho tập dữ liệu  $D = \{f_1, f_2, \dots, f_n\}$ , trong đó mỗi phần tử  $f_i \in \mathbb{R}^k$  là một vector embedding. KD-Tree là cây nhị phân có trọng số, trong đó:

- Các nút trong (*iNode*) chứa vector trọng số  $W_i = [w_{i1}, w_{i2}, \dots, w_{ik}]$  và ngưỡng  $t$ , dùng để phân hoạch không gian.
- Các nút lá (*LNode*) lưu tập các vector  $f_i$  tương ứng với một cụm dữ liệu.



Hình 2.10: *Tổng quan về Kiến trúc MobileNet-V2. Các khối màu xanh biểu diễn các khối tích chập tổng hợp như được hiển thị ở trên. Trong đó, MobileNetV2 building block chính là việc xử lý Inverted Residual Block kết hợp cùng với Depth-wise Separable Convolution.* <https://s.net.vn/syaH>.

Quá trình xây dựng KD-Tree được thực hiện như sau:

- (a) **Phân hoạch không gian:** Tại mỗi  $iNode$ , vector trọng số  $W_i$  được huấn luyện để tối ưu việc phân hoạch dữ liệu thành hai nhánh:

$$S = \text{Sign}(\text{Sigmoid}(W_i \cdot f_i - t) - t).$$

- Nếu  $S \geq 0$ , vector  $f_i$  thuộc nhánh phải.
- Nếu  $S < 0$ , vector  $f_i$  thuộc nhánh trái.

Trong số  $W_i$  tại mỗi  $iNode_i$  được khởi tạo ngẫu nhiên, sau đó được huấn luyện theo phương pháp giảm sai số trung bình tỷ lệ phân lớp tại các nút lá.

- (b) **Phân chia dữ liệu tại mỗi nút:** Dữ liệu tại  $iNode$  được chia thành hai phần dựa trên kết quả phân hoạch:

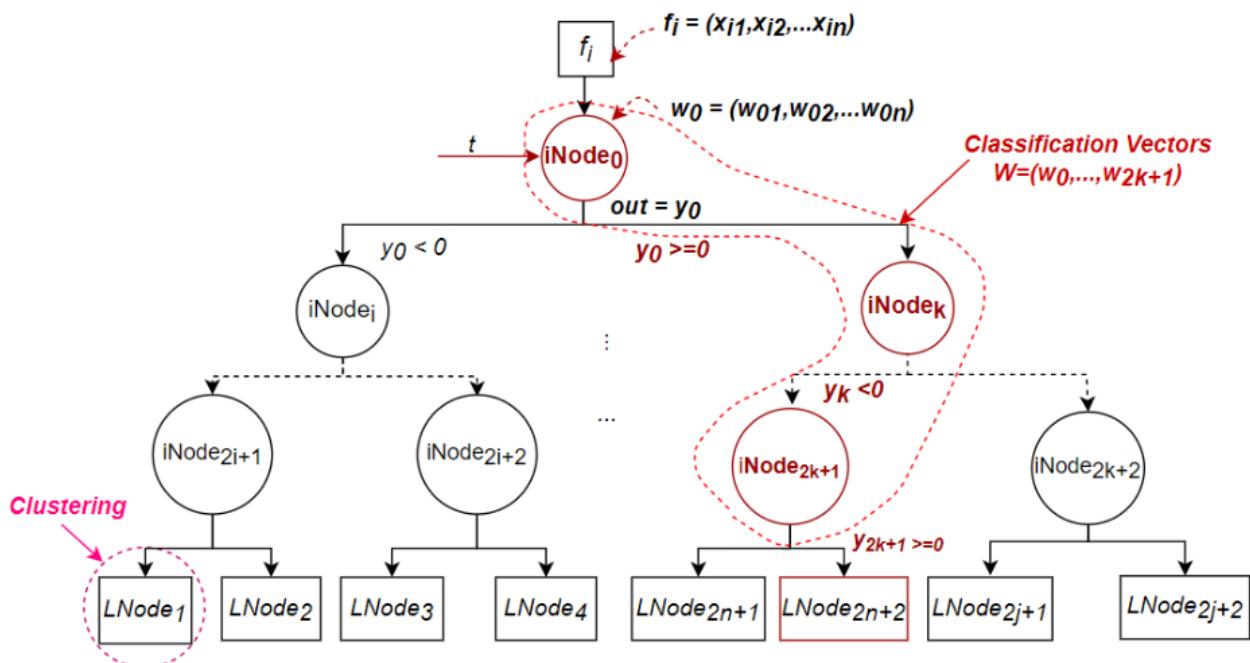
$$D_{\text{left}} = \{f \in D : S < 0\}, \quad D_{\text{right}} = \{f \in D : S \geq 0\}.$$

(c) **Độ quy phân hoạch:** Tiếp tục xây dựng cây cho  $D_{\text{left}}$  và  $D_{\text{right}}$  cho đến khi đạt điều kiện dừng:

- Sự sai khác giữa các embedding vector  $f_i$  trong  $LNode$  là chấp nhận được.
- Kích thước dữ liệu trong  $LNode$  nhỏ hơn một ngưỡng nhất định (ví dụ:  $|D| \leq m$ ).
- Độ sâu của cây đạt giới hạn tối đa  $h$ .

Tại thời điểm này, các nút sẽ trở thành  $LNode$ , lưu tập dữ liệu tương ứng.

Tham khảo cấu trúc KD-Tree ở Hình 2.11.



Hình 2.11: Cấu trúc KD-Tree tổng quát.

2. **Thuật toán truy vấn KD-Tree (tìm  $k$ -ảnh gần nhất):** Với một vector truy vấn  $q \in \mathbb{R}^k$ , thuật toán tìm kiếm  $k$ -vector gần nhất trên KD-Tree được thực hiện như sau:

(a) **Tại mỗi nút phân loại ( $iNode$ ):** Sử dụng vector trọng số  $W_i$  và ngưỡng  $t$  tại nút  $iNode_i$  để tính giá trị phân hoạch:

$$S = \text{Sign}(\text{Sigmoid}(W_i \cdot q - t) - t).$$

- Nếu  $S \geq 0$ , đi vào nhánh phải.
- Nếu  $S < 0$ , đi vào nhánh trái.

- (b) **Khi đến nút lá ( $LNode$ ):** Tập dữ liệu tại nút lá được so sánh với  $q$  để tính khoảng cách:

$$d(q, f_i) = \|q - f_i\|_2^2,$$

với  $f_i \in LNode$ . Cập nhật danh sách  $k$ -vector gần nhất ( $k$ -nearest neighbors):

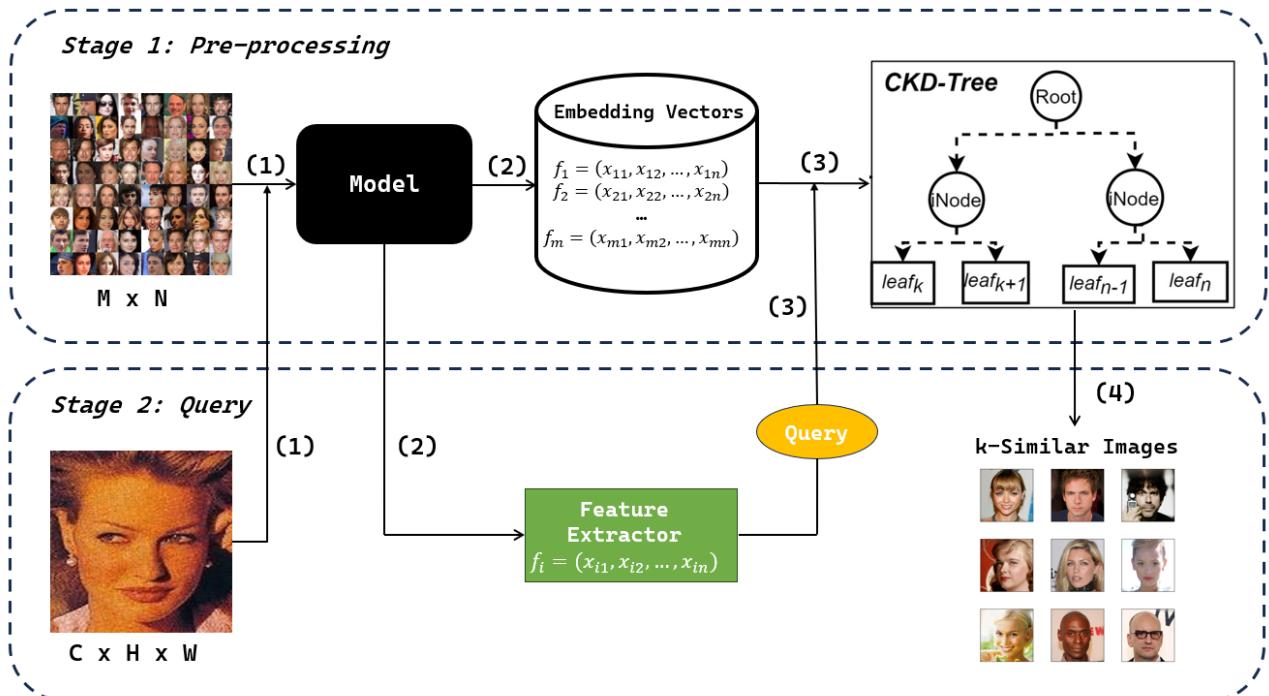
- Nếu danh sách  $k$ -vector chưa đầy, thêm vector  $f_i$  vào danh sách. Hoặc nếu không tồn tại  $f_i$  nào khác trong  $LNode_k$ , thuật toán sẽ đi sang  $LNode_l$  khác sao cho đây là nút lá gần nhất với nút hiện tại.
- Nếu danh sách đã đầy, thay thế vector có khoảng cách lớn nhất trong danh sách nếu khoảng cách của  $f_i$  nhỏ hơn.

- (c) **Kết quả:** Sau khi duyệt hết các nhánh có thể, danh sách  $k$ -vector gần nhất chứa các vector  $f_i \in D$  gần nhất với  $q$ .

3. **Kết quả:** Sau khi duyệt xong cây, tập kết quả *Result* chứa các vector  $f_i$  trong không gian biểu diễn gần nhất với  $q$ :

$$\text{Result} = \bigcup_{LNode} \{f_i \in LNode : f_i \text{ thỏa mãn tiêu chí gần nhất}\}.$$

Xem quy trình tổng quát của việc truy vấn hình ảnh tại Hình 2.12.



Hình 2.12: Quy trình truy vấn ảnh tổng quát.

## Ưu điểm của KD-Tree trong bài toán

- **Hiệu suất cao trong không gian thấp đến trung bình chiều:** Trong không gian embedding ( $d = 128$ ), KD-Tree cung cấp thời gian truy vấn  $O(\log N)$ , đảm bảo khả năng mở rộng tốt với dữ liệu lớn như CelebA.
- **Phân hoạch thông minh:** KD-Tree chia không gian biểu diễn thành các vùng nhỏ hơn, giúp tìm kiếm nhanh các hình ảnh gần nhất trong không gian embedding, hỗ trợ tốt cho bài toán *Face Image Retrieval*.
- **Tối ưu hóa cho khoảng cách Euclid:** Khoảng cách  $L_2$  trong *Triplet Loss* được hỗ trợ tốt bởi KD-Tree, đảm bảo kết quả chính xác khi tìm kiếm các vector tương tự.
- **Tăng tốc độ truy vấn và giảm tài nguyên:** KD-Tree giảm số lượng phép tính cần thiết khi so sánh vector embedding, phù hợp với các bài toán thời gian thực hoặc triển khai trên hệ thống hạn chế.

## Mục đích sử dụng KD-Tree trong bài toán

- **Tăng tốc độ xử lý:** KD-Tree giảm thời gian truy vấn  $k$ -hàng xóm gần nhất từ  $O(N)$  xuống  $O(\log N)$ , cải thiện đáng kể hiệu quả xử lý.
- **Hỗ trợ đánh giá hiệu suất:** Các chỉ số như *Top-k Accuracy* và *Mean Distance* dựa trên khoảng cách từ KD-Tree, giúp đánh giá chất lượng không gian biểu diễn của mô hình.
- **Tích hợp linh hoạt:** KD-Tree dễ dàng tích hợp với các vector embedding sinh ra từ mô hình ResNet hoặc MobileNet, đảm bảo tính đồng nhất trong pipeline xử lý.

# Chương 3

## Đánh giá mô hình

### 3.1 Phương pháp đánh giá

Mục tiêu chính của bài toán là đánh giá khả năng truy xuất ảnh của mô hình dựa trên các vector biểu diễn (embeddings). Do sự đa dạng trong tập ảnh của cùng một danh tính (ID) — chẳng hạn một người có thể xuất hiện trong các bức ảnh với phụ kiện, biểu cảm, hoặc góc mặt khác nhau — việc sử dụng ID để đánh giá truy xuất gấp khó khăn, thậm chí cho kết quả là 0%. Để khắc phục, tôi đã chuyển hướng phương pháp đánh giá sang sử dụng các thuộc tính (attributes) của ảnh, dựa trên danh sách 40 thuộc tính được cung cấp (xem thêm ở <https://s.net.vn/LAd7>). Việc này cho phép định lượng mức độ tương đồng giữa ảnh truy vấn và các ảnh trong tập gallery dựa trên các đặc điểm chung. Phương pháp được chia thành các bước sau:

1. **Tính vector biểu diễn (embeddings):** Mô hình trích xuất vector embedding từ cả tập query và tập gallery. Mỗi embedding là một vector trong không gian  $\mathbb{R}^k$  (trong bài toán này, tôi sử dụng  $k = 128$ ), biểu diễn đặc trưng của ảnh.
2. **Xây dựng KDTree:** Các vector embedding của tập gallery được dùng để xây dựng KDTree, cho phép truy vấn nhanh chóng và hiệu quả (xem thêm ở Hình 2.11).
3. **Truy vấn và so sánh:** Với mỗi vector  $q$  từ tập query, sử dụng KDTree để tìm  $k$  vector gần nhất trong tập gallery dựa trên khoảng cách Euclidean. Sau đó, lấy danh sách  $k$ -ảnh gần nhất (top- $k$ ) cho mỗi ảnh query (xem thêm ở Hình 2.12).
4. **So sánh attributes:** Lấy vector attributes tương ứng của ảnh query và các ảnh trong danh sách top- $k$  và tính toán các chỉ số:
  - **Accuracy:** Tỷ lệ thuộc tính trùng khớp giữa ảnh query và ảnh gallery:

$$\text{Accuracy} = \frac{\text{Number of matching attributes}}{\text{Total number of attributes}}.$$

- **Distance:** Trung bình chênh lệch tuyệt đối giữa các giá trị thuộc tính:

$$\text{Distance} = \frac{\sum_j |\text{attr}_{j,\text{query}} - \text{attr}_{j,\text{gallery}}|}{\text{Total number of attributes}}.$$

5. **Tổng hợp kết quả:** Tính trung bình Accuracy và Distance trên toàn bộ tập query:

$$\text{Top-}k \text{ Attribute Accuracy} = \frac{1}{k} \sum_{i=1}^k \text{Accuracy}_i,$$

$$\text{Mean Attribute Distance} = \frac{1}{k} \sum_{i=1}^k \text{Distance}_i.$$

Các chỉ số này thể hiện mức độ tương đồng giữa ảnh query và các ảnh truy xuất được dựa trên attributes.

## Ưu điểm của phương pháp đánh giá

- **Giải quyết sự đa dạng của ID:** Dánh giá dựa trên attributes thay vì ID giúp mô hình thể hiện hiệu quả trong việc truy xuất ảnh tương tự mà không bị ảnh hưởng bởi sự đa dạng kiểu dáng trong cùng ID.
- **Hiệu quả tính toán:** KDTree tối ưu hóa việc truy vấn top- $k$ , giảm đáng kể chi phí tính toán so với việc tính khoảng cách với toàn bộ tập gallery.

## 3.2 Đánh giá quá trình huấn luyện và kết quả

### 3.2.1 Trước khi thực hiện Fine-Tuning

$k$	ResNet-50			MobileNet-V2		
	Độ chính xác (%)	Khoảng cách TB	Thời gian (s)	Độ chính xác (%)	Khoảng cách TB	Thời gian (s)
$k = 10$	80.19	0.3962	56	79.62	0.4076	54
$k = 20$	79.88	0.4023	56	79.28	0.4143	58
$k = 50$	79.52	0.4097	58	78.96	0.4207	55
$k = 100$	79.22	0.4156	64	78.68	0.4265	63
$k = 1000$	78.07	0.4386	235	77.46	0.4508	233
Trung bình	79.38	0.4125	94s <sup>1</sup> và 93s <sup>2</sup>	78.8	0.424	93s <sup>1</sup> và 49s <sup>2</sup>

Bảng 3.1: So sánh hiệu suất truy xuất giữa ResNet-50 và MobileNet-V2 cho các giá trị  $k$  khác nhau.

<sup>1</sup>Thời gian truy vấn trung bình của tất cả các giá trị  $k$ .

<sup>2</sup>Thời gian tính toán embedding cho toàn bộ tập dữ liệu.

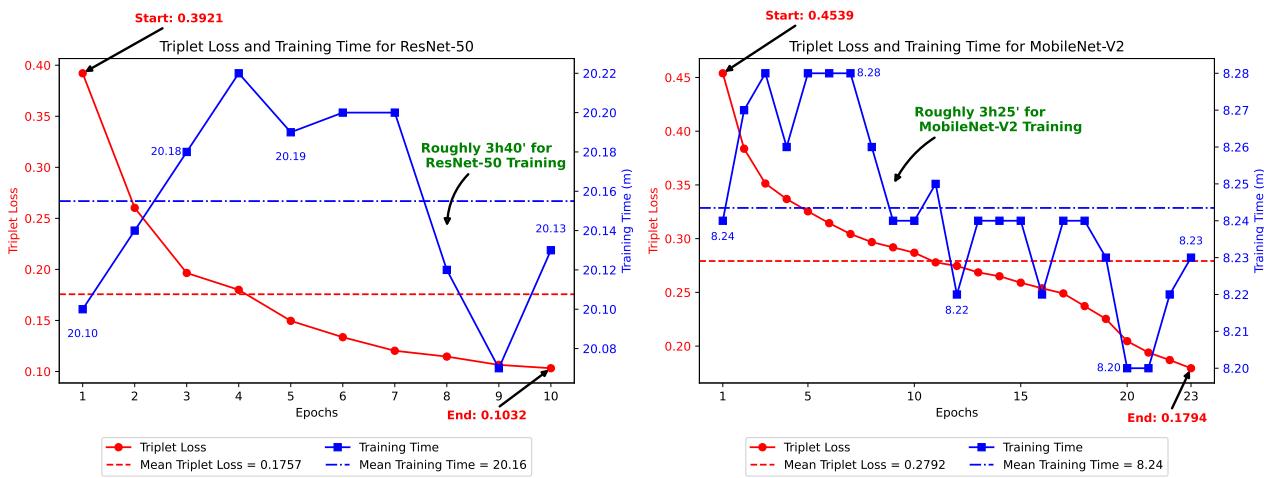
Trước khi tiến hành Fine-Tuning, chúng tôi thực hiện so sánh chi tiết giữa hai mô hình ResNet-50 và MobileNet-V2 dựa trên ba tiêu chí chính: **Độ chính xác**, **Khoảng cách Trung bình (TB)**, và **Thời gian xử lý**. Kết quả được trình bày trong [Bảng 3.1](#).

- **Độ chính xác:** ResNet-50 đạt kết quả cao hơn MobileNet-V2 trên toàn bộ các giá trị  $k$ , với độ chính xác trung bình lần lượt là **79.38%** và **78.8%**. Điều này cho thấy ResNet-50 có khả năng học không gian biểu diễn tốt hơn và phân cụm chính xác hơn trong bài toán truy xuất ảnh.
- **Khoảng cách Trung bình:** ResNet-50 tiếp tục vượt trội với giá trị Khoảng cách TB trung bình thấp hơn (**0.4125**) so với MobileNet-V2 (**0.424**). Khoảng cách TB thấp hơn cho thấy các vector biểu diễn từ ResNet-50 được ánh xạ gần nhau hơn trong không gian embedding, giúp tăng cường hiệu quả truy xuất.
- **Thời gian Xử lý:** MobileNet-V2 tỏ ra vượt trội về tốc độ, với thời gian xử lý trung bình **93 giây** cho toàn bộ các giá trị  $k$ , thấp hơn đáng kể so với ResNet-50 (**94 giây**). Đặc biệt, khi tính toán embedding cho toàn bộ tập dữ liệu, MobileNet-V2 chỉ mất **49 giây**, trong khi ResNet-50 cần tới **93 giây**. Điều này khẳng định MobileNet-V2 phù hợp hơn với các ứng dụng yêu cầu tốc độ cao.
- **Hiệu quả Tổng thể:** Dựa trên các tiêu chí, ResNet-50 chứng minh được hiệu quả vượt trội trong việc tối ưu hóa không gian biểu diễn với độ chính xác và khoảng cách TB tốt hơn. Tuy nhiên, chi phí cho sự vượt trội này là thời gian xử lý dài hơn. MobileNet-V2, dù có hiệu suất biểu diễn thấp hơn, lại là lựa chọn lý tưởng cho các ứng dụng yêu cầu tối ưu chi phí thời gian và tài nguyên tính toán.

**Kết luận**, việc lựa chọn giữa ResNet-50 và MobileNet-V2 phụ thuộc vào yêu cầu cụ thể của bài toán. Nếu độ chính xác và khả năng học không gian biểu diễn là yếu tố quan trọng, ResNet-50 là sự lựa chọn tối ưu. Trong khi đó, MobileNet-V2 lại phù hợp hơn cho các hệ thống cần xử lý nhanh và hiệu quả.

### 3.2.2 Quá trình Fine-Tuning

Trong phần này, chúng tôi thực hiện so sánh hiệu suất giữa hai mô hình nổi bật là ResNet-50 và MobileNet-V2 dựa trên hai tiêu chí chính: *Triplet Loss* và *Thời gian Huấn luyện*. Đây là hai chỉ số quan trọng trong việc đánh giá hiệu quả của mô hình học không gian biểu diễn (*embedding space*) trong bài toán nhận diện khuôn mặt.



Hình 3.1: Biểu đồ Triplet Loss và thời gian huấn luyện cho từng mô hình.

1. **Triplet Loss:** ResNet-50 cho thấy sự vượt trội rõ rệt so với MobileNet-V2 khi xét về tiêu chí này.

Cụ thể, ResNet-50 bắt đầu với giá trị Triplet Loss là **0.3921** và giảm xuống chỉ còn **0.1032** sau 10 epoch. Trong suốt quá trình huấn luyện, giá trị trung bình của Triplet Loss đạt **0.1757**, chứng tỏ mô hình này có khả năng tối ưu hóa không gian biểu diễn một cách hiệu quả. Đường giảm Triplet Loss của ResNet-50 tương đối mượt mà, phản ánh sự ổn định trong quá trình học.

Ngược lại, MobileNet-V2 khởi đầu với Triplet Loss là **0.4539** và giảm xuống **0.1794** sau 23 epoch. Mặc dù có sự cải thiện, nhưng giá trị Triplet Loss trung bình là **0.2792**, cao hơn đáng kể so với ResNet-50. Điều này cho thấy MobileNet-V2 gặp khó khăn hơn trong việc học không gian biểu diễn tối ưu.

2. **Thời gian Huấn luyện:** Thời gian huấn luyện là yếu tố quan trọng khi cân nhắc chi phí tính toán và hiệu quả sử dụng tài nguyên. ResNet-50 và MobileNet-V2 cho thấy sự khác biệt rõ rệt trong tiêu chí này.

ResNet-50 yêu cầu trung bình **20.16 phút** cho mỗi epoch, dẫn đến tổng thời gian huấn luyện khoảng **3 giờ 40 phút** cho 10 epoch. Đây là một mô hình sâu với nhiều tham số, nên chi phí tính toán cao là điều dễ hiểu.

Trong khi đó, MobileNet-V2 nổi bật với thời gian huấn luyện ngắn hơn nhiều. Với trung bình **8.24 phút** mỗi epoch, tổng thời gian huấn luyện cho 23 epoch chỉ khoảng **3 giờ 25 phút**. Điều này cho thấy MobileNet-V2 phù hợp hơn cho các ứng dụng cần tối ưu hóa chi phí tính toán mà vẫn đạt hiệu quả ở mức tương đối.

3. **Hiệu quả Tổng thể:** Kết quả cho thấy, ResNet-50 vượt trội hơn MobileNet-

V2 trong việc tối ưu Triplet Loss, đạt được không gian biểu diễn hiệu quả hơn. Tuy nhiên, chi phí cho sự vượt trội này là thời gian huấn luyện dài hơn đáng kể. Ngược lại, MobileNet-V2 cung cấp một lựa chọn kinh tế hơn về thời gian, nhưng phải đánh đổi bằng chất lượng không gian biểu diễn thấp hơn.

4. **Lựa chọn ngưỡng:** Trong quá trình huấn luyện, việc lựa chọn số epoch tối ưu để dừng lại là một yếu tố quan trọng, nhằm cân bằng giữa hiệu quả mô hình và chi phí tính toán. Trong phần này, chúng tôi sử dụng hai tiêu chí chính để xác định ngưỡng phù hợp: *Triplet Loss* thấp nhất và *Thời gian Huấn luyện* tối ưu.

- Đối với ResNet-50, giá trị Triplet Loss thấp nhất đạt được tại epoch từ 8 đến 10 với giá trị khoảng **0.1032**. Điều này cho thấy đây là ngưỡng lý tưởng để dừng huấn luyện nếu mục tiêu chính là tối ưu hóa không gian biểu diễn. Với MobileNet-V2, giá trị Triplet Loss thấp nhất đạt được ở epoch từ 20 đến 23, với giá trị khoảng từ **0.22** đến **0.1794**. Mặc dù giá trị này cao hơn so với ResNet-50, nó cho thấy rằng mô hình cần nhiều epoch hơn để đạt hiệu quả tối đa.
- Đối với các bài toán yêu cầu thời gian huấn luyện ngắn, MobileNet-V2 là lựa chọn phù hợp hơn. Mỗi epoch của MobileNet-V2 trung bình mất **8.24 phút** và đạt ngưỡng tối ưu ở epoch 20 và 21, trong khi ResNet-50 cần tới **20.16 phút** và đạt ngưỡng thấp nhất ở epoch thứ 9. Do đó, nếu tối ưu hóa thời gian là ưu tiên hàng đầu, MobileNet-V2 có thể được dừng lại sớm hơn với mức độ chấp nhận được của Triplet Loss.

Để xác định ngưỡng tốt nhất, chúng tôi thực hiện cân nhắc giữa hai tiêu chí trên. ResNet-50 là lựa chọn ưu tiên nếu chất lượng không gian biểu diễn được đặt lên hàng đầu, với ngưỡng dừng tối ưu là epoch thứ 9. Ngược lại, MobileNet-V2 phù hợp hơn nếu yêu cầu tính toán nhanh, với ngưỡng dừng tại epoch thứ 20 hoặc 21.

5. **Kết luận:** Lựa chọn giữa ResNet-50 và MobileNet-V2 phụ thuộc vào ưu tiên của bài toán. Nếu chất lượng không gian biểu diễn là yếu tố quyết định, ResNet-50 là lựa chọn tối ưu. Trong khi đó, MobileNet-V2 phù hợp hơn cho các ứng dụng yêu cầu tốc độ và hiệu quả tính toán. Hình 3.1 minh họa chi tiết sự khác biệt về Triplet Loss và Thời gian Huấn luyện giữa hai mô hình này.

### 3.2.3 Sau khi thực hiện Fine-Tuning

Sau khi thực hiện Fine-Tuning, chúng tôi cũng tiến hành so sánh hiệu suất của ResNet-50 và MobileNet-V2 trên ba tiêu chí chính: **Độ chính xác**, **Khoảng**

$k$	ResNet-50			MobileNet-V2		
	Độ chính xác (%)	Khoảng cách TB	Thời gian (s)	Độ chính xác (%)	Khoảng cách TB	Thời gian (s)
$k = 10$	82.33	0.3534	41	81.70	0.3659	39
$k = 20$	82.11	0.3577	44	81.50	0.3700	41
$k = 50$	81.82	0.3635	43	81.25	0.3750	41
$k = 100$	81.57	0.3685	53	81.05	0.3790	51
$k = 1000$	80.47	0.3906	217	80.11	0.3979	215
Trung bình	<b>81.66</b>	<b>0.3667</b>	80s <sup>1</sup> và 102s <sup>2</sup>	<b>81.12</b>	<b>0.3776</b>	77s <sup>1</sup> và 49s <sup>2</sup>

Bảng 3.2: So sánh hiệu suất truy xuất giữa ResNet-50 và MobileNet-V2 cho các giá trị  $k$  khác nhau sau khi thực hiện Fine-tuning.

cách **Trung bình (TB)**, và **Thời gian xử lý**. Kết quả chi tiết được trình bày trong Bảng 3.2.

- **Độ chính xác:** Sau Fine-Tuning, cả hai mô hình đều cải thiện đáng kể độ chính xác. ResNet-50 đạt độ chính xác trung bình **81.66%**, cao hơn MobileNet-V2 (**81.12%**). Điều này khẳng định khả năng học không gian biểu diễn sâu sắc hơn của ResNet-50 sau khi tinh chỉnh.
- **Khoảng cách Trung bình:** Giá trị Khoảng cách TB của ResNet-50 sau Fine-Tuning giảm xuống còn **0.3667**, trong khi MobileNet-V2 đạt **0.3776**. Mặc dù sự chênh lệch không lớn, nhưng ResNet-50 vẫn cho thấy ưu thế trong việc nén các vector biểu diễn, làm tăng khả năng phân biệt các đối tượng trong không gian embedding.
- **Thời gian Xử lý:** Thời gian xử lý trung bình giảm đáng kể ở cả hai mô hình nhờ quá trình Fine-Tuning. MobileNet-V2 tiếp tục chiếm ưu thế với thời gian xử lý trung bình chỉ **77 giây**, thấp hơn ResNet-50 (**80 giây**). Khi tính toán embedding cho toàn bộ tập dữ liệu, MobileNet-V2 chỉ mất **49 giây**, so với **102 giây** của ResNet-50. Điều này củng cố vị thế của MobileNet-V2 trong các ứng dụng yêu cầu tốc độ cao.
- **Hiệu quả Tổng thể:** Sau Fine-Tuning, ResNet-50 giữ vững vị trí dẫn đầu về độ chính xác và khả năng học không gian biểu diễn, nhưng vẫn phải đánh đổi bằng thời gian xử lý dài hơn. MobileNet-V2, mặc dù có hiệu suất thấp hơn một chút, nhưng vẫn đáp ứng tốt nhu cầu về tốc độ và chi phí tính toán.

**Kết luận**, Fine-Tuning đã cải thiện hiệu suất của cả hai mô hình đáng kể, giúp chúng đáp ứng tốt hơn các yêu cầu của bài toán. ResNet-50 phù hợp cho

<sup>1</sup>Thời gian truy vấn trung bình của tất cả các giá trị  $k$ .

<sup>2</sup>Thời gian tính toán embedding cho toàn bộ tập dữ liệu.

các ứng dụng đòi hỏi độ chính xác cao và khả năng phân biệt mạnh mẽ. Trong khi đó, MobileNet-V2 vẫn là lựa chọn lý tưởng cho các hệ thống yêu cầu xử lý nhanh và tiết kiệm tài nguyên.

### 3.3 Áp dụng vào truy vấn ảnh

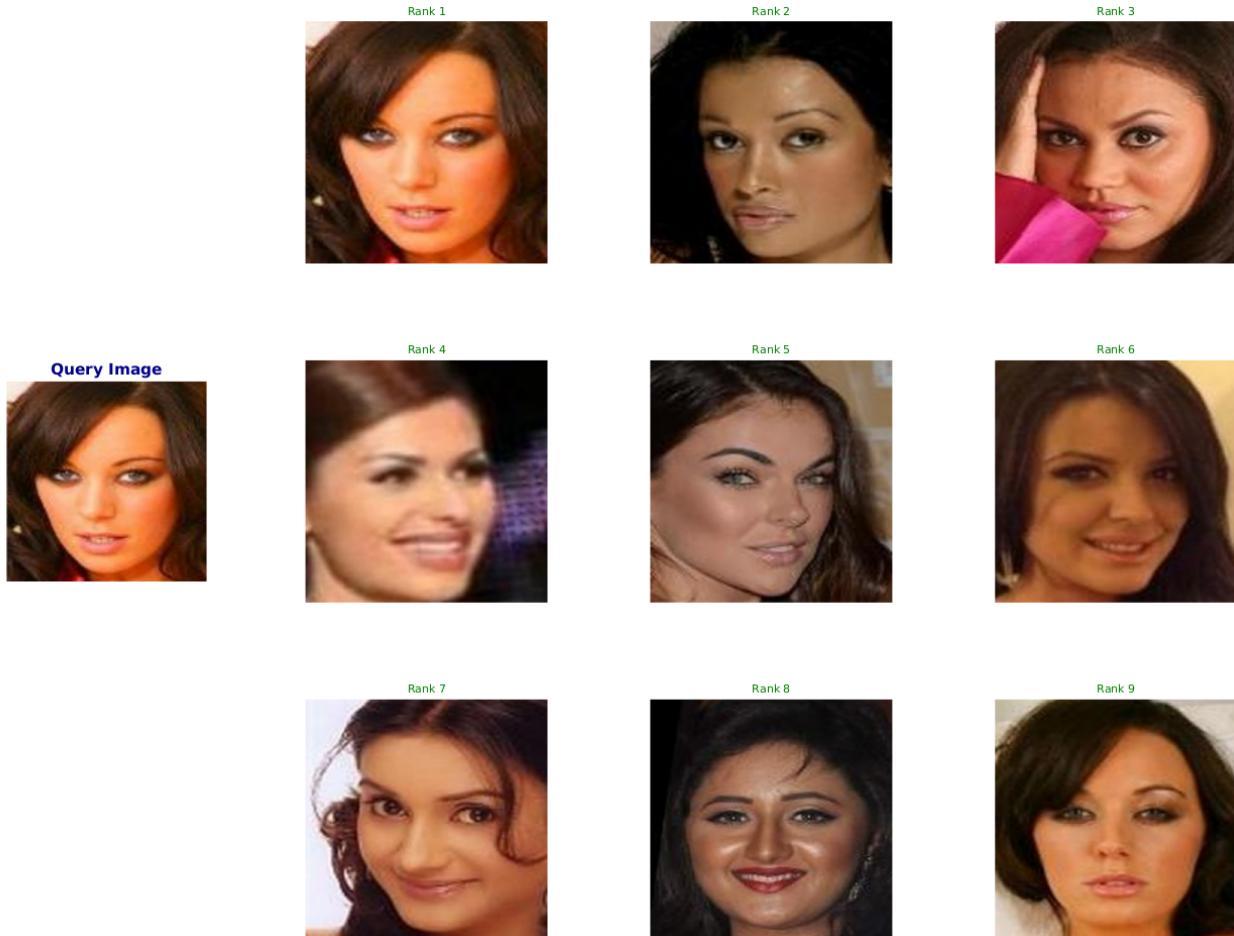
#### Mô hình ResNet-50

Hình 3.2 minh họa kết quả truy vấn hình ảnh dựa trên mô hình Fine-Tuned ResNet-50. Trong hình, ảnh truy vấn (*Query Image*) được đặt ở góc dưới bên trái, với các hình ảnh được sắp xếp theo thứ tự từ Rank 1 đến Rank 9. Dưới đây là nhận xét chi tiết về kết quả:

- **Độ chính xác của truy vấn:**

- **Hình ảnh tại Rank 1:** Kết quả truy vấn ở Rank 1 có độ tương đồng rất cao với ảnh truy vấn và có thể nói đây chính là ảnh mà chúng ta truy vấn trong tập dữ liệu. Đây là minh chứng rõ ràng cho hiệu quả của Fine-Tuned ResNet-50 trong việc học không gian biểu diễn (*embedding space*), giúp mô hình truy xuất chính xác nhất ảnh có cùng danh tính.
- **Các hình ảnh từ Rank 2 đến Rank 6:** Mặc dù có sự khác biệt nhỏ về các chi tiết như ánh sáng, góc chụp hoặc biểu cảm khuôn mặt, nhưng các kết quả này vẫn duy trì sự tương đồng cao với ảnh truy vấn. Điều này phản ánh khả năng ánh xạ các đặc trưng khuôn mặt mạnh mẽ của ResNet-50.
- **Các hình ảnh từ Rank 7 đến Rank 9:** Kết quả ở các hạng thấp này có độ chính xác giảm đáng kể. Một số ảnh trong nhóm này thể hiện sự khác biệt rõ rệt về cấu trúc khuôn mặt hoặc màu da. Điều này cho thấy mô hình gặp khó khăn trong việc phân biệt các khuôn mặt thuộc danh tính khác khi chúng có các đặc điểm tương tự.
- **Tính nhất quán trong không gian biểu diễn:** Kết quả truy vấn cho thấy mô hình đã nhóm các khuôn mặt dựa trên sự tương đồng về các đặc điểm tổng quát, như mắt, mũi và cấu trúc gương mặt. Tuy nhiên, ở các hạng thấp hơn (Rank 7–9), không gian biểu diễn chưa hoàn toàn tối ưu, dẫn đến sự lẫn lộn giữa các danh tính khác nhau.
- **Nhận xét tổng quan:** Kết quả truy vấn hình ảnh từ mô hình Fine-Tuned ResNet-50 cho thấy hiệu quả vượt trội trong việc ánh xạ các khuôn mặt có cùng danh tính vào không gian gần nhau. Tuy nhiên, để cải thiện độ chính xác ở các hạng thấp, ta cần tinh chỉnh thêm các lớp cuối của mô hình để

giảm hiện tượng nhiễu trong không gian biểu diễn. Ngoài ra, sử dụng thêm dữ liệu hoặc các chiến lược học tập mạnh mẽ hơn để cải thiện khả năng phân biệt ở các danh tính khác nhau. Nhìn chung, mô hình ResNet-50 sau Fine-Tuning đã đáp ứng tốt bài toán truy vấn hình ảnh, đặc biệt ở các hạng cao, với sự ưu việt trong việc phân biệt các khuôn mặt tương tự.



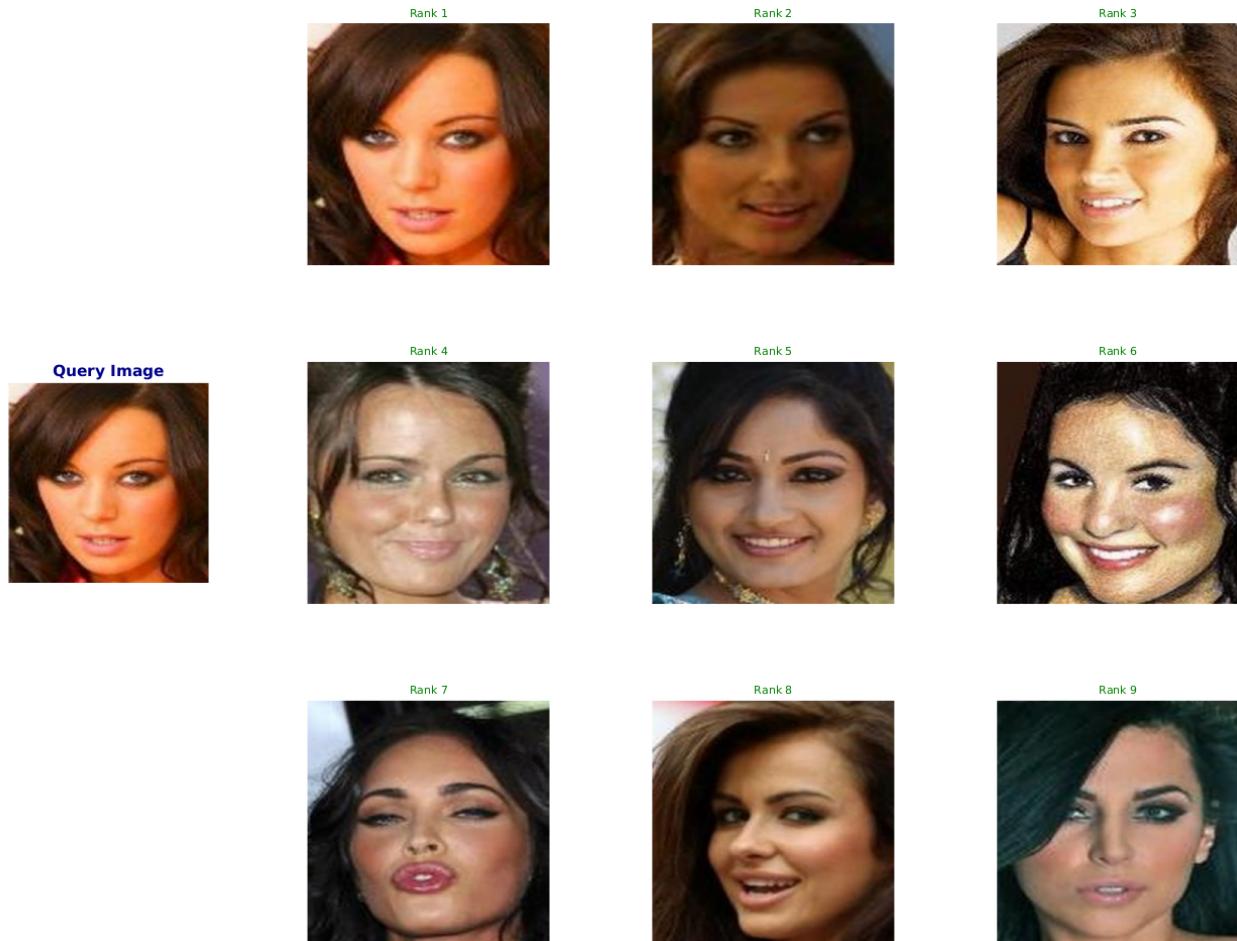
Hình 3.2: Kết quả truy vấn hình ảnh dựa trên mô hình Fine-Tuned Resnet-50.

## Mô hình MobileNet-V2

Hình 3.3 minh họa kết quả truy vấn hình ảnh dựa trên mô hình Fine-Tuned MobileNet-V2. Trong hình, ảnh truy vấn (*Query Image*) được đặt ở góc dưới bên trái, với các hình ảnh được sắp xếp theo thứ tự từ Rank 1 đến Rank 9. Dưới đây là nhận xét chi tiết về kết quả:

- **Độ chính xác của truy vấn:**

- **Hình ảnh tại Rank 1:** Kết quả truy vấn ở Rank 1 có độ tương đồng rất cao với ảnh truy vấn và có thể nói đây chính là ảnh mà chúng ta truy vấn trong tập dữ liệu. Đây là minh chứng rõ ràng cho hiệu quả



Hình 3.3: Kết quả truy vấn hình ảnh dựa trên mô hình Fine-Tuned MobileNet-V2.

của Fine-Tuned MobileNet-V2 trong việc học không gian biểu diễn (*embedding space*), giúp mô hình truy xuất chính xác nhất ảnh có cùng danh tính.

- **Các hình ảnh từ Rank 2 đến Rank 6:** Các kết quả này vẫn duy trì sự tương đồng tương đối với ảnh truy vấn. Tuy nhiên, sự khác biệt về ánh sáng, góc chụp và biểu cảm bắt đầu rõ rệt hơn. Điều này cho thấy khả năng học của MobileNet-V2 vẫn hiệu quả nhưng không mạnh mẽ bằng ResNet-50.
- **Các hình ảnh từ Rank 7 đến Rank 9:** Các kết quả ở thứ hạng thấp này có sự khác biệt đáng kể về cấu trúc khuôn mặt, màu da, và đặc điểm tổng thể. Điều này phản ánh giới hạn của MobileNet-V2 trong việc phân biệt các danh tính khác nhau khi chúng có các đặc điểm tương tự.
- **Tính nhất quán trong không gian biểu diễn:** MobileNet-V2 đã thể hiện khả năng nhóm các khuôn mặt tương tự trong các hạng cao (Rank 1 đến Rank 6), nhờ học được các đặc trưng tổng quát như mắt, mũi, và

cấu trúc khuôn mặt. Tuy nhiên, ở các hạng thấp hơn (Rank 7 đến Rank 9), không gian biểu diễn chưa hoàn toàn tối ưu, dẫn đến sự nhầm lẫn giữa các danh tính khác nhau.

- **Nhận xét tổng quan:** Kết quả truy vấn hình ảnh từ mô hình Fine-Tuned MobileNet-V2 cho thấy khả năng xử lý tốt ở các thứ hạng cao, đặc biệt là Rank 1 đến Rank 3. Tuy nhiên, độ chính xác giảm đáng kể ở các hạng thấp hơn do giới hạn của không gian biểu diễn. Để cải thiện kết quả, cần tinh chỉnh thêm các lớp cuối của mô hình hoặc sử dụng các phương pháp tối ưu hóa bổ sung để tăng khả năng phân biệt giữa các danh tính khác nhau. Nhìn chung, MobileNet-V2 sau Fine-Tuning đã đạt hiệu quả tốt với thời gian xử lý nhanh và khả năng học không gian biểu diễn hợp lý, phù hợp cho các ứng dụng yêu cầu tốc độ cao và tài nguyên hạn chế.

# Chương 4

## DEMO - Query Interface Design

Trong chương này, chúng tôi mô tả chi tiết quy trình triển khai demo của dự án truy xuất hình ảnh khuôn mặt sử dụng giao diện Streamlit. Chương trình cung cấp một nền tảng đơn giản, hiệu quả và dễ truy cập cho người dùng để thực hiện các bước truy xuất định danh hình ảnh. Tham khảo giao diện của phần DEMO ở Hình 4.1.



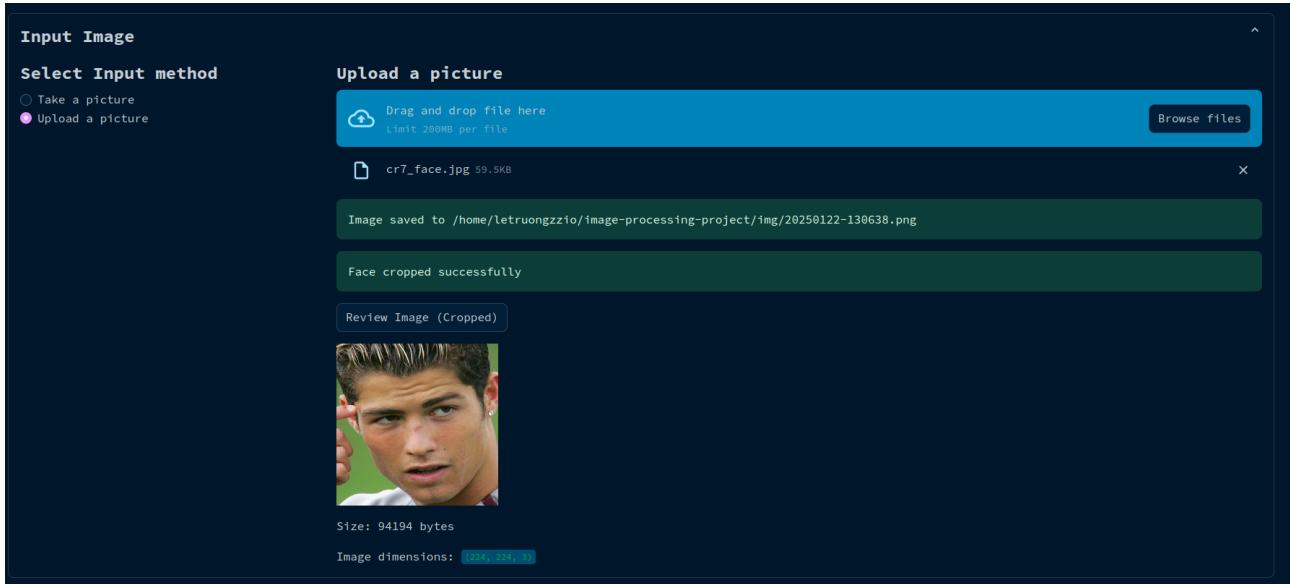
Hình 4.1: Giao diện và các chức năng chính của DEMO.

Dưới đây là cách thức hoạt động và quy trình vận hành chi tiết của demo.

- Nhập dữ liệu:** Người dùng có hai lựa chọn để cung cấp hình ảnh cho ứng dụng là: **Chụp ảnh** trực tiếp từ camera hoặc **Tải lên** một hình ảnh từ thiết bị.
- Tiền xử lý hình ảnh:** Sau khi nhập dữ liệu, hình ảnh được xử lý qua các bước sau:
  - Cắt khuôn mặt (crop):** Chương trình tự động xác định khuôn mặt trong hình ảnh đã nhập và tự động cắt khu vực trung tâm khuôn mặt phục vụ truy xuất.

- (b) **Tiền xử lý:** Hình ảnh được chuẩn hóa và biến đổi để đảm bảo tính nhất quán với dữ liệu huấn luyện.

Tham khảo các Bước 1 và 2 ở Hình 4.2.



Hình 4.2: Ví dụ về việc tải ảnh và tiền xử lý ảnh.

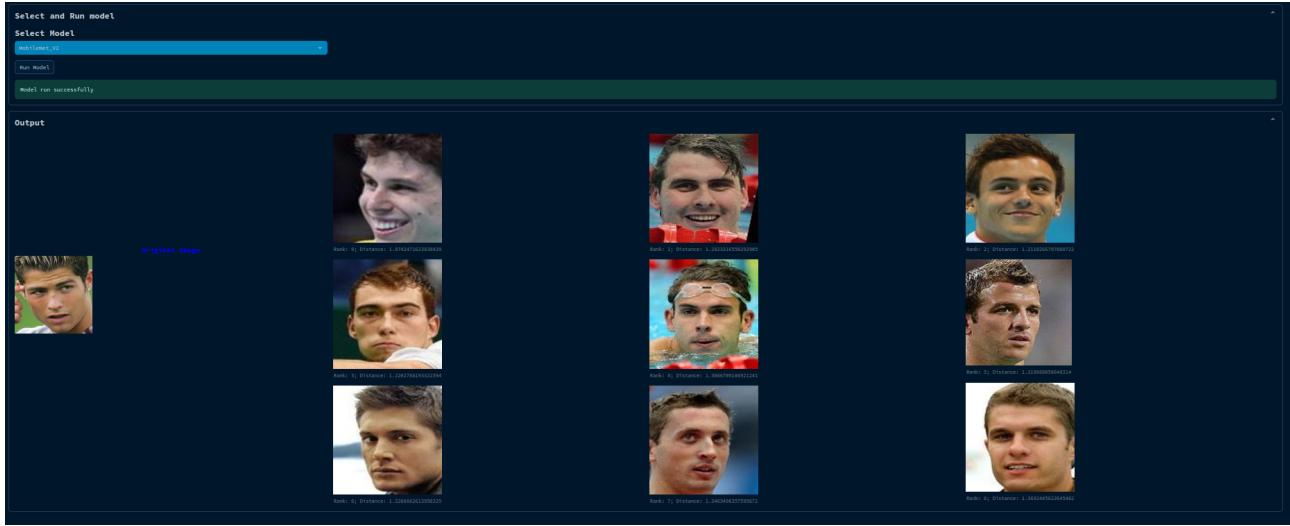
3. **Lựa chọn mô hình:** Người dùng có thể lựa chọn giữa hai mô hình truy xuất đã được fine-tune.

- (a) **ResNet-50:** Mô hình đạt độ chính xác cao nhất, phù hợp cho các bài toán đòi hỏi tính phân biệt cao.
- (b) **MobileNet-V2:** Mô hình nhẹ và nhạy, phù hợp cho các bối cảnh cần tính toán nhanh.

Phần mềm sử dụng mô hình đã chọn để trích xuất vector biểu diễn embedding từ hình ảnh người dùng nhập và thực hiện truy xuất các hình ảnh tương đồng từ tập dữ liệu gallery sử dụng KD-Tree.

4. **Trả kết quả:** Hệ thống sẽ hiển thị 9 hình ảnh tương đồng nhất với hình ảnh đã nhập, được sắp xếp theo thứ tự tương đồng (Rank 1 đến Rank 9). Kết quả gồm hình ảnh kèm theo thông tin chi tiết về mức độ tương đồng dựa trên các thuộc tính khuôn mặt. Tham khảo kết quả trả về từ mô hình MobileNet-V2 ở Hình 4.3.

Bạn đọc có thể tham khảo chi tiết về quá trình xử lý và truy vấn ảnh của DEMO thông qua video được cung cấp trong đường dẫn <https://s.net.vn/w51u>.



Hình 4.3: Kết quả 9 ảnh trả về từ mô hình MobileNet-V2.

**Tổng kết:** Mặc dù demo hiện tại còn một số hạn chế về triển khai và yêu cầu phần cứng, nhưng nó đã chứng minh tính khả thi và tiềm năng lớn trong bài toán truy xuất hình ảnh khuôn mặt. Bằng cách tiếp tục cải thiện và mở rộng, chúng tôi tin rằng ứng dụng này có thể được triển khai rộng rãi và mang lại giá trị thực tiễn cao trong nhiều lĩnh vực.

# Lời kết

Trong dự án này, nhóm đã phát triển và đánh giá hai mô hình học sâu ResNet-50 và MobileNet-V2 trong bài toán truy xuất hình ảnh khuôn mặt sử dụng hàm mất mát Triplet Loss. Thông qua quá trình chuẩn bị dữ liệu chi tiết, tinh chỉnh mô hình và phân tích so sánh, nhóm đã thu thập nhiều kết quả quan trọng về hiệu suất và độ phù hợp của từng mô hình trong các tình huống khác nhau.

## Những kết quả chính

**1. Hiệu quả của Triplet Loss:** Hàm mất Triplet Loss đã giúp các mô hình học được không gian biểu diễn vững chắc, nơi các hình ảnh có cùng danh tính được nhóm lại gần nhau, trong khi các hình ảnh khác danh tính được tách biệt rõ ràng. Hiệu quả này đã được kiểm chứng qua các chỉ số độ chính xác và khoảng cách trung bình.

**2. Hiệu suất của ResNet-50:**

- ResNet-50 cho thấy hiệu suất vượt trội về độ chính xác truy xuất và chất lượng không gian biểu diễn. Với Top-k Accuracy đạt 81.66% sau khi tinh chỉnh, ResNet-50 vượt trội MobileNet-V2 trên nhiều giá trị khác nhau.
- Tuy nhiên, mô hình này đòi hỏi chi phí tính toán cao hơn, với thời gian huấn luyện và suy diễn dài hơn, phù hợp cho các trường hợp cần đặt độ chính xác lên hàng đầu.

**3. Hiệu suất của MobileNet-V2:**

- MobileNet-V2 đạt độ chính xác khá (đạt 81.12% sau khi tinh chỉnh) trong khi giảm thiểu đáng kể chi phí tính toán. Kiến trúc tối ưu và thời gian xử lý nhạy giúp mô hình này phù hợp cho các tình huống đòi hỏi tốc độ cao và hạn chế tài nguyên.

**4. Tác động của Fine-Tuning:** Quá trình tinh chỉnh cả hai mô hình trên tập dữ liệu CelebA đã cải thiện đáng kể hiệu suất truy xuất. Quá trình

này đã tối đa hóa các lớp embedding, đảm bảo biểu diễn tốt hơn các đặc trưng khuôn mặt.

5. **Tích hợp KD-Tree:** Việc sử dụng KD-Tree cho việc lập chỉ mục và truy vấn các vector embedding đã cải thiện hiệu quả truy xuất, giảm đáng kể thời gian tính toán từ xuồng .

## Đóng góp

Dự án này đóng góp cho lĩnh vực truy xuất hình ảnh khuôn mặt bằng cách:

- Chứng minh hiệu quả của hàm mất mát Triplet Loss trong việc học biểu diễn phân biệt.
- Cung cấp phân tích chi tiết giữa ResNet-50 và MobileNet-V2 về độ chính xác, chi phí tính toán và khả năng ứng dụng thực tế.
- Giới thiệu một phương pháp truy xuất sử dụng KD-Tree, cân bằng tốc độ truy xuất và độ chính xác.
- Nhấn mạnh tầm quan trọng của việc tinh chỉnh các mô hình đã được tiền huấn luyện cho các bài toán đặc thù.

## Hạn chế

- Vì bộ dữ liệu này được lấy từ các diễn viên nổi tiếng ở Mỹ, bộ dữ liệu mất cân bằng lớn với số lượng người da màu ít hơn đáng kể với người da trắng. Vì vậy, khi ảnh đầu vào là người da màu, có thể mô hình xử lý không tốt.
- Hạn chế của bộ lọc Haar nằm ở việc vì bản chất thuật toán này là được huấn luyện, nên không đảm bảo chính xác 100%. Ta có thể bỏ ảnh vì bộ lọc không nhận ra chỉ có 1 gương mặt trong ảnh, cũng như không đảm bảo được crop chính xác vào mặt trong ảnh.
- Quá trình xử lý dữ liệu có giả định rằng ảnh đầu vào có 1 và chỉ 1 gương mặt. Nếu người dùng nhập vào ảnh không có hoặc có nhiều mặt, có thể mô hình sẽ đưa ra kết quả không tốt.
- Một trong những hạn chế chính của demo là yêu cầu phần cứng GPU để thực hiện quá trình xử lý hình ảnh và truy xuất hiệu quả. Tuy nhiên, Streamlit hiện không hỗ trợ việc triển khai GPU trên các môi trường trực tuyến như Streamlit Cloud, dẫn đến việc không thể cung cấp ứng dụng trực tiếp trên web để mọi người có thể sử dụng rộng rãi.

## Hướng phát triển trong tương lai

- **Mở rộng và đa dạng hóa bộ dữ liệu:** Trong tương lai, nhóm sẽ mở rộng bộ dữ liệu sử dụng, bao gồm các khuôn mặt thuộc nhiều sắc tộc, độ tuổi, giới tính và điều kiện ánh sáng khác nhau. Điều này giúp giảm thiểu sự thiên vị và tăng tính tổng quát của mô hình khi áp dụng trong thực tế.
- **Nghiên cứu các hàm mất tiên tiến hơn:** Bên cạnh Triplet Loss, nhóm sẽ thử nghiệm các hàm mất tiên tiến hơn như ArcFace, CosFace hay SphereFace để cải thiện độ phân biệt và khả năng nhận diện khuôn mặt trong không gian biểu diễn.
- **Tối ưu hóa hiệu suất tính toán:** Nhóm sẽ tiếp tục tìm kiếm các kỹ thuật nén và tối ưu hóa mô hình như Knowledge Distillation, Pruning và Quantization nhằm giảm thiểu chi phí tính toán, giúp mô hình dễ dàng triển khai trên các thiết bị di động hoặc môi trường không có GPU.
- **Tích hợp phương pháp tìm kiếm tiên tiến:** Thay vì KD-Tree, nhóm sẽ nghiên cứu các thuật toán tìm kiếm gần đúng khác như HNSW (Hierarchical Navigable Small World) hoặc IVF (Inverted File Index) để tăng tốc độ truy xuất mà vẫn duy trì độ chính xác cao.
- **Xây dựng hệ thống hỗ trợ đa gương mặt trong ảnh:** Nhóm sẽ phát triển các kỹ thuật xử lý ảnh để nhận diện và phân biệt nhiều khuôn mặt trong cùng một ảnh, từ đó nâng cao khả năng ứng dụng của mô hình trong các tình huống thực tế như giám sát hoặc phân tích video.
- **Phát triển ứng dụng thân thiện với người dùng:** Nhóm sẽ tối ưu hóa ứng dụng Streamlit để phù hợp với các môi trường trực tuyến khác hỗ trợ GPU, hoặc chuyển đổi sang các framework mạnh hơn như Flask, FastAPI hoặc sử dụng nền tảng cloud như AWS, Azure để triển khai dịch vụ web có hiệu suất cao.
- **Ứng dụng trong các lĩnh vực khác:** Ngoài bài toán nhận diện khuôn mặt, mô hình có thể được điều chỉnh để ứng dụng vào các lĩnh vực khác như truy xuất hình ảnh sản phẩm, giám sát an ninh, hoặc nhận diện biển số xe.
- **Tăng cường bảo mật và quyền riêng tư:** Để phù hợp với các yêu cầu bảo mật dữ liệu, nhóm sẽ tích hợp các kỹ thuật mã hóa và lưu trữ bảo mật nhằm đảm bảo rằng dữ liệu khuôn mặt người dùng không bị lộ hoặc sử dụng sai mục đích.

# Tài liệu tham khảo

- [VJ01] P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: [10.1109/CVPR.2001.990517](https://doi.org/10.1109/CVPR.2001.990517).
- [He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: [1512.03385 \[cs.CV\]](https://arxiv.org/abs/1512.03385). URL: <https://arxiv.org/abs/1512.03385>.
- [HA15] Elad Hoffer and Nir Ailon. “Deep Metric Learning Using Triplet Network”. In: *International Workshop on Similarity-Based Pattern Recognition*. 2015, pp. 84–92.
- [SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823.
- [How+17] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: [1704.04861 \[cs.CV\]](https://arxiv.org/abs/1704.04861). URL: <https://arxiv.org/abs/1704.04861>.
- [KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: [1412.6980 \[cs.LG\]](https://arxiv.org/abs/1412.6980). URL: <https://arxiv.org/abs/1412.6980>.
- [Ti] Vũ Hữu Tiệp. *Machine Learning cơ bản*. Revised edition, March 27, 2018. Nhà xuất bản Khoa học và Kỹ thuật, 2017. Chap. 29, pp. 364–378.
- [DVL21] Nguyen Dinh, Thanh Van, and MANH LE. “Phân lớp ảnh bằng cây KD-Tree cho bài toán tìm kiếm ảnh tương tự”. In: *Journal of Research and Development on Information and Communication Technology* (June 2021), pp. 40–52. DOI: [10.32913/mic-ict-research-vn.v2021.n1.966](https://doi.org/10.32913/mic-ict-research-vn.v2021.n1.966).