

Fashion Image Search Engine

Nguyễn Công Nguyên^{1, 2}, Lê Minh Nguyệt^{1, 2}, Hồ Đức Trưởng^{1, 2}

¹Khoa Khoa học máy tính, Trường Đại học Công nghệ Thông tin

²Đại học Quốc gia Thành phố Hồ Chí Minh

{21521200, 21521211, 21522730}@gm.uit.edu.vn

Abstract

Báo cáo này trình bày quy trình xây dựng một công cụ truy vấn hình ảnh liên quan trong lĩnh vực thời trang, với đặc trưng được trích xuất từ ba mô hình pre-trained VGG16, ResNet50 và Xception. Kết quả đánh giá qua độ đo *mAP* cho thấy mô hình Xception cho kết quả tốt nhất, với *mAP* là 0.724. Qua thực nghiệm trên các mô hình CNN kể trên, nhóm đã có những quan sát và nhận xét đặc điểm của các loại mạng học sâu đối với hiệu suất trong ngữ cảnh bài toán truy vấn hình ảnh.

1 Tổng quan đề tài

1.1 Bối cảnh

Trong thời đại ngày nay, việc mua sắm trực tuyến đang trở nên ngày càng phổ biến, đặc biệt là trong lĩnh vực thời trang. Nhu cầu của người dùng tăng cao và cũng ngày một trở nên phức tạp. Việc tìm kiếm những sản phẩm mong muốn có thể trở nên khá thách thức khi người tiêu dùng muốn định vị những mẫu sản phẩm tương tự với hình ảnh họ thấy trên mạng Internet, hay đơn giản là giống với một sản phẩm mà họ vô tình nhìn thấy và chụp lại.

Nhận thức được điều đó, nhóm đã quyết định thực hiện phát triển một hệ thống truy vấn hình ảnh trong lĩnh vực thời trang, với mục tiêu giúp người dùng có thể dễ dàng tìm kiếm và xác định những sản phẩm thông qua hình ảnh cụ thể.

1.2 Lí do lựa chọn đề tài

1.2.1 Tăng cường trải nghiệm người dùng

Người dùng thường gặp khó khăn trong việc tìm kiếm các sản phẩm thời trang trực tuyến, do số lượng sản phẩm lớn và sự đa dạng về mẫu

mã. Hệ thống truy vấn hình ảnh giúp người dùng dễ dàng tìm thấy những sản phẩm mà họ yêu thích chỉ bằng cách cung cấp một hình ảnh, tiết kiệm thời gian và công sức so với việc phải tìm kiếm thủ công.

1.2.2 Nâng cao hiệu quả kinh doanh

Các cửa hàng thời trang trực tuyến có thể tận dụng công nghệ này để cải thiện khả năng gợi ý sản phẩm, từ đó tăng tỷ lệ chuyển đổi và doanh thu. Khả năng tìm kiếm sản phẩm tương tự một cách nhanh chóng và chính xác có thể khuyến khích người dùng mua sắm nhiều hơn.

1.2.3 Xu hướng cá nhân hóa

Việc sử dụng hình ảnh để tìm kiếm sản phẩm giúp đáp ứng tốt hơn nhu cầu cá nhân của từng người dùng. Các gợi ý dựa trên sở thích và phong cách cá nhân giúp tăng cường sự hài lòng và trung thành của khách hàng.

1.3 Ứng dụng

1.3.1 Các nền tảng mua sắm trực tuyến

Ứng dụng truy vấn hình ảnh có thể tích hợp vào các trang web và ứng dụng mua sắm, giúp người dùng dễ dàng tìm kiếm sản phẩm theo hình ảnh.

1.3.2 Công cụ gợi ý thời trang

Các ứng dụng gợi ý thời trang có thể sử dụng công nghệ này để đưa ra những gợi ý phù hợp với phong cách và sở thích của người dùng, từ đó giúp họ dễ dàng tạo nên các bộ trang phục hoàn hảo.

035
036
037
038
039

040
041
042
043
044
045
046

047
048
049
050
051
052

053
054
055
056
057
058

059
060
061
062
063
064

1.3.3 Quản lý & kiểm kê kho hàng

Các hệ thống quản lý kho có thể sử dụng công nghệ này để tự động nhận diện và phân loại sản phẩm, giúp việc kiểm kê và quản lý kho trở nên hiệu quả hơn.

1.3.4 Mạng xã hội & cộng đồng thời trang

Các ứng dụng mạng xã hội về thời trang có thể sử dụng tính năng truy vấn hình ảnh để kết nối người dùng với những sản phẩm và xu hướng mới nhất, từ đó tạo ra một cộng đồng chia sẻ và trao đổi thông tin phong phú hơn.

1.3.5 Ngành công nghiệp bán lẻ

Các cửa hàng bán lẻ có thể sử dụng công nghệ này để cung cấp dịch vụ tìm kiếm sản phẩm cho khách hàng tại chỗ, giúp họ dễ dàng tìm thấy các sản phẩm tương tự ngay trong cửa hàng.

2 Định nghĩa bài toán

2.1 Tổng quát bài toán

Bài toán sẽ được giải quyết trong báo cáo này là bài toán truy vấn hình ảnh liên quan trong phạm vi lĩnh vực thời trang. Người dùng chỉ cần cung cấp một hình ảnh của sản phẩm mong muốn, chẳng hạn như giày, quần áo, hoặc ví. Kết quả trả về sẽ là một danh sách các sản phẩm tương tự hoặc có tính tương đồng với đầu vào, danh sách sẽ được sắp xếp theo thứ tự giảm dần về độ tương đồng.



Hình 1: Minh họa input của bài toán

2.2 Mô tả bài toán

Input (Đầu vào): Một hình ảnh truy vấn (query image) chứa sản phẩm thời trang mà người dùng quan tâm như quần áo, giày dép, ví, v.v.

Output (Đầu ra): Một danh sách được xếp hạng các hình ảnh có độ tương đồng cao nhất với hình ảnh truy vấn, với hình ảnh giống nhất đứng đầu danh sách.

Ví dụ về input và output của bài toán được minh họa trong các Hình 1 và 2.

2.3 Bộ dữ liệu

Bộ dữ liệu Farfetch Listings (TAN, 2019) là một tập dữ liệu lớn chứa thông tin về các sản phẩm thời trang được bán trên nền tảng thương mại điện tử Farfetch. Bộ dữ liệu này được cung cấp cho cộng đồng Kaggle để hỗ trợ các nghiên cứu và phát triển các thuật toán học máy trong lĩnh vực thời trang. Tác giả đã thu thập dữ liệu từ máy chủ FarFetch bằng cách sử dụng thư viện Requests trong Python, sau đó chuẩn hóa và lưu trữ trong một DataFrame bằng Pandas.

Dữ liệu bao gồm một file CSV là current_farfetch_listings.csv và hai thư mục cutout-img, model-img chứa hình ảnh thời trang.

File current_farfetch_listings.csv chứa 23 trường thông tin về 188,817 mẫu, bao gồm các trường:

- images.model: Đường dẫn tới hình ảnh sản phẩm thời trang khi được mặc với model.
- images.cutOut: Đường dẫn tới hình ảnh chỉ chứa sản phẩm.
- brand.name: Nhãn hiệu sản phẩm.
- id: Mã định danh của hình ảnh tương ứng với name của hình ảnh trong thư mục cutout-img và model-img.
- availableSizes,
priceInfo.discountLabel,
priceInfo.formattedFinalPrice, v.v.

Chi tiết các trường dữ liệu trong Hình 3.

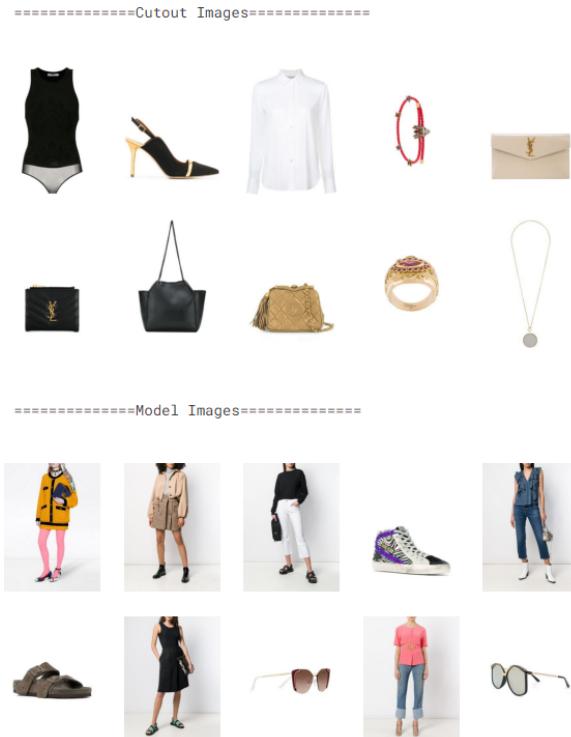
Hai thư mục cutout-img và model-img, mỗi thư mục đều chứa 187,620 hình ảnh thời



Hình 2: Minh họa output của input trong hình 1

#	Column	Non-Null Count	Dtype
0	availableSizes	162711	non-null object
1	brand.id	188817	non-null int64
2	brand.name	188817	non-null object
3	gender	188817	non-null object
4	hasSimilarProducts	188817	bool
5	id	188817	non-null int64
6	images.cutout	188817	non-null object
7	images.model	188817	non-null object
8	isCustomizable	188817	bool
9	merchandiseLabel	56441	non-null object
10	merchandiseLabelField	188817	non-null object
11	merchantId	188817	non-null int64
12	priceInfo.currencyCode	188817	non-null object
13	priceInfo.discountLabel	68287	non-null object
14	priceInfo.finalPrice	188817	non-null int64
15	priceInfo.formattedFinalPrice	188817	non-null object
16	priceInfo.formattedInitialPrice	188817	non-null object
17	priceInfo.initialPrice	188817	non-null int64
18	priceInfo.installmentsLabel	0	non-null float64
19	priceInfo.isOnSale	188817	non-null bool
20	shortDescription	188817	non-null object
21	stockTotal	188817	non-null int64
dtypes: bool(3), float64(1), int64(6), object(12)			

Hình 3: Các trường dữ liệu trong bộ Farfetch Listings



Hình 4: Hình ảnh trong thư mục Cutout và Model

136
137
138
trang từ trang web FarFetch, với định dạng
.jpg. Các hình ảnh trong hai thư mục xem [Hình 4](#).

2.4 Gán nhãn dữ liệu

Nhóm thực hiện gán nhãn cho các mẫu dữ liệu bằng cách dựa vào từ cuối cùng cột "short-Description". Ví dụ với mẫu có "shorDescrption" là "logo print strap sandals", nhóm sẽ gán nhãn "sandals" cho dữ liệu, còn phần mô tả là "FendiMania sock style sneakers" sẽ được gán nhãn "sneakers"

Kết quả thu được 100 nhãn (nhóm chỉ chọn những nhãn có hơn 100 mẫu dữ liệu). Tuy nhiên, trong 100 nhãn này tồn tại những nhãn

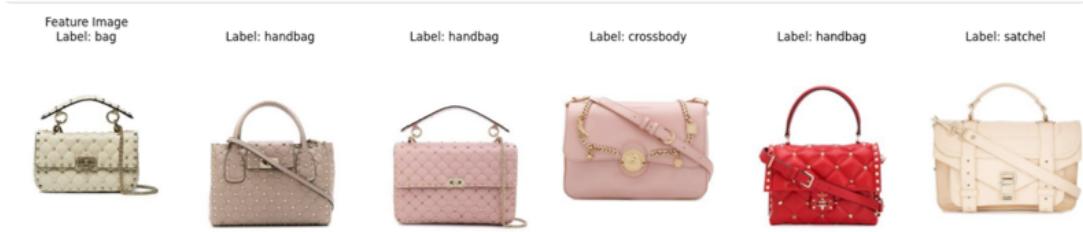
là từ đồng nghĩa, nên nhóm tiến hành gom nhóm thủ công lại. Minh họa trong [Hình 5](#). Sau khi tiến hành gộp các từ đồng nghĩa trong 100 nhãn nhóm thu được 43 nhóm vs 2 nhãn không xác định, nhóm sẽ quyết định không sử dụng 2 nhãn này. Chi tiết các hình ảnh thuộc hai nhãn này trong [Hình 6](#) và [7](#).

2.5 Phân chia tập dữ liệu

Nhóm thực hiện chia bộ dữ liệu thành hai tập như sau.

- Tập train: chỉ chứa những loại có hơn 80

150	
151	
152	
153	
154	
155	
156	
157	
158	
159	
160	



Hình 5: Minh họa các hình ảnh với các nhãn được gán



Hình 6: Các ảnh thuộc nhãn không xác định thứ nhất "detail"

161 mẫu, tổng cỗng 7840 mẫu.

- 162 • Tập test: chỉ chứa những loại có hơn 20
163 mẫu, tổng cỗng 1960 mẫu.

164 Kết quả thu được như Hình 8.

165 3 Cơ sở lý thuyết

166 3.1 Hệ thống truy vấn hình ảnh

167 Cốt lõi của một hệ thống truy vấn hình ảnh
168 (Image Retrieval System) là tính toán sự tương
169 đồng (similarity) giữa hình ảnh truy vấn và các
170 hình ảnh có trong cơ sở dữ liệu để trích xuất
171 những hình ảnh có độ tương đồng cao nhất.

172 Kiến trúc của một hệ thống truy vấn hình
173 ảnh thông thường được thể hiện trong Hình 9.

174 Đầu tiên, các hình ảnh trong cơ sở dữ liệu sẽ
175 được trích xuất đặc trưng và lưu trữ lại. Khi có
176 một yêu cầu truy vấn, hình ảnh mới đó cũng sẽ
177 được trích xuất đặc trưng, độ tương đồng được
178 tính toán dựa trên các đặc trưng này và những
179 đặc trưng của hình ảnh trong cơ sở dữ liệu.

3.2 CNN & ImageNet

CNN (Convolutional Neural Network hay Mạng nơ-ron tích chập) là một loại mô hình học sâu phổ biến, đặc biệt được thiết kế để xử lý và phân loại ảnh, nhờ vào các lớp như Convolutional, Activation, Pooling, Fully Connected, Residual Connection, v.v.

Tổng quan kiến trúc của một CNN được mô tả trong Hình 10.

ImageNet là một tập dữ liệu lớn chứa hình ảnh được phân loại vào hàng nghìn danh mục khác nhau, được sử dụng rộng rãi trong lĩnh vực máy học và thị giác máy tính để đánh giá và so sánh hiệu suất của các mô hình học sâu, đặc biệt là các CNNs. Trong cuộc thi hàng năm ILSVRC về ImageNet, nhiều kiến trúc CNN nổi tiếng đã được tạo ra, có tính ứng dụng mạnh mẽ và đóng vai trò nền tảng đối với các nghiên cứu trong lĩnh vực thị giác máy tính. Một số kiến trúc đạt hạng cao trong cuộc thi như ResNet, VGG, Xception sẽ được sử dụng trong đồ án này.

Việc sử dụng các mô hình đã được huấn



Hình 7: Các ảnh thuộc nhãn không xác định thứ hai "set"

203
204
205
206
207
luyện trước trên tập dữ liệu lớn như ImageNet
mang lại nhiều lợi ích không chỉ về mặt hiệu
suất mà còn về mặt thời gian. Các mô hình này
cũng đã được công bố và có thể truy cập một
cách dễ dàng.

208 3.3 Học chuyển giao

209
210
211
212
213
214
Học chuyển giao (Transfer Learning) là một
kỹ thuật trong học máy mà một mô hình được
huấn luyện trước đó trên một tập dữ liệu lớn,
được chuyển giao (transfer) để làm nền tảng
cho việc huấn luyện một mô hình mới trên một
tập dữ liệu tương đối nhỏ hoặc có liên quan
đến tác vụ mới.

215
216
217
218
219
220
221
222
223
Ý tưởng cơ bản của học chuyển giao là sử
dụng kiến thức được học từ một tác vụ trước
đó, để cải thiện hiệu suất của mô hình trên một
tác vụ khác liên quan mà có ít dữ liệu huấn
luyện. Thay vì bắt đầu từ việc huấn luyện một
mô hình từ đầu, ta sử dụng một mô hình đã
được huấn luyện trước (pre-trained model) và
điều chỉnh nó cho tác vụ mới.

224
225
Sự khác biệt giữa Học thông thường và Học
chuyển giao được thể hiện trong Hình 11.

226
227
228
229
230
231
232
233
Học chuyển giao cung cấp nhiều lợi ích như
tăng tốc quá trình huấn luyện, và đạt được hiệu
suất tốt hơn với ít dữ liệu. Dựa trên ý tưởng của
học chuyển giao, đồ án sử dụng một số kiến
trúc pre-trained CNN thường được sử dụng phổ
biến trong các tác vụ thị giác máy tính để trích
xuất các đặc trưng từ hình ảnh (Manchanda,
2022).

4 Phương pháp trích xuất đặc trưng

234 4.1 VGG16

235
VGG16 là một CNN được cấu thành từ 2 khối
chính. Kiến trúc được biểu diễn trong Hình 12.

236
237
Khối đầu tiên là điểm đặc biệt của loại mạng
này vì nó hoạt động như một bộ trích xuất đặc
trưng. Lớp đầu tiên lọc hình ảnh với các convo-
lution kernels và trả về "feature maps", sau đó
được chuẩn hóa bởi một hàm kích hoạt (ReLU)
hoặc thay đổi kích thước (pooling) (Simonyan
and Zisserman, 2014) (Smeda, 2022).

238
239
240
Việc lặp qua lặp lại quá trình filter feature
maps này mang lại các bản đồ đặc trưng mới
để chuẩn hóa và thay đổi kích thước.

241
242
Cuối cùng các giá trị của các bản đồ đặc
trưng được nối thành 1 vector (đầu ra của khối
đầu tiên và là đầu vào của khối thứ 2).

243
244
Cũng như các mạng nơ-ron thông thường,
các tham số của các lớp được xác định bằng
cách lan truyền ngược Gradient, hàm Cross-
Entropy được tối ưu hóa trong giai đoạn huấn
luyện.

245
246
Cách thành phần chính của VGG16 như sau.

247 4.1.1 Đầu vào

248
249
 $W_x H_x D$: rộng x , cao x số kênh (1 cho ảnh đen
trắng, 3 cho ảnh màu RGB) với x là 224.

250 4.1.2 Lớp tích chập

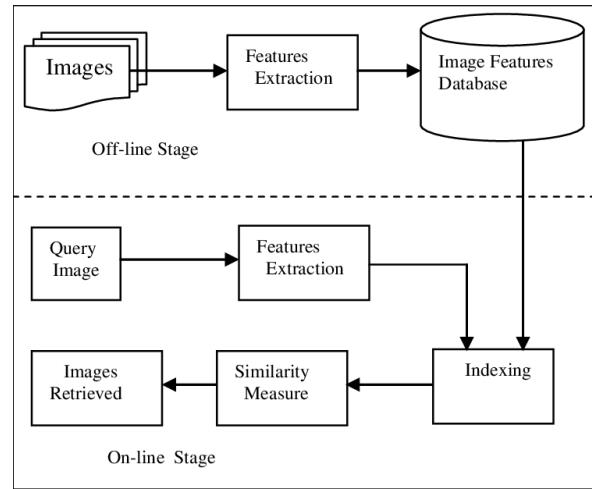
251
252
Lớp tích chập là thành phần chính của CNN,
giúp xác định các đặc trưng hình ảnh bằng cách
thực hiện phép lọc tích chập.

253
254
Các Siêu tham số của lớp tích chập:

giaycode	800	giaycode	200
transucnho	640	transucnho	160
tuixach	560	tuixach	140
aolenni	480	aolenni	120
quanaudai	400	quanaudai	100
vi	320	vi	80
aokhoacdai	320	aokhoacdai	80
aocanh	240	vay	60
vay	240	aocanh	60
giaycaogot	160	aophong	40
dep	160	aokhoangan	40
jumpsuit	160	aoghile	40
bodysuit	160	giaycaogot	40
aophong	160	matkinh	40
mu	160	dep	40
aoquanboi	160	daychuyen	40
matkinh	160	aoquanboi	40
aokhoangan	160	mu	40
aochoang	160	jumpsuit	40
aovest	160	aovest	40
quanlot	160	quanlot	40
daychuyen	160	bodysuit	40
aoghile	160	aochoang	40
khanco	80	quanjean	20
quanshort	80	balo	20
quanjean	80	khanco	20
caibangdau	80	thatlung	20
aolotnguc	80	opdienthoai	20
thatlung	80	aolotnguc	20
gangtay	80	tatchan	20
charm	80	quanshort	20
tatchan	80	quanlegging	20
quanlegging	80	daymockhoa	20
daydeo	80	dongho	20
mulen	80	gangtay	20
opdienthoai	80	mulen	20
daymockhoa	80	daydeo	20
dongho	80	charm	20
aongunu	80	lythuytinh	20
nen	80	caibangdau	20
lythuytinh	80	aongunu	20
caidia	80	nen	20
		caidia	20

Hình 8: Số lượng mẫu nhãn trong tập train và test

- Số lượng filters K .
- Bộ lọc kích thước F : Mỗi bộ lọc có kích thước $F \times F \times D$.
- Stride: Số pixel di chuyển qua mỗi bước, nếu stride càng cao kích thước feature maps càng nhỏ.
- Zero-padding: Nếu có 1 số lớp tích chập ở sau trong khi kích thước feature maps quá nhỏ, khi đó sẽ bổ sung các pixel 0 vào mỗi bên của hình ảnh, giúp cho feature maps cùng kích thước với đầu vào. Minh họa trong Hình 14.



Hình 9: Kiến trúc hệ thống truy vấn hình ảnh

Hoạt động của phép tích chập: Phép toán nhân tổng hợp theo từng phần tử giữa 1 phần hình ảnh với filter. Chi tiết xem [Hình 13, 15 \(Pramoditha, 2022\)](#). 277
278
279
280

4.1.3 Lớp pooling (lớp gộp) 281
Lớp này được đặt giữa hai lớp tích chập và áp dụng phép pooling lên mỗi bản đồ đặc trưng. Phép pooling giúp giảm kích thước của hình ảnh trong khi vẫn bảo toàn các đặc điểm quan trọng. Phép maxpooling thường được sử dụng, với các cell kích thước 2×2 liền kề hoặc 3×3 với bước di chuyển là 2. Minh họa trong [Hình 16 \(Pramoditha, 2022\)](#). 282
283
284
285
286
287
288
289

Giúp giảm số lượng tham số và chi phí tính toán. Từ đó, cải thiện hiệu suất của mạng và tránh over-learning. 290
291
292
293

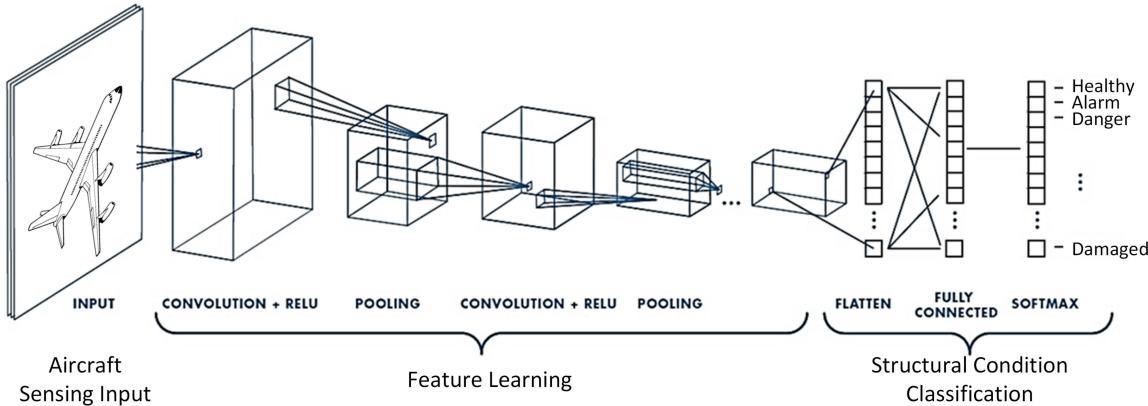
Các siêu tham số:

- Kích thước của ô: Hình ảnh được chia thành các ô vuông $F \times F$. 294
295
- Stride: $F = 2$ và $S = 2$ là một lựa chọn tốt. Việc chọn các ô lớn hơn gây mất mát quá nhiều thông tin và dẫn đến kết quả kém tốt hơn trong thực tế. 296
297
298
299

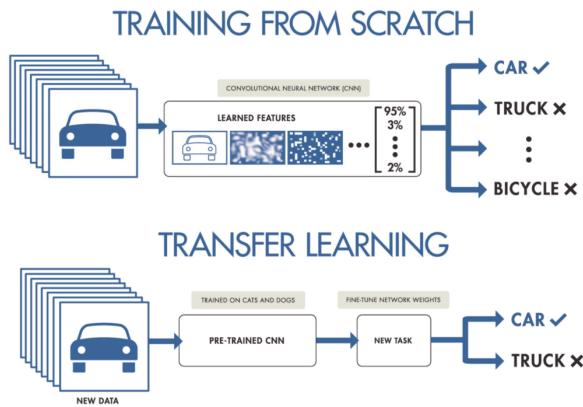
4.1.4 Lớp Flatten 300
Đầu ra đc trả về từ lớp gộp cuối cùng sẽ được làm phẳng. Minh họa trong [Hình 17](#). 301
302

4.1.5 Lớp ReLU 303
ReLU (Rectified Linear Units) đề cập đến hàm phi tuyến thực được định nghĩa bởi:

$$ReLU(x) = \max(0, x)$$



Hình 10: Hình minh họa tổng quan kiến trúc CNN



Hình 11: Học thông thường và Học chuyển giao

Về mặt trực quan, nó trông như [Hình 18](#).

Lớp hiệu chỉnh ReLU thay thế tất cả các giá trị âm nhận được làm đầu vào bằng số không. Nó hoạt động như một hàm kích hoạt.

ReLU giúp giảm bớt chi phí tính toán so với các hàm kích hoạt khác, hạn chế vấn đề vanishing gradient (đạo hàm trở nên cực kì nhỏ trong quá trình lan truyền ngược (đạo hàm nhiều bậc), dẫn đến trọng số mạng không được cập nhật đáng kể, khiến mạng cập nhật chậm hoặc không học được gì từ dữ liệu, giúp cho mô hình học nhanh và hiệu quả hơn.

4.1.6 Minh họa cách sử dụng VGG16 và cấu hình chi tiết

Cách sử dụng VGG16 của thư viện Keras trong [Hình 19](#) ([Intelligence, 2019b](#)).

Cấu hình chi tiết VGG16 trong [Bảng 1](#) ([Smeda, 2022](#)).

4.2 ResNet50

ResNet (Residual Network - Mạng nơ-ron dư thừa) được giới thiệu đến công chúng vào năm 2015 và đã giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Hiện tại có rất nhiều biến thể của kiến trúc ResNet với số lớp khác nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152,... Với tên là ResNet theo sau là một số chỉ kiến trúc ResNet với số lớp tích chập nhất định.

ResNet là một trong những kiến trúc mạng sâu tiên tiến, giúp giải quyết vấn đề vanishing gradient khi mạng nơ-ron trở nên quá sâu. Trong các mạng nơ-ron sâu, dữ liệu đầu vào thường đánh mất dần các thông tin quan trọng qua nhiều lớp, cuối cùng dẫn đến những dự đoán không mấy ý nghĩa. Vì vậy, thay vì hoàn toàn dựa vào kết quả học tập qua các lớp của mạng để biến đổi đầu vào thành một dự đoán nào đó mà có thể không liên quan đến giá trị thực sự, ResNet học cách giữ lại thông tin đầu vào và chỉ học thêm các thay đổi nhỏ qua từng lớp để đưa nó đến gần hơn với giá trị đầu ra thực sự ([He et al., 2015](#)).

4.2.1 Đặc trưng của ResNet

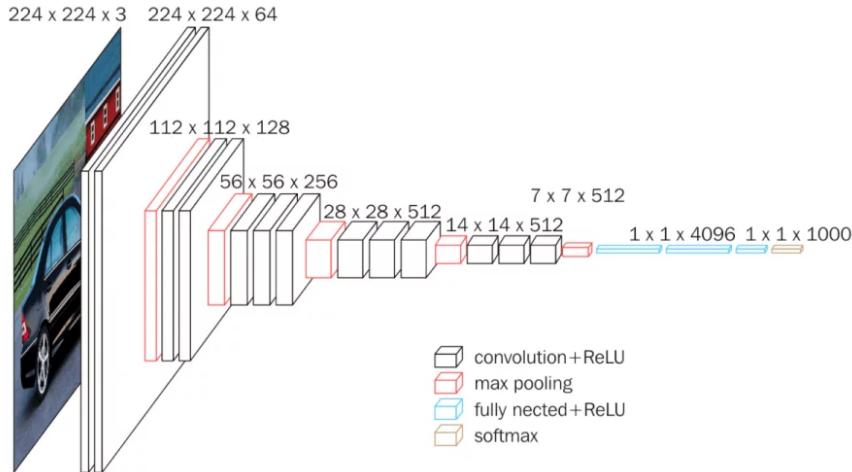
Dựa trên ý tưởng chung, cấu trúc của một ResNet được phát triển từ một CNN thông thường với các điều chỉnh sau:

- **Residual Blocks**

Điểm đặc trưng của ResNet là các Residual Blocks. Thay vì chỉ truyền dữ liệu huấn luyện theo một đường qua các lớp, ResNet nhóm một số lớp tích chập lại và

304	322
305	323
306	324
307	325
308	326
309	327
310	328
311	329
312	330
313	331
314	332
315	333
	334
	335
	336
	337
	338
	339
	340
	341
	342
	343
	344
	345

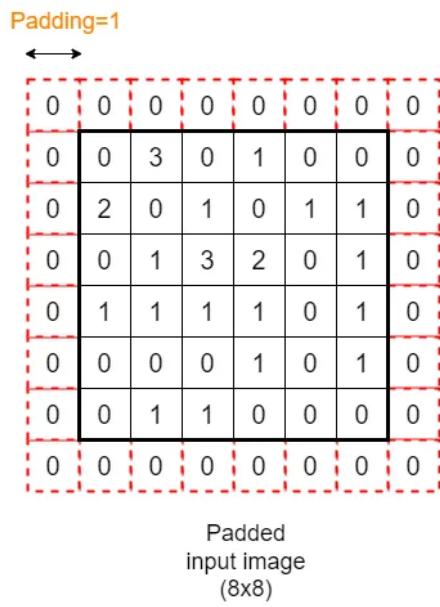
316	346
317	347
318	348
319	349
320	350
321	351
	352
	353
	354



Hình 12: Kiến trúc VGG16

```
# Row-wise
(0*0 + 3*1 + 0*1) + (2*0 + 0*1 + 1*0) + (0*1 + 1*0 + 3*0) = 3
```

Hình 13: Công thức cho phép tích chập trong Hình 15a



Hình 14: Padding trong CNN

355
356
357
358
359
360
361
362
363
364

gọi nó là một Residual Block. Khi đó, dữ liệu đầu vào của một Block không chỉ đi qua các lớp bên trong để học những thay đổi mà còn được truyền trực tiếp (identity mapping) đến đầu ra của Block tạo thành đầu ra hoàn chỉnh cho Block. Đầu ra này sẽ mang cả thông tin hữu ích từ đầu vào, tránh trường hợp mất thông tin quan trọng trong quá trình Forward Propagation ([Kundu, 2023](#)).

• Shortcut Connections

Đây là các kết nối từ đầu vào đến đầu ra của một Block, cho phép tín hiệu được truyền trực tiếp qua các lớp mà không cần phải trải qua tất cả các lớp bên trong block, duy trì tín hiệu và giảm thiểu mất mát thông tin.

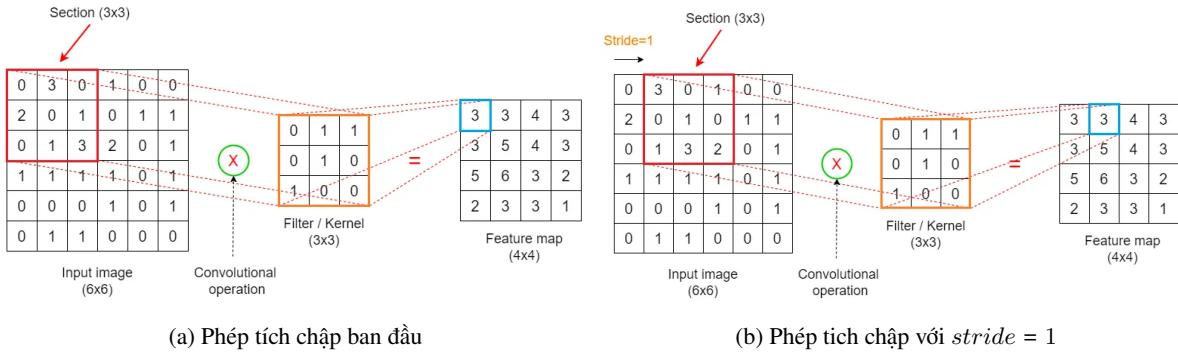
Minh họa về Shortcut Connection trong [Hình 20](#).

Như vậy, trong trường hợp gradient vanishing diễn ra do đầu ra của một lớp gần như bằng 0, Residual Block khắc phục bằng cách chuyển đầu vào từ trước biến đổi của lớp đó đến đầu ra cuối cùng. Điều này làm giảm sự ảnh hưởng của các lớp trung gian trong quá trình học.

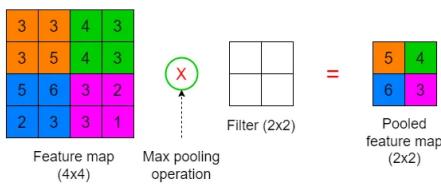
Trong đồ án này, nhóm lựa chọn ResNet50 để tiến hành thực nghiệm. ResNet50 với 50 lớp tích chập và Fully-Connected, có độ sâu vừa đủ để học các đặc trưng phức tạp trong ngữ cảnh bài toán của nhóm. ResNet34 có số lượng tham số ít hơn, nên có thể bị hạn chế trong việc học các đặc trưng phức tạp từ dữ liệu, còn ResNet100+ có khả năng học các đặc trưng rất phức tạp, tuy nhiên, điều này có thể dẫn đến overfitting nếu không có đủ dữ liệu hoặc không có các kỹ thuật regularization phù hợp.

ResNet50 là một mô hình trung bình giữa ResNet34 và ResNet100+, yêu cầu tài nguyên tính toán vừa phải và thời gian huấn luyện hợp lý, trong khi vẫn đạt được độ chính xác cao.

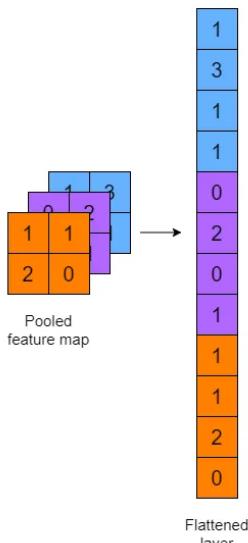
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395



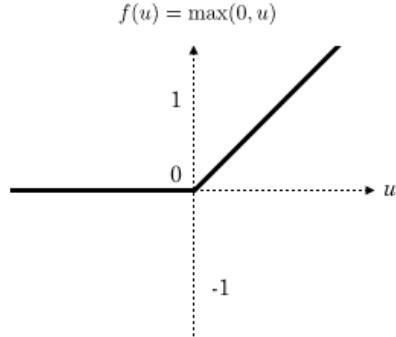
Hình 15: Phép tích chập trên CNN



Hình 16: Lớp Max Pooling



Hình 17: Lớp Flatten



Hình 18: Đồ thị hàm ReLU

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model

def create_vgg16_model():
    # Tải mô hình VGG-16 đã được huấn luyện trước trên tập dữ liệu ImageNet
    base_model = VGG16(weights='imagenet')
    # Tạo mô hình mới với đầu vào là đầu vào của VGG-16 và đầu ra là lớp 'fc1'
    model = Model(inputs=base_model.input, outputs=base_model.get_layer('fc1').output)
    return model

def extract_features(img_path, model):
    # Mở ảnh và thay đổi kích thước thành 224x224
    img = Image.open(img_path)
    img = img.resize((224, 224))
    # Chuyển đổi ảnh sang RGB
    img = img.convert('RGB')
    # Chuyển đổi ảnh thành mảng numpy
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    # Trích xuất đặc trưng
    features = model.predict(x)
    # Chuẩn hóa đặc trưng
    features = features / np.linalg.norm(features)
    return features
```

Hình 19: Cách sử dụng VGG16

396
397
Hình 21 so sánh cấu trúc các lớp của các
mạng ResNet khác nhau (Ruiz, 2024).

4.2.2 Cấu trúc ResNet50

398
399
400
401
402
403
404
ResNet50 có tổng cộng 49 lớp tích chập và
1 lớp Fully-Connected. Thông thường, một
Residual Block trong ResNet sẽ có 3 lớp tích
chập. ResNet50 nhận đầu vào là một hình ảnh
có kích thước 224x224x3 và được chia thành 6
phân đoạn (Sharma, 2023), lần lượt như sau:

- Stage 1: Bao gồm 1 lớp tích chập khởi

405
tạo, cùng Batch Normalization và Max
Pooling.

- Stage 2, 3, 4, 5: Lần lượt bao gồm 3, 4, 6,
3 Residual Blocks với số lượng filter và
kernel size trong mỗi lớp là khác nhau.
- Stage 6: Kết quả từ Stage 5 sẽ được đưa
vào một lớp Global Average Pooling, đây
cũng là thời điểm đặc trưng trong bài toán
của nhóm sẽ được trích xuất. Sau đó, tiến

406
407

408

409

410

411

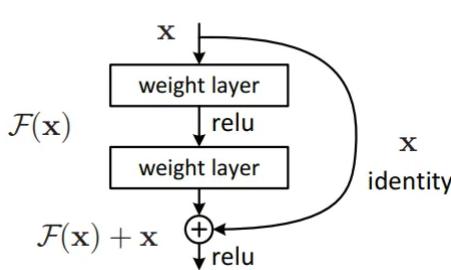
412

413

414

Layer Type	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544

Bảng 1: Cấu hình chi tiết của VGG16



Hình 20: Shortcut Connection trong ResNet

415 hành Flatten và Fully-Connected với kết
416 quả thu được.

417 Cụ thể kiến trúc của ResNet50 được thể hiện
418 trong Hình 22 (Sharma, 2023).

419 Qua từng stage, chiều cao và rộng dữ liệu sẽ
420 giảm dần, trong khi chiều sâu thì tăng lên. Đặc
421 trưng được trích xuất có kích thước $1 \times 1 \times 2048$.

422 Các thành phần chính của một Residual 423 Block

424 Lớp tích chập: Trong mỗi Residual Block,
425 các lớp tích chập được sử dụng để học các đặc
426 trưng.

layer name	34-layer	50-layer	101-layer
conv1		$7 \times 7, 64, \text{stride } 2$	
conv2_x	$[3 \times 3, 64] \times 3$ $[3 \times 3, 64]$	$[1 \times 1, 64] \times 3$ $[3 \times 3, 64]$ $[1 \times 1, 256]$	$[1 \times 1, 64] \times 3$ $[3 \times 3, 64]$ $[1 \times 1, 256]$
conv3_x	$[3 \times 3, 128] \times 4$ $[3 \times 3, 128]$	$[1 \times 1, 128] \times 4$ $[3 \times 3, 128]$ $[1 \times 1, 512]$	$[1 \times 1, 128] \times 4$ $[3 \times 3, 128]$ $[1 \times 1, 512]$
conv4_x	$[3 \times 3, 256] \times 6$ $[3 \times 3, 256]$	$[1 \times 1, 256] \times 6$ $[3 \times 3, 256]$ $[1 \times 1, 1024]$	$[1 \times 1, 256] \times 23$ $[3 \times 3, 256]$ $[1 \times 1, 1024]$
conv5_x	$[3 \times 3, 512] \times 3$ $[3 \times 3, 512]$	$[1 \times 1, 512] \times 3$ $[3 \times 3, 512]$ $[1 \times 1, 2048]$	$[1 \times 1, 512] \times 3$ $[3 \times 3, 512]$ $[1 \times 1, 2048]$
		average pool, 2048-d fc	

Hình 21: So sánh các phiên bản ResNet

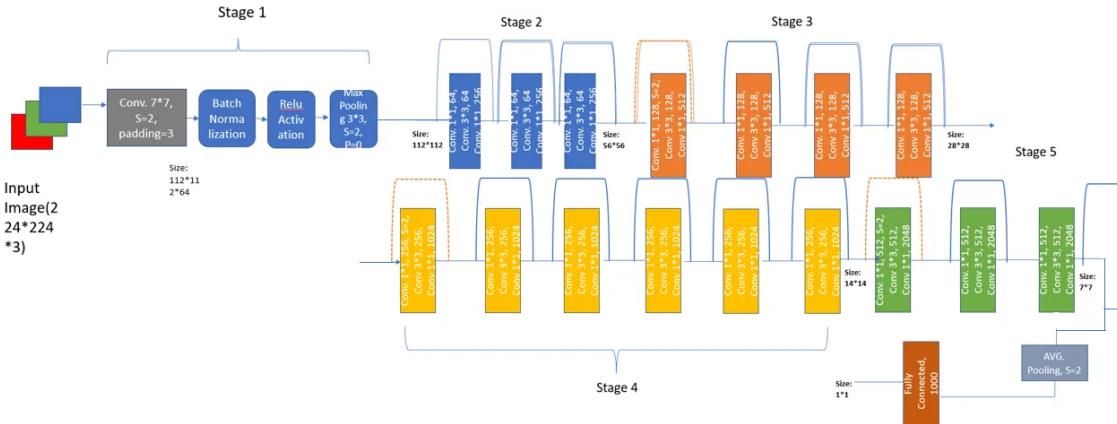
Batch Normalization: Thường được áp dụng sau mỗi lớp tích chập để chuẩn hóa các đầu ra, giúp tăng tốc độ huấn luyện và ổn định mạng.

ReLU: Sử dụng hàm kích hoạt ReLU sau mỗi lớp tích chập.

Shortcut Connection: Các kết nối tắt cho phép tín hiệu đi tắt qua một số lớp, giúp giảm thiểu vấn đề gradient vanishing.

Cách hoạt động của Residual Block

427
428
429
430
431
432
433
434
435



Hình 22: Kiến trúc của ResNet50

Đầu vào x được truyền qua một số lớp tích chập. Đầu ra sau khi qua các lớp tích chập được cộng với đầu vào x thông qua một kết nối tắt. Kết quả của phép cộng này được truyền qua hàm kích hoạt ReLU. Minh họa trong Hình 20.

4.2.3 Minh họa cách sử dụng ResNet50 và cấu hình chi tiết

Khi sử dụng ResNet, các trọng số của mô hình có thể được thiết lập dựa trên việc học từ tập dữ liệu ImageNet. Điều này có nghĩa là các lớp tích chập trong mô hình đã học được cách nhận dạng các đặc trưng chung từ hình ảnh. Chi tiết tại Hình 23 và Bảng 2 (Intelligence, 2019a).

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
def create_resnet50_model():
    # Tải mô hình ResNet-50 đã được huấn luyện trước trên tập dữ liệu ImageNet
    base_model = ResNet50(weights='imagenet')
    # Tạo mô hình mới với đầu vào là đầu vào của ResNet-50 và đầu ra là lớp 'avg_pool'
    model = Model(inputs=base_model.input, outputs=base_model.get_layer('avg_pool').output)
    return model
def extract_features(img_path, model):
    # Mở ảnh và thay đổi kích thước thành 224x224
    img = Image.open(img_path)
    img = img.resize((224, 224))
    # Chuyển đổi ảnh sang RGB
    img = img.convert('RGB')
    # Chuyển đổi ảnh thành mảng numpy
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    # Trích xuất đặc trưng
    features = model.predict(x)
    # Chuẩn hóa đặc trưng
    features = features / np.linalg.norm(features)
    return features
```

Hình 23: Cách sử dụng ResNet50

4.3 Xception

Xception (Extreme Inception) là một biến thể của kiến trúc Inception, được giới thiệu bởi François Chollet. Xception mở rộng ý tưởng của Inception bằng cách thay thế các khôi In-

ception bằng các khôi tích chập tách biệt (separable convolution), giúp tăng hiệu suất và độ chính xác của mô hình. Xception đặc biệt hiệu quả trong việc xử lý các tác vụ thị giác máy tính phức tạp (Chollet, 2016).

4.3.1 Cấu trúc của Xception

Xception được cấu trúc từ nhiều khôi chính, bao gồm các khôi tích chập tách biệt và các lớp chuẩn hóa. Cấu trúc này giúp tăng cường khả năng trích xuất đặc trưng và giảm thiểu số lượng tham số (Chollet, 2016).

Thành phần chính của Xception như sau:

- **Lớp đầu vào**

Kích thước đầu vào: W_xH_xD (rộng x cao x số kênh, ví dụ: 299x299x3 cho ảnh màu RGB).

- **Lớp tích chập ban đầu**

Lớp tích chập tiêu chuẩn với số lượng filters và kích thước kernel lớn, thường là 32 filters với kernel 3x3.

- **Các khôi Entry Flow**

Gồm các khôi tích chập tách biệt với các kết nối tắt (shortcut connections).

Mỗi khôi Entry Flow bao gồm các lớp tích chập, Batch Normalization, và ReLU.

Các khôi Middle Flow gồm 8 khôi tích chập tách biệt, mỗi khôi có các lớp tích chập và ReLU.

Stage	Number of Residual Blocks	Number of Convolutional Layers	Number of Parameters	Output Shape
Input	0	0	0	(224, 224, 3)
Conv1 (7x7, 64, stride 2)	0	1	9,472	(112, 112, 64)
MaxPooling (3x3, stride 2)	0	0	0	(56, 56, 64)
Conv2_x	3	9	~75K/block	(56, 56, 256)
Conv3_x	4	12	~300K/block	(28, 28, 512)
Conv4_x	6	18	~1.2M/block	(14, 14, 1024)
Conv5_x	3	9	~6M/block	(7, 7, 2048)
Global Average Pooling	0	0	0	(1, 1, 2048)
Fully Connected (Softmax)	0	0	2,049,000	(1000)

Bảng 2: Cấu hình chi tiết của ResNet50

• Các khối Exit Flow

Tương tự như Entry Flow, nhưng với số lượng filters và kích thước kernel tăng lên.

Kết thúc bằng một lớp Global Average Pooling và Fully-Connected với hàm kích hoạt softmax.

4.3.2 Minh họa cách sử dụng Xception và cấu hình chi tiết

Khi sử dụng Xception, các trọng số của mô hình có thể được thiết lập dựa trên việc học từ tập dữ liệu ImageNet. Điều này có nghĩa là các lớp convolutional trong mô hình đã học được cách nhận dạng các đặc trưng chung từ hình ảnh. Chi tiết tại Hình 25 và Bảng 3 (Intelligence, 2020).

5 Độ tương đồng và Faiss Indexing

5.1 Độ tương đồng

Khoảng cách Euclidean được dùng để tính khoảng cách giữa các query features và các features của mọi images được trích xuất.

Sau đó, nhóm sẽ thực hiện lấy ra top 10 hình

ảnh có khoảng cách Euclidean nhỏ nhất.

$$d(A, B) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2} \quad (1)$$

5.2 Faiss Indexing

Sử dụng thư viện Faiss để tối ưu hóa việc tìm kiếm tương tự trong không gian đa chiều. Kiến trúc của Faiss như sau:

5.2.1 Indexer

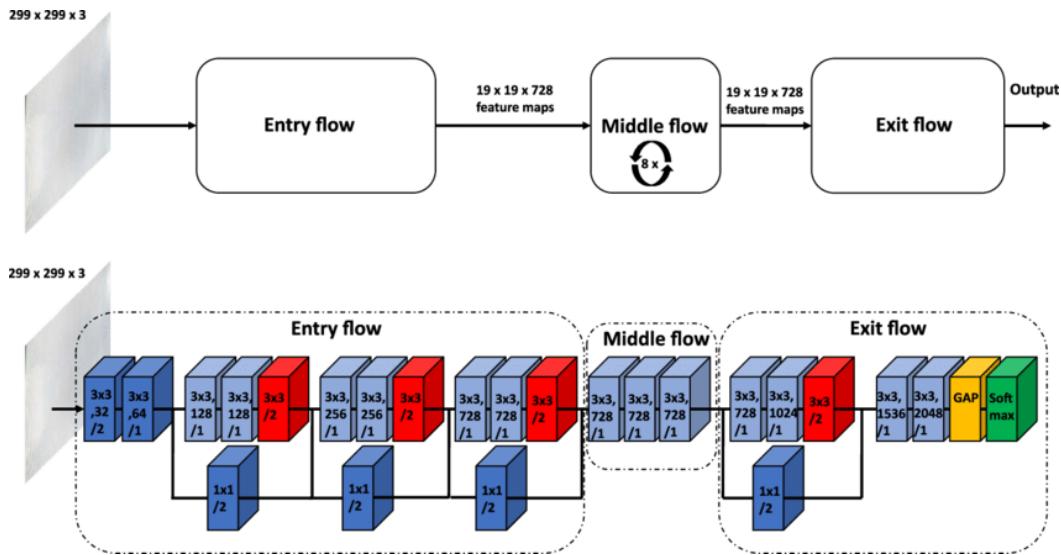
- Là cấu trúc dữ liệu để lưu trữ dữ liệu đa chiều.
- Các loại indexer phổ biến gồm IVF, HNSW, và PQ.

5.2.2 Searcher

- Là thuật toán dùng để tìm kiếm vector tương tự trong một indexer.
- Các loại searcher phổ biến gồm Brute-Force Searcher, K-Nearest Neighbors Searcher, và Approximate Nearest Neighbors Searcher.

Layer Type	Output Shape	Param #	Details
Input	(None, 299, 299, 3)	0	Input layer for the model
Convolution	(None, 149, 149, 32)	864	3x3, stride 2
Convolution	(None, 147, 147, 64)	18432	3x3
MaxPooling	(None, 73, 73, 64)	0	3x3, stride 2
SeparableConv2D	(None, 73, 73, 128)	8768	Depthwise + pointwise (1x1), followed by BN and ReLU
SeparableConv2D	(None, 73, 73, 128)	17536	Depthwise + pointwise (1x1), followed by BN and ReLU
MaxPooling	(None, 37, 37, 128)	0	3x3, stride 2
SeparableConv2D	(None, 37, 37, 256)	33920	Depthwise + pointwise (1x1), followed by BN and ReLU
SeparableConv2D	(None, 37, 37, 256)	67840	Depthwise + pointwise (1x1), followed by BN and ReLU
MaxPooling	(None, 19, 19, 256)	0	3x3, stride 2
SeparableConv2D	(None, 19, 19, 728)	188032	Depthwise + pointwise (1x1), followed by BN and ReLU
SeparableConv2D	(None, 19, 19, 728)	536536	Depthwise + pointwise (1x1), followed by BN and ReLU
MaxPooling	(None, 10, 10, 728)	0	3x3, stride 2
SeparableConv2D (12x)	(None, 10, 10, 728)	23761504	Depthwise + pointwise (1x1), followed by BN and ReLU (repeated 12 times)
SeparableConv2D	(None, 10, 10, 1024)	749952	Depthwise + pointwise (1x1), followed by BN and ReLU
SeparableConv2D	(None, 10, 10, 1024)	1534976	Depthwise + pointwise (1x1), followed by BN and ReLU
MaxPooling	(None, 5, 5, 1024)	0	3x3, stride 2
SeparableConv2D	(None, 5, 5, 1536)	1582080	Depthwise + pointwise (1x1), followed by BN and ReLU
SeparableConv2D	(None, 5, 5, 2048)	3159552	Depthwise + pointwise (1x1), followed by BN and ReLU
GlobalAverage Pooling2D	(None, 2048)	0	
Dense	(None, 1000)	2049000	Softmax activation for classification

Bảng 3: Cấu hình chi tiết của Xception



Hình 24: Kiến trúc Xception

```

from tensorflow.keras.applications import Xception
from tensorflow.keras.models import Model
def create_xception_model():
    # Tải mô hình Xception đã được huấn luyện trước trên tập dữ liệu ImageNet
    base_model = Xception(weights='imagenet')
    # Tạo mô hình mới với đầu vào là đầu vào của Xception và đầu ra là lớp 'avg_pool'
    model = Model(inputs=base_model.input, outputs=base_model.get_layer('avg_pool').output)
    return model

def extract_features(img_path, model):
    # Mở ảnh và thay đổi kích thước thành 224x224
    img = Image.open(img_path)
    img = img.resize((224, 224))
    # Chuyển đổi ảnh sang RGB
    img = img.convert('RGB')
    # Chuyển đổi ảnh thành mảng numpy
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    # Trích xuất đặc trưng
    features = model.predict(x)
    # Chuẩn hóa đặc trưng
    features = features / np.linalg.norm(features)
    return features

```

Hình 25: Cách sử dụng Xception

5.2.3 Metric

- Là hàm dùng để đo lường độ tương đồng giữa hai vector.
- Các loại metric phổ biến gồm L2 distance, Cosine similarity, và Jaccard similarity.

Lưu trữ mỗi vector trong tập dữ liệu dưới dạng một điểm trong không gian. Khi thực hiện tìm kiếm, Faiss sẽ tính toán khoảng cách Euclidean (L2) giữa vector truy vấn và tất cả các vector trong indexer sử dụng BruteForce Searcher. Vector có khoảng cách Euclidean nhỏ nhất với vector truy vấn được coi là vector tương tự nhất. Minh họa Faiss Indexing trong Hình 26.

```

import faiss
# Xây dựng một chỉ số FlatL2
index = faiss.IndexFlatL2(feature_dim)

# Thêm các vector đặc trưng vào chỉ số
index.add(features)

# Thực hiện tìm kiếm gần nhất cho một vector đặc trưng mới
# Ví dụ: Tìm 5 hình ảnh gần nhất với vector đặc trưng này
k = 5
distances, indices = index.search(query_feature, k)

# Kết quả sẽ là các chỉ mục của các hình ảnh gần nhất và khoảng cách tương ứng
print("Indices of nearest images:", indices)
print("Distances:", distances)

```

Hình 26: Faiss Indexing

6 Thực nghiệm

6.1 Độ đo

Trong bài toán này, nhóm sử dụng mean Average Precision (mAP), một độ đo phổ biến được sử dụng để đánh giá hiệu suất của hệ thống Content-Based Image Retrieval (CBIR). mAP đánh giá hiệu suất của hệ thống trong việc truy xuất hình ảnh có liên quan từ cơ sở dữ liệu, dựa trên một truy vấn của người dùng.

Công thức tính độ chính xác trung bình (Average Precision):

$$AP = \frac{1}{C} \sum_{k=1}^n (\text{Precision at } k \times rel(k)) \quad (2)$$

Ở đây, C là tổng số tài liệu liên quan và $rel(k)$ là một hàm chỉ số bằng 1 nếu mục ở hạng k là một tài liệu có liên quan, bằng 0 nếu không phải. Lưu ý rằng trung bình được tính trên các tài liệu có liên quan trong top- k tài liệu được trả về, và các tài liệu có liên quan

Mô hình	MAP Score
ResNet50	0.711
VGG 16	0.713
Xception	0.724

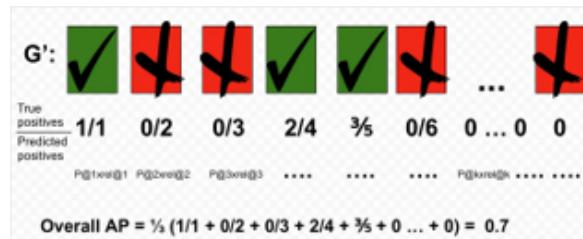
Bảng 4: Kết quả thực nghiệm trên ba mô hình

mà không được trả về sẽ có điểm độ chính xác bằng 0. Minh họa trong Hình 27.

Công thức tính trung bình độ chính xác trung bình (mean Average Precision):

$$mAP = \frac{1}{Q} \sum_{q=1}^Q AP_q \quad (3)$$

Trong đó, Q là số lượng truy vấn. Điểm mAP càng cao, hiệu suất của hệ thống CBIR càng tốt. mAP là một độ đo phổ biến được sử dụng để đánh giá hệ thống CBIR, vì nó xem xét tính liên quan của mỗi hình ảnh được truy xuất và cung cấp một đo lường toàn diện về hiệu suất của hệ thống qua nhiều truy vấn.



Hình 27: Minh họa độ đo AP

Cách thực hiện tính toán độ đo *mAP* được thể hiện trong Hình 28.

6.2 Kết quả thực nghiệm

Dựa vào Bảng 4, mô hình Xception thể hiện hiệu suất ấn tượng và ổn định nhất với mAP Score là 0.724. Điều này cho thấy rằng Xception đang làm việc tốt nhất trong việc xếp hạng các kết quả truy vấn đúng.

ResNet50 đạt điểm *mAP* là 0.713, chỉ thấp hơn một chút so với Xception. Điều này cho thấy ResNet50 cũng hoạt động tốt và gần tương đương với Xception.

Với điểm *mAP* là 0.711, VGG16 có hiệu suất thấp nhất trong ba mô hình.

```

1 import faiss
2 def compute_ap(ranked_labels, true_label):
3     """
4     Tính toán Average Precision cho một mẫu
5     """
6     num_relevant = 0
7     precision_sum = 0
8     for i, label in enumerate(ranked_labels):
9         if label == true_label:
10             num_relevant += 1
11             precision_sum += num_relevant / (i + 1)
12     if num_relevant == 0:
13         return 0
14     return precision_sum / num_relevant
15 def mAP(X_train, y_train, X_test, y_test, k=10):
16     """
17     Tính toán mean Average Precision (mAP) sử dụng FAISS
18     """
19     X_train=X_train.reshape(X_train.shape[0],-1)
20     X_test=X_test.reshape(X_test.shape[0],-1)
21     # Khởi tạo FAISS index
22     d = X_train.shape[1]
23     index = faiss.IndexFlatL2(d)
24     # Thêm dữ liệu huấn luyện vào chỉ mục
25     index.add(X_train)
26     # Tìm k láng giềng gần nhất cho các mẫu trong X_test
27     D, I = index.search(X_test, k)
28     ap_list = []
29     for i in range(len(X_test)):
30         # Lấy nhãn của k láng giềng gần nhất
31         ranked_labels = y_train[I[i]]
32         true_label = y_test[i]
33
34         # Tính toán Average Precision cho mẫu hiện tại
35         ap = compute_ap(ranked_labels, true_label)
36         ap_list.append(ap)
37
38     # Tính toán mean Average Precision
39     mAP_value = np.mean(ap_list)
40     return mAP_value

```

Hình 28: Cách tính độ đo mAP

Một minh họa cho kết quả truy vấn thực nghiệm của ba mô hình được thể hiện trong Hình 29 và 30.

7 Nhân xét

7.1 VGG16

7.1.1 Ưu điểm

Kiến trúc đơn giản và nhất quán

VGG16 sử dụng các lớp tích chập với kích thước bộ lọc cố định là 3×3 và các lớp pooling 2×2 , điều này tạo ra một kiến trúc nhất quán và dễ hiểu. Mỗi lớp tích chập được sau bởi một lớp ReLU, giúp làm tăng tính phi tuyến của mô hình.

Hiệu suất cao

VGG16 đã đạt được kết quả xuất sắc trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2014, đặc biệt là về độ chính xác trong nhận dạng hình ảnh.

Khả năng tổng quát hóa tốt

Mô hình VGG16 đã chứng tỏ khả năng tổng quát hóa tốt trên nhiều tác vụ khác nhau và có



Hình 29: Hình ảnh đầu vào (query image)

thể được sử dụng như một mô hình tiền huấn luyện (pre-trained model) cho nhiều ứng dụng khác nhau trong thị giác máy tính.

Dễ dàng áp dụng Transfer Learning

VGG16 là một trong những mô hình phổ biến nhất được sử dụng cho Transfer Learning. Bạn có thể tải trọng số tiền huấn luyện trên ImageNet và tinh chỉnh (fine-tune) mô hình cho các tác vụ cụ thể.

7.1.2 Nhược điểm

Kích thước mô hình lớn

VGG16 có khoảng 138 triệu tham số, làm cho nó trở thành một mô hình rất lớn và nặng nề. Điều này dẫn đến nhu cầu lớn về bộ nhớ và thời gian huấn luyện dài hơn so với các mô hình nhẹ hơn.

Tốc độ chậm

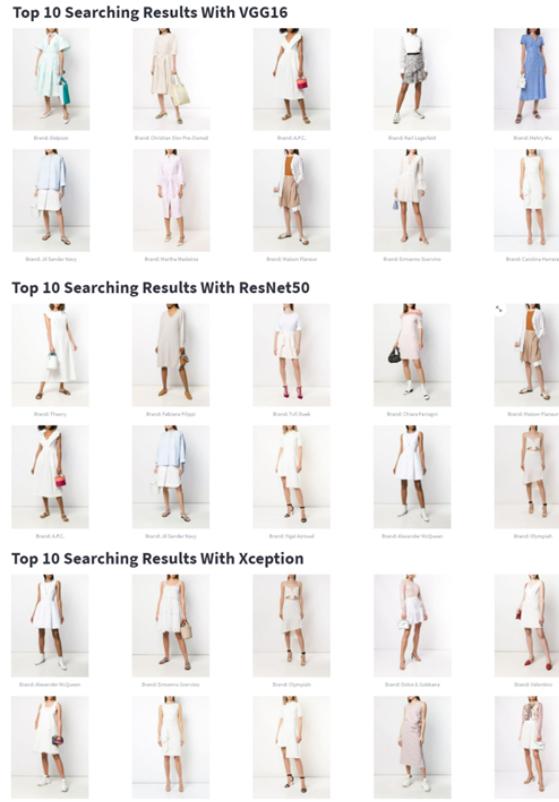
Do số lượng tham số lớn và độ sâu của mạng, VGG16 có tốc độ tính toán chậm hơn và không hiệu quả về mặt tính toán so với các mô hình hiện đại hơn như ResNet hoặc EfficientNet.

Khả năng mở rộng hạn chế

Kiến trúc VGG16 không linh hoạt trong việc mở rộng hoặc thay đổi các lớp mà không làm tăng đáng kể số lượng tham số và độ phức tạp tính toán.

Không hiệu quả về mặt tài nguyên

VGG16 yêu cầu tài nguyên phần cứng mạnh mẽ để huấn luyện và triển khai, điều này có thể là rào cản đối với các ứng dụng thực tế trên thiết bị có hạn chế về tài nguyên.



Hình 30: Kết quả truy vấn thực nghiệm của Hình 29

7.1.3 Nhận xét chung

Mặc dù có một số nhược điểm liên quan đến kích thước và tốc độ, ưu điểm về hiệu suất và khả năng tổng quát hóa đã làm cho VGG16 trở thành một công cụ quan trọng cho nhiều nhà nghiên cứu và ứng dụng thực tế trong nhận dạng hình ảnh và các tác vụ liên quan đến thị giác máy tính.

7.2 ResNet50

7.2.1 Ưu điểm

Hiệu suất tốt trên các bộ dữ liệu lớn

ResNet50 được đào tạo trên tập dữ liệu ImageNet lớn, nên có khả năng nhận diện hình ảnh và phân loại các lớp vật thể với hiệu suất tốt.

Khả năng học sâu

Kiến trúc ResNet với các khối residual cho phép mạng học sâu hơn mà không gặp vấn đề của gradient vanishing.

Dễ huấn luyện

Các khối residual giúp giảm thiểu vấn đề gradient vanishing và học được hiệu quả hơn. Điều này làm cho việc huấn luyện ResNet-50 trở nên dễ dàng hơn so với các mạng sâu khác.

654	<i>Tính tổng quát hóa cao</i>	700
655	ResNet50 có khả năng tổng quát hóa tốt, có thể áp dụng cho nhiều tác vụ trong thị giác máy tính, bao gồm phân loại hình ảnh, nhận dạng vật thể, phát hiện vật thể, và nhiều ứng dụng khác.	701
656	<i>Sử dụng hiệu quả tài nguyên</i>	702
657	ResNet50 sử dụng ít tài nguyên hơn so với một số mạng nơ-ron sâu khác, nhưng vẫn đạt được hiệu suất cao.	703
658		704
659		705
660		706
661		707
662		708
663		709
664	7.2.2 Nhược điểm	710
665	<i>Số lượng tham số lớn</i>	711
666	Mặc dù ResNet50 có hiệu suất tốt, nhưng số lượng tham số của nó vẫn khá lớn, đặc biệt khi so sánh với các mô hình nhỏ hơn như VGG16.	712
667		713
668		714
669	<i>Tính tài nguyên yêu cầu cao</i>	715
670	Trong một số ứng dụng có giới hạn về tài nguyên tính toán, ResNet50 có thể yêu cầu nhiều tài nguyên tính toán hơn để huấn luyện và triển khai.	716
671		717
672		718
673		719
674	<i>Độ phức tạp cao</i>	720
675	Do số lượng lớp và khối residual, ResNet50 có độ phức tạp cao, có thể gây khó khăn trong việc triển khai trên các thiết bị có tài nguyên hạn chế.	721
676		722
677		723
678		724
679	<i>Đòi hỏi dữ liệu lớn</i>	725
680	ResNet50 thường cần một lượng lớn dữ liệu huấn luyện để học các trọng số hiệu quả và tránh overfitting.	726
681		727
682		728
683	7.2.3 Nhận xét chung	729
684	Sử dụng ResNet với trọng số được huấn luyện trước giúp tận dụng được kiến thức từ một mô hình đã được huấn luyện kỹ lưỡng, giúp tăng tốc quá trình phát triển mô hình và cải thiện hiệu quả tổng thể. ResNet đã chứng minh được hiệu quả vượt trội trong nhiều bài toán về nhận dạng và phân loại hình ảnh.	730
685		731
686		732
687		733
688		734
689		735
690		736
691	7.3 Xception	737
692	7.3.1 Ưu điểm	738
693	<i>Hiệu suất cao hơn</i>	739
694	Sử dụng các khối tích chập tách biệt (depth- wise separable convolution) giúp tăng hiệu suất và độ chính xác của mô hình. Các khối này tách việc học đặc trưng không gian và kênh, cho phép mô hình học được các đặc trưng tinh vi hơn.	740
695		741
696		742
697		743
698		744
699		745
	<i>Giảm số lượng tham số</i>	746
	So với các mô hình có số lớp tương tự, Xcep- tion giảm đáng kể số lượng tham số, giúp giảm chi phí tính toán và bộ nhớ cần thiết.	
	<i>Tăng tốc độ huấn luyện và suy luận</i>	
	Các khối tích chập tách biệt không chỉ giảm số lượng tham số mà còn tăng tốc độ tính toán, do đó, thời gian huấn luyện và suy luận nhanh hơn.	
	<i>Tính linh hoạt</i>	
	Kiến trúc Xception có thể dễ dàng mở rộng và điều chỉnh cho các bài toán khác nhau. Điều này giúp mô hình dễ dàng áp dụng vào các tác vụ thị giác máy tính khác nhau, từ nhận dạng hình ảnh đến phân loại.	
	<i>Khả năng tổng quát hóa tốt</i>	
	Nhờ cấu trúc đặc biệt, Xception có khả năng học các đặc trưng phong phú và tổng quát hóa tốt hơn, giúp cải thiện hiệu suất trên các tập dữ liệu chưa từng thấy trong quá trình huấn luyện.	
	<i>Giảm hiện tượng vanishing gradient</i>	
	Các kết nối tắt (shortcut connections) giúp giảm hiện tượng vanishing gradient, giữ cho gradient duy trì ổn định trong suốt quá trình huấn luyện, đặc biệt khi mạng trở nên sâu hơn.	
	<i>Tối ưu hóa sử dụng tài nguyên phần cứng</i>	
	Xception được thiết kế để tối ưu hóa việc sử dụng các tài nguyên phần cứng hiện đại, như GPU, giúp mô hình chạy nhanh và hiệu quả hơn trên các thiết bị tính toán mạnh.	
	7.3.2 Nhược điểm	
	<i>Độ phức tạp tính toán cao</i>	
	Mặc dù các tích chập tách biệt theo chiều sâu làm giảm số lượng tham số, việc tính toán các tích chập này có thể phức tạp và đòi hỏi tài nguyên tính toán mạnh mẽ hơn, đặc biệt khi xử lý các lớp có độ sâu lớn.	
	<i>Yêu cầu về bộ nhớ cao</i>	
	Xception, với số lượng lớn các lớp tích chập, đòi hỏi một lượng bộ nhớ đáng kể. Điều này có thể là một rào cản đối với các thiết bị có hạn chế về tài nguyên, chẳng hạn như thiết bị di động hoặc các hệ thống nhúng.	
	<i>Khả năng không hiệu quả với dữ liệu nhỏ</i>	
	Kiến trúc Xception được thiết kế để làm việc với các tập dữ liệu lớn như ImageNet. Khi áp dụng cho các tập dữ liệu nhỏ hơn, mô hình có	

747 thể dễ bị overfitting do số lượng tham số lớn.
748

Độ phức tạp trong thiết kế và điều chỉnh

749 So với các mô hình đơn giản hơn như VGG
750 hay ResNet, Xception có cấu trúc phức tạp hơn,
751 đòi hỏi hiểu biết sâu hơn về cách tối ưu hóa và
752 điều chỉnh các siêu tham số (hyperparameters)
753 để đạt được hiệu suất tốt nhất.

Thời gian huấn luyện lâu hơn

755 Với số lượng lớn các lớp và độ phức tạp của
756 các phép tính toán, thời gian huấn luyện của
757 Xception có thể lâu hơn so với các mô hình
758 đơn giản hơn. Điều này có thể làm tăng chi phí
759 và thời gian trong quá trình phát triển mô hình.
760

Hiệu suất không tốt trong mọi trường hợp

761 Mặc dù Xception thường có hiệu suất cao
762 trên nhiều tập dữ liệu lớn, nó không luôn luôn
763 vượt trội hơn so với các kiến trúc khác trong
764 mọi trường hợp. Hiệu suất của mô hình có thể
765 phụ thuộc vào đặc điểm của dữ liệu và tác vụ
766 cụ thể.

7.3.3 Nhận xét chung

768 Xception là một kiến trúc mạng nơ-ron tiên
769 tiến, kết hợp giữa các khối tích chập tách biệt
770 và các kết nối tắt để tối ưu hóa việc học các
771 đặc trưng từ dữ liệu hình ảnh. Nhờ cấu trúc
772 đặc biệt này, Xception có thể học được các
773 đặc trưng phức tạp hơn với hiệu suất cao và
774 độ chính xác tốt hơn so với các kiến trúc trước
775 đây như Inception.

8 Tổng kết

8.1 Kết luận

778 Nhóm đã thành công xây dựng hệ thống tìm
779 kiếm hình ảnh tương tự với hình ảnh đầu vào
780 với hiệu suất cao. Mô hình CNN thể hiện sự
781 xuất sắc trong việc tìm kiếm hình ảnh tương tự
782 với hình ảnh đầu vào. Điều này là minh chứng
783 cho hiệu quả và hiệu suất cao của các mô hình
784 CNN trong ngữ cảnh của bài toán truy vấn
785 hình ảnh. Nhóm nghiên cứu đã thành công giải
786 quyết bài toán này.

787 Sử dụng các mô hình pre-trained như Xception
788 và ResNet50 mang lại hiệu suất cao và
789 giảm đáng kể thời gian và tài nguyên. Học
790 chuyển giao trong truy vấn hình ảnh thời trang
791 cải thiện khả năng trích xuất đặc trưng, tăng

Thành viên	Mức độ hoàn thành
Nguyễn Công Nguyên	100%
Hồ Đức Trưởng	100%
Lê Minh Nguyệt	90%

Bảng 5: Đánh giá mức độ hoàn thành

cường tổng quát hóa và giảm nguy cơ overfitting, làm mô hình trở nên ổn định và đáng tin cậy.

Euclidean Distance đơn giản và dễ hiểu, nó là phương pháp tốt để phản ánh độ tương đồng, đặc biệt là khi áp dụng vào không gian đặc trưng đã được trích xuất từ các mô hình pre-trained.

8.2 Hướng phát triển

Nhóm sẽ mở rộng và nâng cao khả năng học của hệ thống truy vấn hình ảnh thời trang bằng cách thử nghiệm và đánh giá hiệu suất của các mô hình pretrained khác nhau như EfficientNet, MobileNet, và các mô hình mới khác. Qua đó, nhóm mong muốn tìm ra mô hình phù hợp nhất cho bài toán cụ thể của họ, có thể cung cấp kết quả tốt hơn và cải thiện trải nghiệm người dùng trong quá trình tìm kiếm sản phẩm thời trang.

Tích hợp các tính năng tương tác người dùng có thể làm tăng trải nghiệm sử dụng. Các tính năng như tìm kiếm theo màu sắc, xu hướng thời trang mới, hoặc gợi ý sản phẩm có thể cải thiện sự tương tác và tìm kiếm của người dùng.

Nhóm sẽ nghiên cứu và so sánh các phương pháp đo lường độ tương đồng khác nhau như Cosine Similarity, Jaccard Similarity để xem liệu chúng có thể cung cấp kết quả tốt hơn Euclidean distance trong ngữ cảnh cụ thể của bài toán.

8.3 Mức độ hoàn thành công việc

Mức độ hoàn thành công việc của các thành viên trong nhóm được đánh giá tại Bảng 5.

References

- F. Chollet. 2016. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv.org*.

828 K. He, Zhang, X., S. Ren, and J. Sun. 2015. Deep
829 residual learning for image recognition. *arXiv.org*.

830 MLT Artificial Intelligence. 2019a. Mlt cnn archi-
831 -tectures: Resnet - implementation. https://www.youtube.com/watch?v=LogyPK6M_L0. Accessed: 2024-05-20.

834 MLT Artificial Intelligence. 2019b. Mlt cnn archi-
835 -tectures: Vgg - implementation. <https://www.youtube.com/watch?v=cmHhqvvDBqo>. Accessed: 2024-05-20.

838 MLT Artificial Intelligence. 2020. Mlt cnn architectures
839 - xception implementation. <https://www.youtube.com/watch?v=nMBCSroJ7bY>. Accessed: 2024-05-20.

842 N. Kundu. 2023. Exploring ReSNet50: An In-Depth
843 look at the model architecture and code implementa-
844 tion. *Medium*.

845 Chitwan Manchanda. 2022. Fashion Image Search En-
846 gine. *Medium*.

847 R. Pramoditha. 2022. Convolutional Neural Network
848 (CNN) architecture explained in plain English using
849 simple diagrams. *Medium*.

850 P. Ruiz. 2024. Understanding and visualizing ResNets.
851 *Towards Data Science*.

852 T. Sharma. 2023. Detailed explanation of Resnet CNN
853 model. *Medium*.

854 K. Simonyan and A. Zisserman. 2014. Very deep convo-
855 lutional networks for Large-Scale image recognition.
856 *arXiv.org*.

857 K. Smeda. 2022. Understand the architecture of CNN.
858 *Towards Data Science*.

859 LILING TAN. 2019. Farfetch listings. <https://www.kaggle.com/datasets/alvations/farfetch-listings>. Accessed: 2024-05-20.