ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN MÔN HỌC
THỊ GIÁC MÁY TÍNH NÂNG CAO
CS331.N12.KHCL

SINH VIÊN THỰC HIỆN: ĐÀO TRẦN ANH TUẤN – 20522107
TRẦN PHÚ VINH - 20522161
GIẢNG VIÊN HƯỚNG DẪN: MAI TIẾN DŨNG

TP. HỒ CHÍ MINH, 2/2023

Mục lục

1. Giới thiệu bài toán

1.1. Tính cấp thiết của đề tài

Bài toán "Ước tính độ tuổi qua khuôn mặt bằng mạng học sâu" là một nghiên cứu trong lĩnh vực thị giác máy tính. Nghiên cứu này nhằm mục đích sử dụng mạng học sâu để xác định độ tuổi của một người dựa trên hình ảnh khuôn mặt của họ.

Bài toán cung cấp cách xác định dễ dàng và chính xác một cách tự động độ tuổi của một người, điều này có thể được áp dụng trong nhiều lĩnh vực thực tiễn như: marketing, bảo mật an ninh, giám sát truy cập,…

1.2. Input và output của bài toán

Input: Một bức ảnh có chứa khuôn mặt.

Output: Độ tuổi của các khuôn mặt trong ảnh.

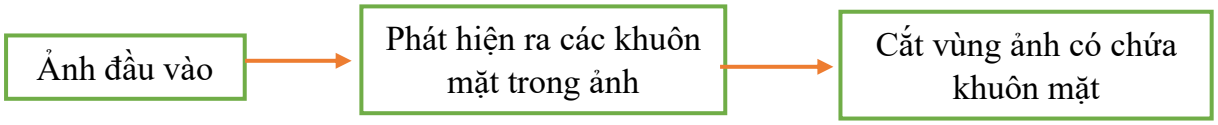
Hình 1: Minh họa Input và Output bài toán

2. PIPELINE

Nhóm đề xuất chia bài toán thành 2 giai đoạn: phát hiện khuôn mặt và ước tính độ tuổi khuôn mặt vừa phát hiện được.

2.1. Phát hiện khuôn mặt

Nhóm em đề xuất trình tự xử lí như sau



Hình 2: Các bước phát hiện khuôn mặt

Phát hiện khuôn mặt là một bài toán phát hiện vật thể (object detection) do đó nhóm sử dụng model YOLOv7[1] với tốc độ xử lí nhanh và độ chính xác cao trong các bài toán phát hiện vật thể.

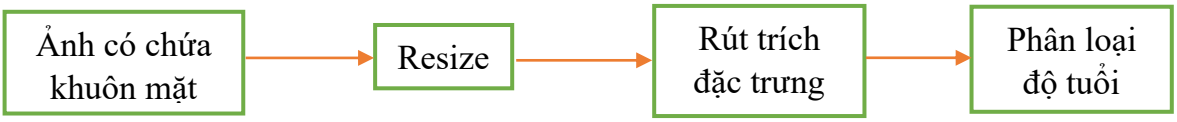
Nhóm sử dụng bộ dữ liệu WIDER FACE[2], tiến hành chia train, val, test theo tỷ lệ:

- Train (70%): 5076 ảnh
- Validation (20%): 1451 ảnh
- Test (10%): 725 ảnh

Các ảnh được resize về kích thước $640 \times 640 \times 3$

2.2. Ước tính độ tuổi

Nhóm đề xuất trình tự xử lí như sau:



Hình 3: Các bước ước tính độ tuổi

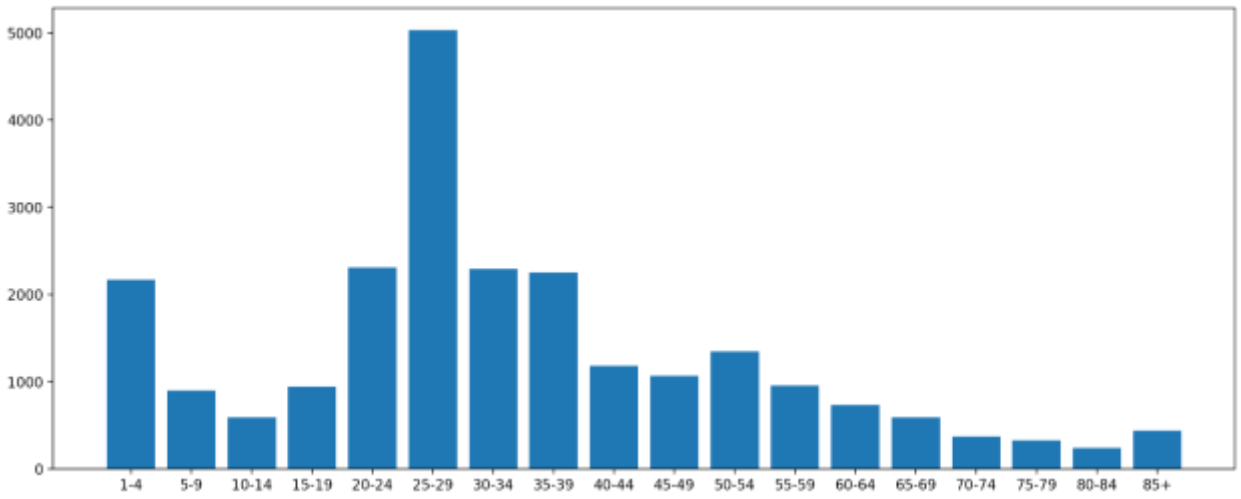
Vì chỉ ước tính độ tuổi nên nhóm dùng các mạng học sâu để trích xuất đặc trưng từ hình ảnh khuôn mặt, sau đó phân loại vào các độ tuổi tương ứng.

2.2.1. Bộ dữ liệu

Bộ dữ liệu mà nhóm sử dụng là: UTKFace[3], bộ dữ liệu gồm 23707 ảnh với độ tuổi từ 1 – 116, và nhiều sắc tộc.

Hình 4: Một số ảnh trong tập UTKFace

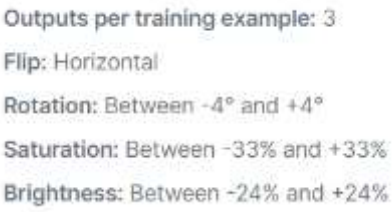
Nhóm tiến hành chia thành 18 độ tuổi: 1-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75-79, 80-84, 85+.



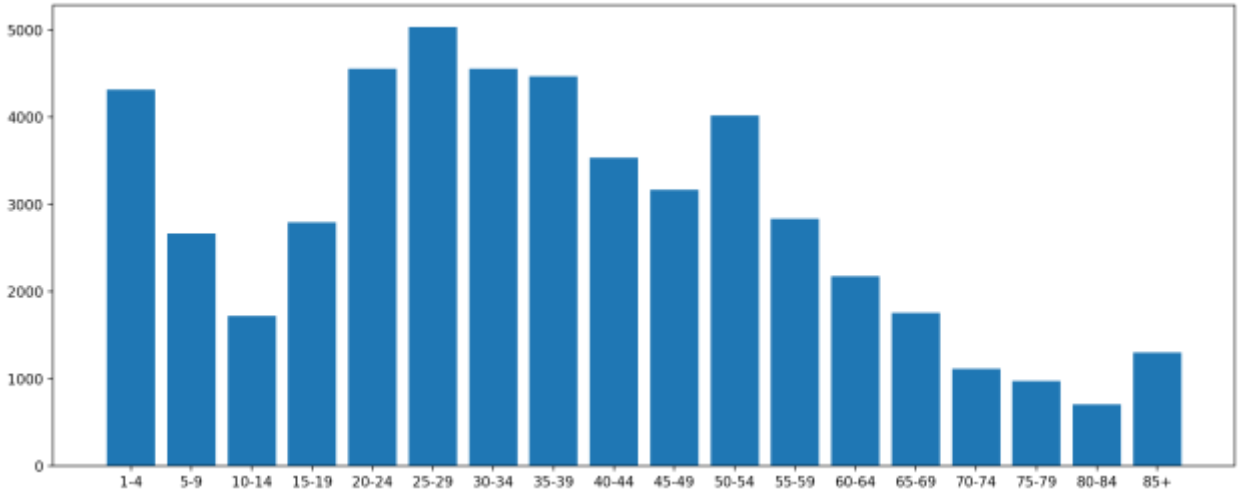
Hình 5: Phân bố độ tuổi trong tập dữ liệu

Vì độ tuổi liên tục biến đổi theo thời gian, do đó biểu đồ phân bố trên có thể phù hợp lúc này nhưng không phù hợp vào lúc khác, do đó nhóm tiến hành tăng cường dữ liệu ở một số độ tuổi có số lượng ít để bộ dữ liệu cân bằng hơn, giúp cho mô hình học sâu sau khi được huấn luyện có thể hoạt động tốt với nhiều lứa tuổi, điều này giúp mô hình hoạt động trong nhiều điều kiện khác nhau.

Nhóm sử dụng công cụ Roboflow để tiến hành tăng cường dữ liệu. Các phép biến đổi ảnh được sử dụng như sau:



Tuy nhiên, vì tài nguyên của Roboflow cung cấp có hạn, nên một nhóm chỉ có thể tăng cường tối đa 3 lần. Số lượng ảnh trong bộ dữ liệu là 51654 ảnh.



Hình 6: Phân bố dữ liệu sau khi được tăng cường

Sau đó nhóm tiến hành chia tập train, validation với tỉ lệ:

- Train (80%): 41324 ảnh.
- Validation (20%): 10330 ảnh.

Phân bố các độ tuổi trong 2 tập train và validation vẫn đảm bảo giống với bộ dữ liệu trước khi chia.

2.2.2. Model

Nhóm sử dụng kĩ thuật transfer learning[4], điều này giúp cho việc huấn luyện model trở nên hiệu quả hơn, tốn ít tài nguyên hơn. Các model pre-train được nhóm sử dụng là

EfficientNetV2B0, MobileNetV3Large, InceptionV3, VGG16, ResNet50. Các model đều được framework Keras[5] hỗ trợ sẵn.

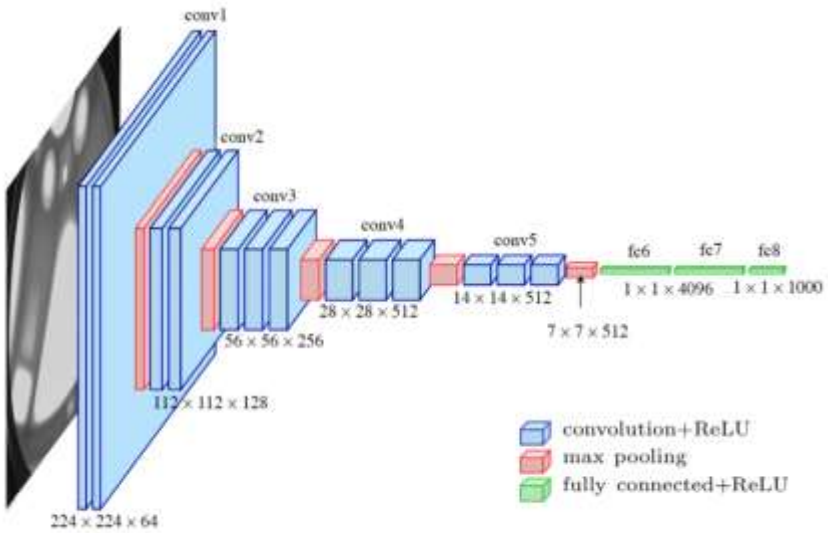
Model	Top-1 Accuracy	Size (MB)	Parameters (M)
MobileNetV3Large	75.6	21.6	5.4
EfficientNetV2B0	78.7	29	7.2
InceptionV3	77.9	92	23.9
ResNet50	74.9	98	25.6
VGG16	71.3	528	138.4

Bảng 1: Accuracy và thông số của các mô hình trên tập ImageNet

Nhóm thử nghiệm với các model có số parameters nhỏ (MobileNetV3Large, EfficientNetV2B0), vừa (InceptionV3, ResNet50) và lớn (VGG16) để có các model phù hợp với nhiều mục đích sử dụng. Các model đều nhận input ảnh đầu vào là (224,224,3).

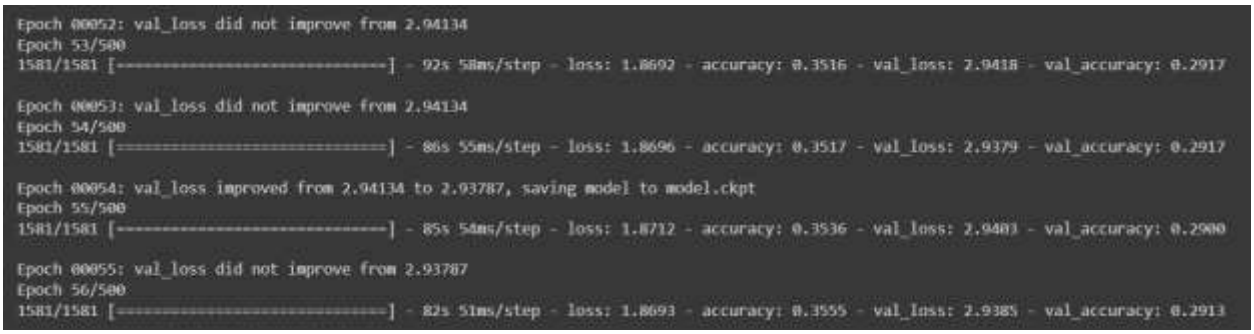
2.2.2.1. Training VGG16

VGG16[6] là một mô hình deep learning được thiết kế bởi Visual Geometry Group (VGG) của Đại học Oxford. Nó được sử dụng để phân loại hình ảnh và đã đạt được kết quả tốt trong các cuộc thi computer vision như ImageNet.



Hình 7: Kiến trúc của VGG16

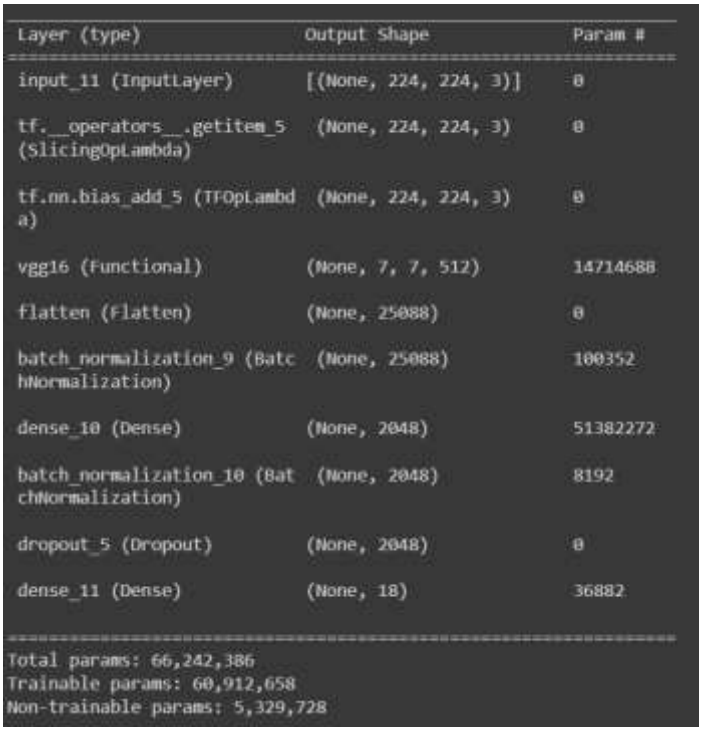
Đây là model đầu tiên nhóm thử nghiệm, ở những lần training đầu tiên, bộ dữ liệu chưa được tăng cường, model cho ra kết quả không tốt.



Có thể thấy rằng quá trình training gặp phải trường hợp underfitting, cho ra điểm accuracy trên cả tập train và val rất thấp. Do đó nhóm tiến hành tăng cường dữ liệu như bên trên.

Sau khi tiến hành tăng cường dữ liệu, số lượng mẫu giữa các lớp vẫn chưa thật sự cân bằng. Để đạt được mục tiêu nhóm đề ra, nhóm kết hợp thêm kĩ thuật cân bằng trọng số (tính toán bằng thư viện sklearn) để giúp model tập trung hơn vào các lớp có số mẫu thấp.

Cấu trúc model nhóm sử dụng training như sau:

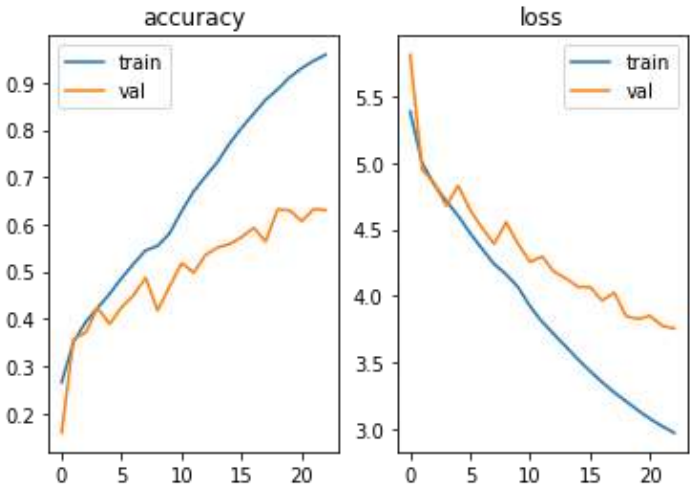


Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 224, 224, 3)]	0
tf.__operators__.getitem_5 (SlicingOpLambda)	(None, 224, 224, 3)	0
tf.nn.bias_add_5 (TFOpLambda)	(None, 224, 224, 3)	0
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
batch_normalization_9 (BatchNormalization)	(None, 25088)	100352
dense_10 (Dense)	(None, 2048)	51382272
batch_normalization_10 (BatchNormalization)	(None, 2048)	8192
dropout_5 (Dropout)	(None, 2048)	0
dense_11 (Dense)	(None, 18)	36882

Total params: 66,242,386
Trainable params: 60,912,658
Non-trainable params: 5,329,728

Hình 8: Cấu trúc VGG16 thử nghiệm

Lớp feature extraction của VGG16 được "freeze" tức là trong quá trình training không làm thay đổi parameters của lớp mà tiến hành cập nhật parameter cho các lớp phía sau lớp flatten (gọi là feature prediction). Các lớp fully connected (dense) được thêm với cấu trúc giống mạng VGG16 có sẵn trong Keras, thêm vào 2 lớp Dropout, BatchNormalization và dùng thêm Regularizer với các thông số được điều chỉnh phù hợp để ngăn chặn hiện tượng overfitting trong quá trình training. Tuy nhiên, các kĩ thuật trên có thể gây ra hiện tượng underfitting nếu chỉnh tham số không phù hợp, do đó, nhóm gặp nhiều trường hợp overfitting và underfitting khi tinh chỉnh tham số, khó có thể để tìm được tham số phù hợp. Trong số các trường hợp đó, nhóm thu được 1 model được cho là tốt nhất khi có điểm accuracy cao nhất trên tập train – 95% và tập validation – 63%.



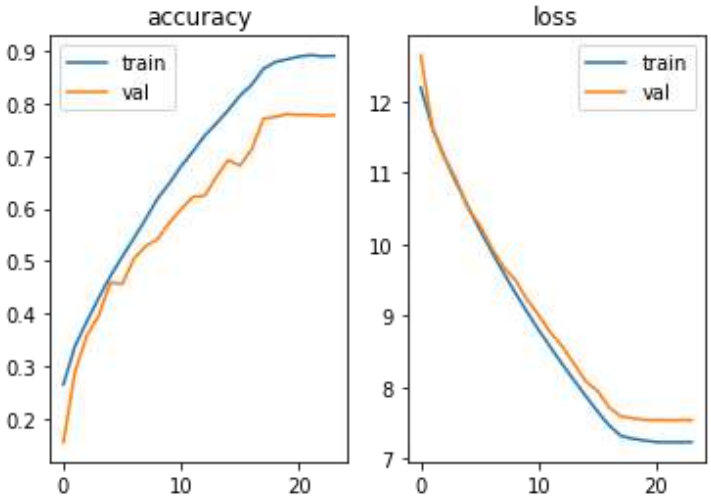
Hình 9: Accuracy và Loss trên tập train, validation

Tuy vẫn gặp hiện tượng overfitting, do đó, nhóm sử dụng thêm kĩ thuật fine-tuning, tức là thay vì "freeze" toàn bộ phần feature extraction, nhóm chỉ "freeze" các lớp đầu tiên và cho những lớp cuối cùng của phần feature extraction được cập nhật parameter trong quá trình training, và chỉnh lại các parameter cho hợp lí hơn. Kết quả nhóm thu được rất khả quan tuy vẫn còn bị overfitting nhẹ (accuracy là 89% trên tập train và 78% trên tập validation).

Layer (type)	Output Shape	Param #
input_18 (InputLayer)	[(None, 224, 224, 3)]	0
sequential (Sequential)	(None, 224, 224, 3)	0
tf.__operators__.getitem_8 (	(None, 224, 224, 3)	0
tf.nn.bias_add_8 (TFOpLambda	(None, 224, 224, 3)	0
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
batch_normalization_16 (Batc	(None, 25088)	100352
dense_24 (Dense)	(None, 4096)	102764544
dropout_10 (Dropout)	(None, 4096)	0
batch_normalization_17 (Batc	(None, 4096)	16384
dense_25 (Dense)	(None, 2048)	8390656
dropout_11 (Dropout)	(None, 2048)	0
dense_26 (Dense)	(None, 18)	36882

Total params: 126,023,506
Trainable params: 118,329,874
Non-trainable params: 7,693,632

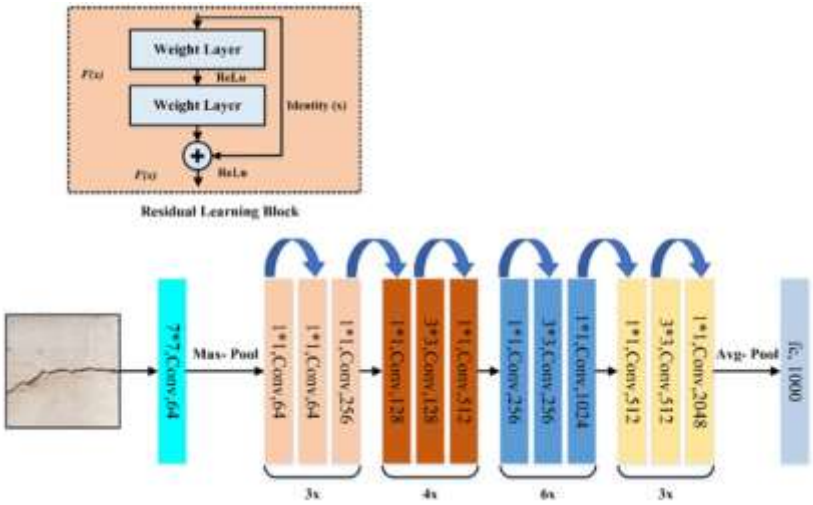
Hình 10: Cấu trúc VGG16 sau khi thực hiện fine-tuning



Hình 11: Accuracy và Loss trên tập train, validation

2.2.2.2. Training ResNet50

ResNet[7] (Residual Network) là một loại deep neural network trong lĩnh vực thị giác máy tính, sử dụng Residual block để tránh hiện tượng vanishing gradient thường gặp khi huấn luyện các mạng học sâu.



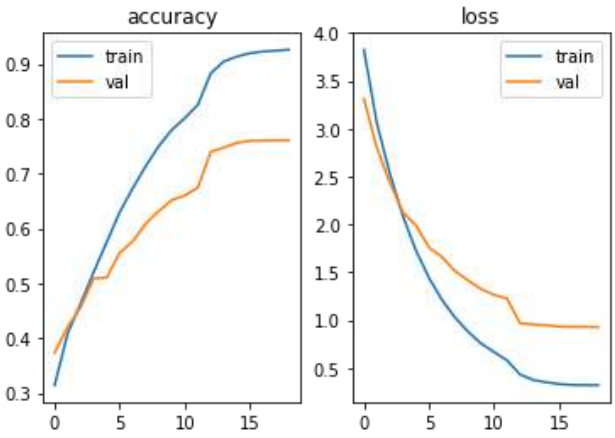
Hình 12: Kiến trúc ResNet50 với 50 layers

Với kinh nghiệm từ việc huấn luyện model VGG16, nhóm đã sử dụng các kĩ thuật tương tự và điều chỉnh parameter, kết quả thu được: accuracy là 92% trên tập train và 76% trên tập validation.

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 224, 224, 3)]	0
sequential (Sequential)	(None, 224, 224, 3)	0
tf.__operators__.getitem_2 (	(None, 224, 224, 3)	0
tf.nn.bias_add_2 (TFOpLambda	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
global_average_pooling2d_2 (	(None, 2048)	0
dropout_4 (Dropout)	(None, 2048)	0
dense_4 (Dense)	(None, 2048)	4196352
dropout_5 (Dropout)	(None, 2048)	0
dense_5 (Dense)	(None, 18)	36882

Total params: 27,820,946
Trainable params: 13,164,562
Non-trainable params: 14,656,384

Hình 13: Cấu trúc Resnet50 nhóm sử dụng



Hình 14: Accuracy và Loss trên tập train, validation

2.2.2.3. Training EfficientNetV2B0:

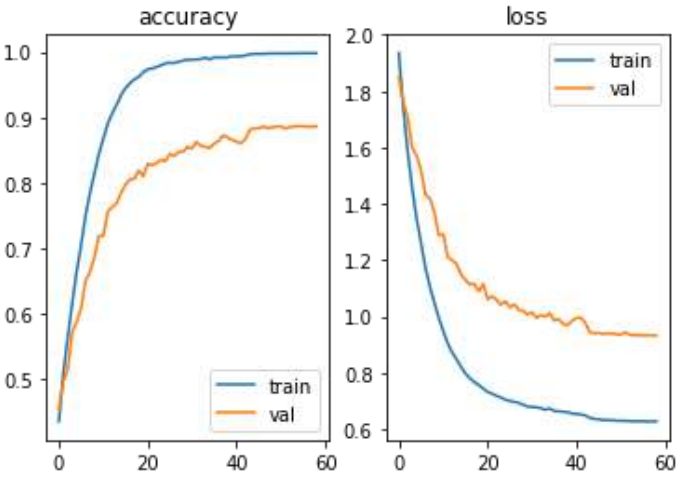
EfficientNet[8] là một loại deep neural network được sử dụng rộng rãi trong lĩnh vực thị giác máy tính. Nó đặc biệt ở chỗ có thể scale model theo chiều rộng của mạng, chiều sâu của mạng và độ phân giải của hình ảnh đưa vào các lớp CNN. Với EfficientNet-V2B0, các nhà phát triển đã tối ưu hóa các tham số và cấu trúc của mạng để cải thiện hiệu năng và chính xác trong phân loại và phát hiện đối tượng trong hình ảnh. EfficientNet-V2B0 cũng cung cấp một cách tiết kiệm bộ nhớ và tài nguyên hơn so với các kiểu mạng khác có cùng độ sâu.

Tương tự như khi train model ResNet50, nhóm sử dụng lớp BatchNormalization và Dropout với các tham số tham khảo từ trang “EfficientNet for PyTorch”, kết quả thu được rất tốt trên tập train – accuracy 98%, tuy nhiên bị overfitting khi accuracy trên tập validation là 89%.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
efficientnetv2-b0 (Functional)	(None, 7, 7, 1280)	5919312
avg_pool (GlobalAveragePooling2D)	(None, 1280)	0
batch_normalization (BatchNormalization)	(None, 1280)	5120
top_dropout (Dropout)	(None, 1280)	0
pred (Dense)	(None, 18)	23058

Total params: 5,947,490
Trainable params: 0
Non-trainable params: 5,947,490

Hình 15: Cấu trúc EfficientNet v2-B0 nhóm sử dụng



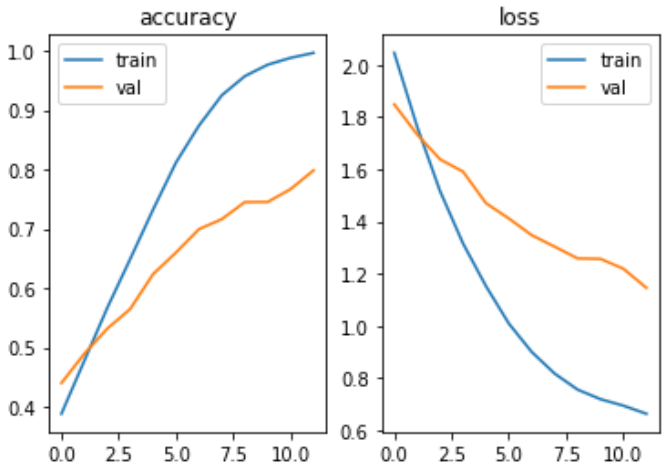
Hình 16: Accuracy và Loss trên tập train, validation

2.2.2.4. Training MobileNetV3Large

MobileNet[9] là một kiểu mạng neural (neural network) trong computer vision và deep learning. Nó được thiết kế để sử dụng trên thiết bị di động với các giới hạn tài nguyên tính toán như RAM và CPU. MobileNetV3-Large được thiết kế với mục đích cải thiện hiệu năng và chính xác so với các kiểu mạng trước đó của MobileNet. Nó sử dụng các kỹ thuật như nén số lượng lớp và cấu trúc mạng để tối ưu hóa bộ nhớ và tài nguyên.

```
Model: "MobileNetV3Large"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
MobilenetV3large (Functional (None, 1, 1, 1280)        4226432
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 1, 1280)        0
_________________________________________________________________
Logits (Conv2D)              (None, 1, 1, 18)          23058
_________________________________________________________________
flatten_6 (Flatten)          (None, 18)                0
_________________________________________________________________
dense (Dense)                (None, 18)                342
=================================================================
Total params: 4,249,832
Trainable params: 4,225,432
Non-trainable params: 24,400
```

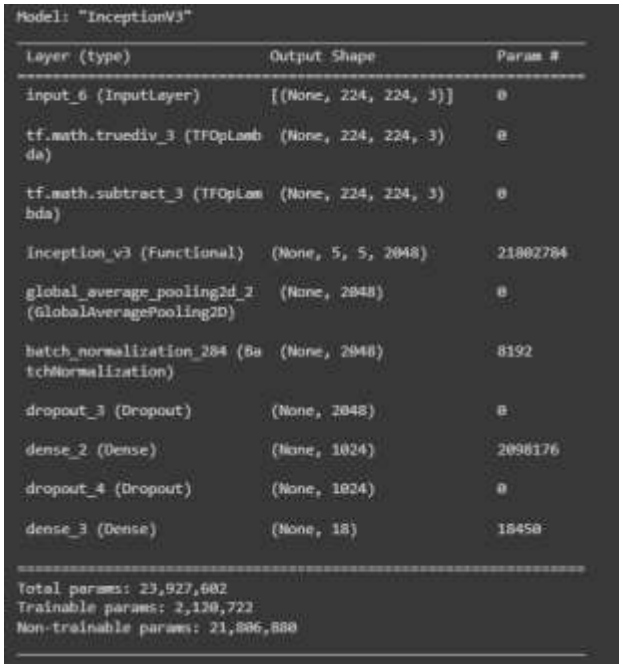
Hình 17: Cấu trúc MobileNetV3Large



Hình 18: Accuracy và Loss trên tập train, validation

2.2.2.5. Training InceptionV3

Inception v3[10] là một mô hình mạng học sâu dùng để phân loại hình ảnh đạt được accuracy hơn 78.1% trên tập ImageNet. Mô hình được giới thiệu bởi các nhà nghiên cứu Google vào năm 2015 và là một trong những mô hình học sâu nổi tiếng. Inception v3 sử dụng các lớp tích chập có kích thước kernel khác nhau song song với nhau để trích xuất đặc trưng ảnh ở nhiều mức thu phóng và góc nhìn. Mô hình là sự kết tinh từ nhiều ý tưởng khác nhau của nhiều nhà nghiên cứu qua nhiều năm, dựa trên bài báo góc "Rethinking the Inception Architecture for Computer Vision" của Szegedy và cộng sự.



```
Model: "InceptionV3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 224, 224, 3)]     0

tf.math.truediv_3 (TFOpLamb  (None, 224, 224, 3)       0
da)

tf.math.subtract_3 (TFOpLam  (None, 224, 224, 3)       0
bda)

inception_v3 (Functional)    (None, 5, 5, 2048)        21802784

global_average_pooling2d_2   (None, 2048)              0
(GlobalAveragePooling2D)

batch_normalization_284 (Ba  (None, 2048)              8192
tchNormalization)

dropout_3 (Dropout)          (None, 2048)              0

dense_2 (Dense)              (None, 1024)              2098176

dropout_4 (Dropout)          (None, 1024)              0

dense_3 (Dense)              (None, 18)                18450

=================================================================
Total params: 23,927,602
Trainable params: 2,130,722
Non-trainable params: 21,806,880
_________________________________________________________________
```

Hình 19: Cấu trúc Inception V3 nhóm sử dụng

```
Epoch 1/500
646/646 [==============================] - 159s 236ms/step - loss: 2.2931 - accuracy: 0.3525 - val_loss: 2.2905 - val_accuracy: 0.3556

Epoch 00001: val_accuracy improved from -inf to 0.35557, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 2/500
646/646 [==============================] - 150s 232ms/step - loss: 2.0240 - accuracy: 0.4302 - val_loss: 2.2470 - val_accuracy: 0.3725

Epoch 00002: val_accuracy improved from 0.35557 to 0.37253, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 3/500
646/646 [==============================] - 153s 235ms/step - loss: 1.8717 - accuracy: 0.4807 - val_loss: 2.0910 - val_accuracy: 0.4206

Epoch 00003: val_accuracy improved from 0.37253 to 0.42062, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 4/500
646/646 [==============================] - 152s 235ms/step - loss: 1.6971 - accuracy: 0.5416 - val_loss: 2.0527 - val_accuracy: 0.4295

Epoch 00004: val_accuracy improved from 0.42062 to 0.42953, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 5/500
646/646 [==============================] - 149s 228ms/step - loss: 1.5835 - accuracy: 0.6108 - val_loss: 2.3445 - val_accuracy: 0.3778

Epoch 00005: val_accuracy did not improve from 0.42953

Epoch 00005: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
Epoch 6/500
646/646 [==============================] - 147s 227ms/step - loss: 1.2488 - accuracy: 0.7311 - val_loss: 1.6256 - val_accuracy: 0.5909

Epoch 00006: val_accuracy improved from 0.42953 to 0.59090, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 7/500
646/646 [==============================] - 150s 231ms/step - loss: 1.1468 - accuracy: 0.7832 - val_loss: 1.6387 - val_accuracy: 0.6056

Epoch 00007: val_accuracy improved from 0.59090 to 0.60358, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 8/500
646/646 [==============================] - 151s 233ms/step - loss: 1.0796 - accuracy: 0.8197 - val_loss: 1.5833 - val_accuracy: 0.6238

Epoch 00008: val_accuracy improved from 0.60358 to 0.62197, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 9/500
646/646 [==============================] - 153s 236ms/step - loss: 1.0155 - accuracy: 0.8500 - val_loss: 1.5632 - val_accuracy: 0.6360

Epoch 00009: val_accuracy improved from 0.62197 to 0.63601, saving model to ./InceptionV3/Fine_tuning/model.ckpt
Epoch 10/500
646/646 [==============================] - 152s 235ms/step - loss: 0.9565 - accuracy: 0.8834 - val_loss: 1.5566 - val_accuracy: 0.6465

Epoch 00010: val_accuracy improved from 0.63601 to 0.64647, saving model to ./InceptionV3/Fine_tuning/model.ckpt

Epoch 00010: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-07.
Epoch 11/500
646/646 [==============================] - 148s 228ms/step - loss: 0.9098 - accuracy: 0.9158 - val_loss: 1.5272 - val_accuracy: 0.6614

Epoch 00011: val_accuracy improved from 0.64647 to 0.66137, saving model to ./InceptionV3/Fine_tuning/model.ckpt
```

Hình 20: Accuracy và Loss trên tập train, validation ở một số epoch đầu

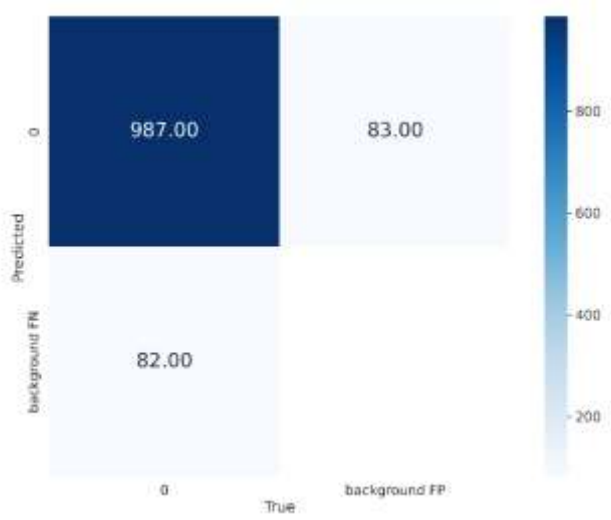
3. Kết quả

Để đánh giá kết quả của các mô hình, nhóm sử dụng các thang đo mAP cho phần phát hiện khuôn mặt và Accuracy cho phần ước tính độ tuổi khuôn mặt. Quá trình đánh giá được thực hiện trên môi trường Google Colab với CPU 2 core Intel(R) Xeon(R) @ 2.20GHz và GPU Tesla T4.

3.1. Phát hiện khuôn mặt

Đánh giá YOLOv7 trên 725 ảnh

	YOLOv7
mAP@0.5	0.94
mAP@0.5:0.95	0.63
Time (ms/image)	14.3

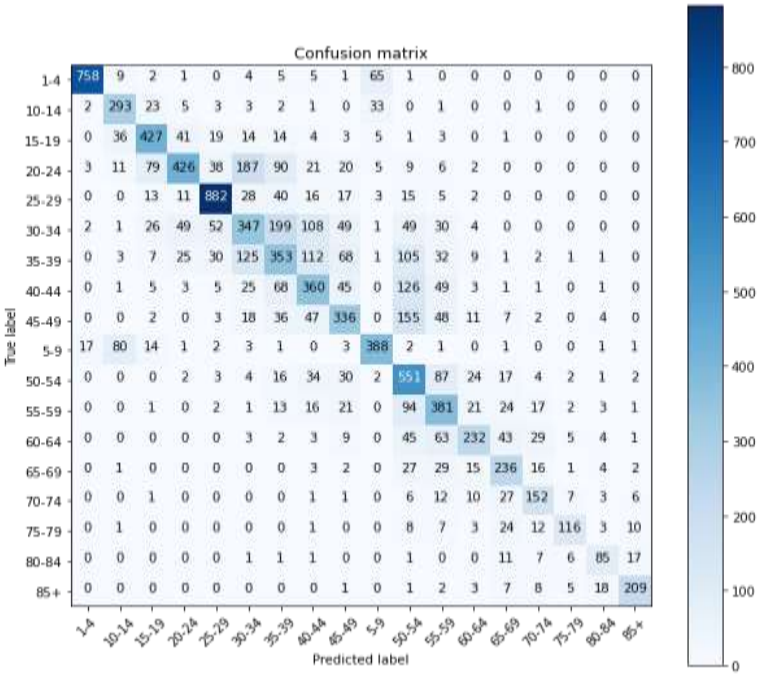


Hình 21: Confusion matrix của YOLOv7

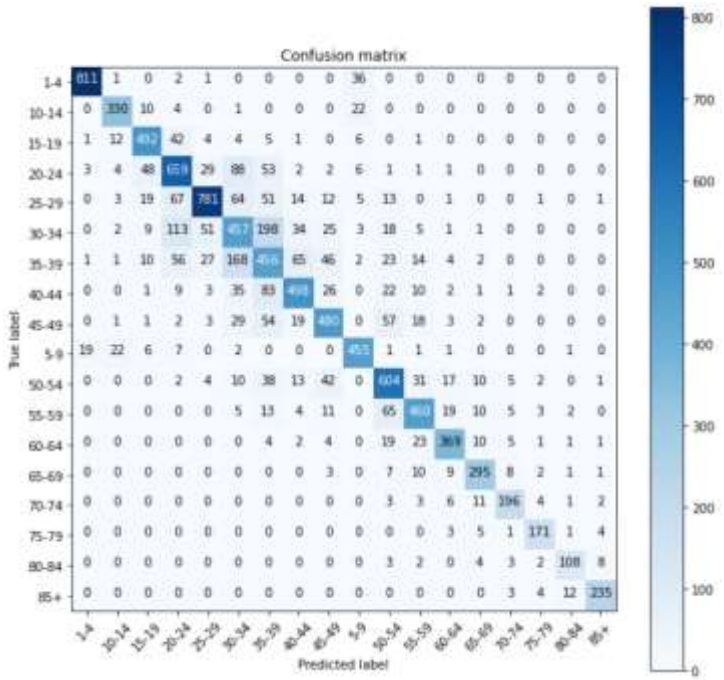
Ta thấy điểm mAP@0.5 khá cao, điều này chứng tỏ YOLOv7 đã làm tốt trong việc phát hiện các khuôn mặt cỡ lớn và trung bình trong ảnh, các khuôn mặt nhỏ trong ảnh có thể bị bỏ sót với tỉ lệ 7%.

3.2. Ước tính độ tuổi trên tập validation

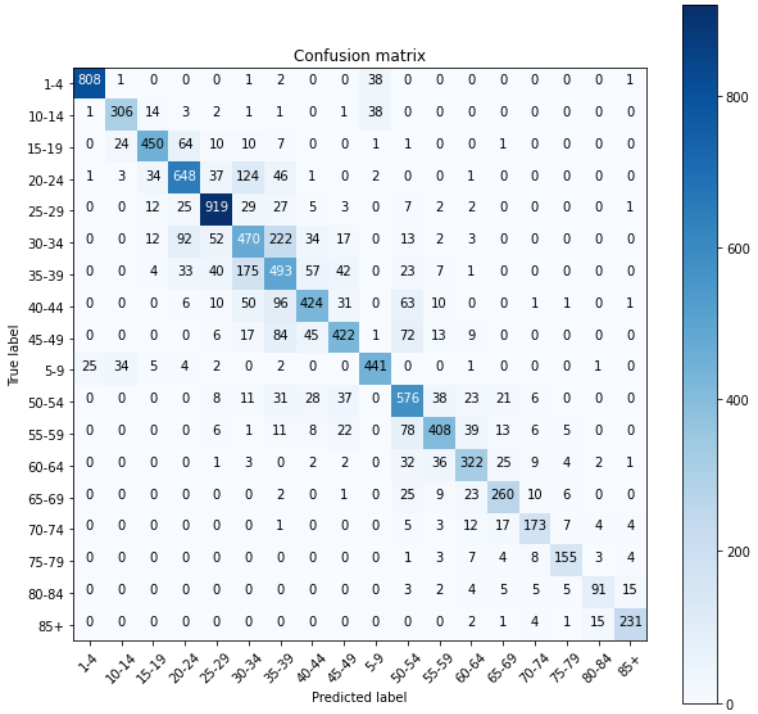
3.2.1. Confusion matrix



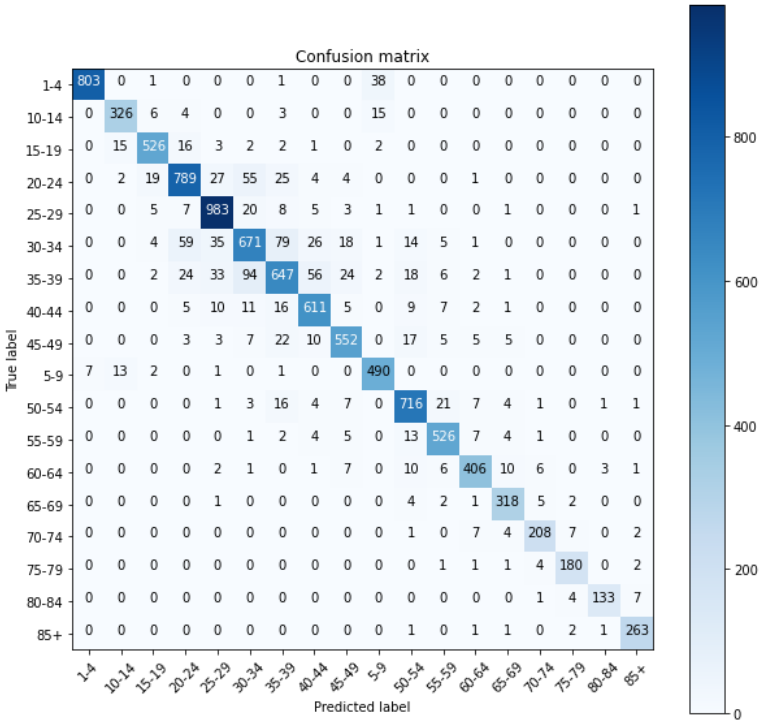
Hình 22: VGG16



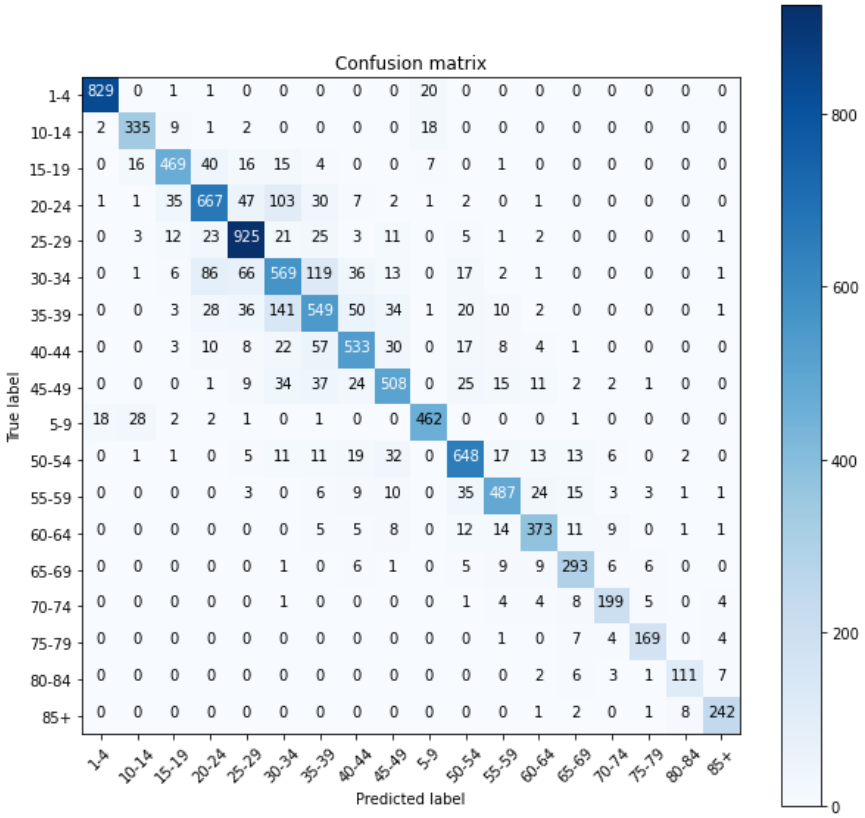
Hình 23: ResNet50



Hình 24: Inception V3



Hình 25: EfficientNetV2B0



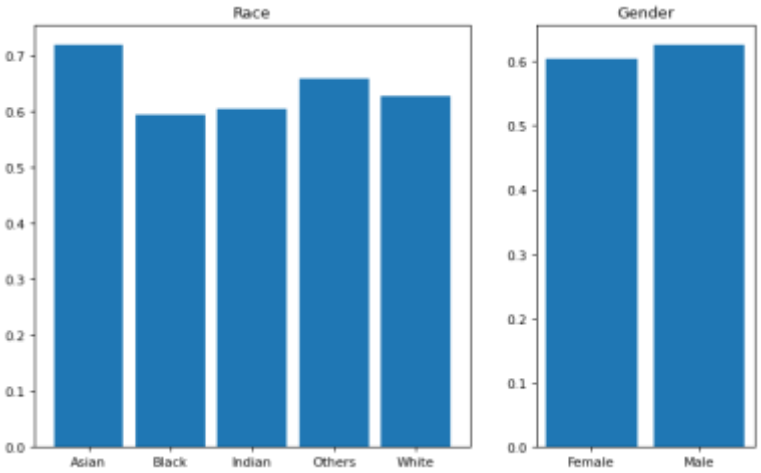
Hình 26: MobileNetV3Large

Model	Accuracy	Precision	Recall	F1-score
VGG16	0.78	0.81	0.82	0.81
ResNet50	0.76	0.8	0.8	0.8
EfficientNetV2B0	0.89	0.9	0.9	0.89
MobileNetV3Large	0.81	0.83	0.83	0.83
InceptionV3	0.74	0.76	0.75	0.75

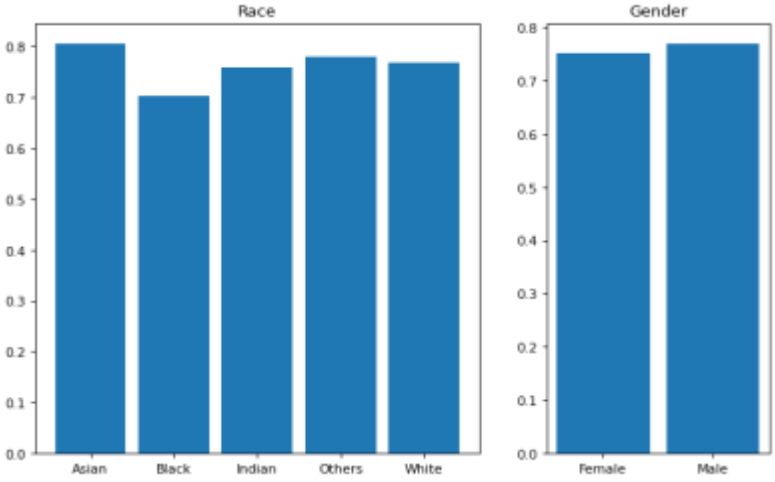
Bảng 2: Kết quả đánh giá của các model trên tập validation

3.2.2. Accuracy theo nhóm

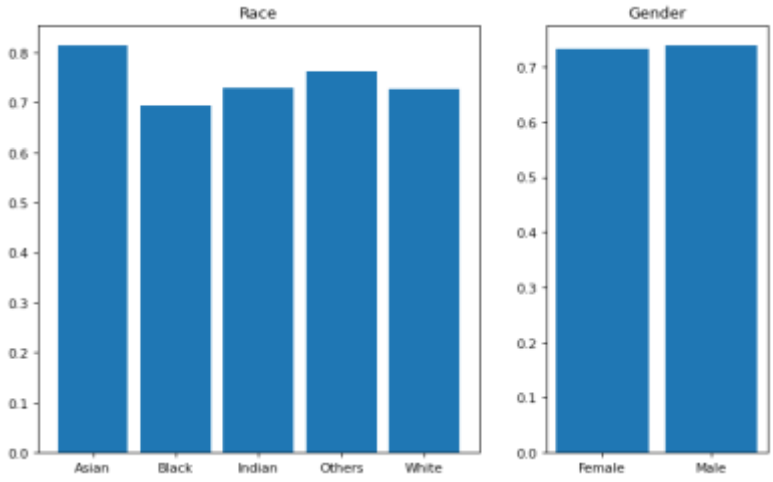
Nhóm cũng tiến hành đánh giá accuracy của các model trên 5 chủng tộc, và 2 giới tính. Kết quả như sau:



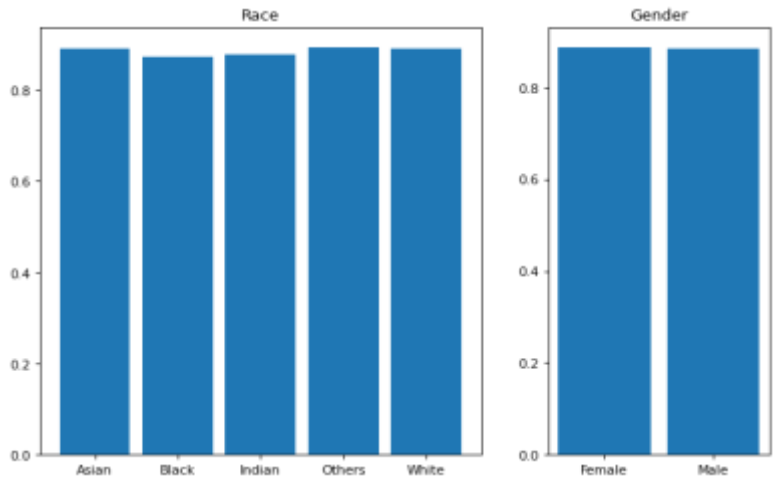
Hình 25: VGG16



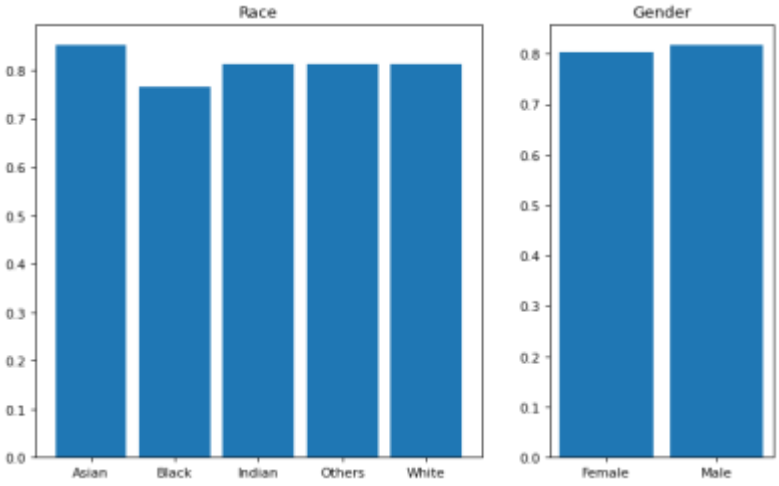
Hình 26: ResNet50



Hình 27: Inception V3



Hình 28: EfficientNetV2B0



Hình 29: MobileNetV3Large

4. Các trường hợp làm tốt và chưa tốt:

Mặc dù trong các bộ dữ liệu được huấn luyện không có ảnh đeo khẩu trang, nón nhưng các model vẫn dự đoán tốt trong một số trường hợp:

20-25

20-25

20-25

Tuy nhiên, gương mặt còn có thể bị ảnh hưởng bởi các yếu tố như trang điểm, phẫu thuật,… khiến cho khuôn mặt của một số người không thật sự phản ánh đúng độ tuổi của họ (trong hình trên nhân vật chỉ 25 tuổi nhưng sử dụng một số chất hóa học đã khiến nó già đi), đây là một thách thức của bài toán.

5. Nhận xét - hướng phát triển

5.1. Nhận xét

Có thể thấy ước tính độ tuổi chỉ từ khuôn mặt là một bài toán đầy thách thức, khuôn mặt có thể đeo phụ kiện, trang điểm, các yếu tố về chủng tộc, hoặc có những người có khuôn mặt trẻ hay già hơn tuổi thật của họ. Do đó, bài toán cần một bộ dữ liệu đa dạng hơn để cho ra kết quả tốt hơn trong thực tế, trong nhiều điều kiện khác nhau.

Các quy trình phát hiện khuôn mặt và ước tính độ tuổi đều phải sử dụng GPU để tối ưu hóa tốc độ. Việc này chưa phù hợp cho môi trường hạn chế tài nguyên khi thương mại hóa. Ngoài ra, nhóm cũng chưa tìm được phương pháp đánh giá hiệu năng của từng mô hình (thời gian phát hiện, dự đoán trên từng ảnh,…)

5.2. Hướng phát triển

Trong tương lai, nhóm sẽ tìm cách đánh giá mô hình để có thể chọn lựa, tối ưu hóa mô hình, giúp cho việc transfer learning trong các trường hợp hướng tới lứa tuổi nhất định tốt hơn, phân loại chính xác số tuổi. Hoàn thiện pipeline để giảm tiêu hao tài nguyên, đảm

bảo tốc độ. Ngoài ra, nhóm dự định sẽ kết hợp thêm các đặc trưng từ khuôn mặt như landmark thay vì chỉ đưa ảnh vào các mô hình. Do còn hạn chế về mặt tài nguyên dùng cho huấn luyện và tinh chỉnh mô hình nên nếu có điều kiện nhóm muốn thử huấn luyện trên tập dữ liệu đa dạng hơn. Và điều quan trọng nhóm cần phải thực hiện là hỗ trợ gương mặt đeo khẩu trang, đội nón, trang điểm,…

Tham khảo

[1] YOLOv7: https://github.com/WongKinYiu/yolov7

Dataset:

[2] http://shuoyang1213.me/WIDERFACE/

[3] https://susanqq.github.io/UTKFace/

[4] Hướng dẫn transfer learning and fine tuning:

https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/

Cài đặt mô hình Effiecientnet:

https://catalog.ngc.nvidia.com/orgs/nvidia/resources/efficientnet_for_pytorch

Tensorflow:

https://www.tensorflow.org

[5] Keras:

https://keras.io/

Paper tham khảo pipeline:

https://www.researchgate.net/publication/355777953_Real-Time_Age_Estimation_from_Facial_Images_Using_YOLO_and_EfficientNet

Sklearn: https://scikit-learn.org/stable/

[6] VGG16: Very Deep Convolutional Networks for Large-Scale Image Recognition
https://arxiv.org/abs/1409.1556

[7] ResNet50 paper: Deep Residual Learning for Image Recognition
https://arxiv.org/abs/1512.03385

[8] EfficientNetV2B0: Rethinking Model Scaling for Convolutional Neural Networks

https://arxiv.org/abs/1905.11946

[9] InceptionV3: Rethinking the Inception Architecture for Computer Vision

https://arxiv.org/abs/1512.00567

[10] MobileNetV3: Efficient Convolutional Neural Networks for Mobile Vision Applications

https://arxiv.org/abs/1704.04861

Google Colab

https://colab.research.google.com/