BÁO CÁO ĐỀ TÀI :

Shopee Code League 2020 Data Science

Giới thiệu về đề tài

Đây là bộ dữ liệu được lấy từ cuộc thi "Shopee Code League 2020 Data Science" được tổ chức trên Kaggle. Cuộc thi tính điểm dựa trên việc phân loại 42 label được đánh nhãn từ "00" đến "41" là các hình ảnh quảng cáo về các món đồ được bán trên shopee.

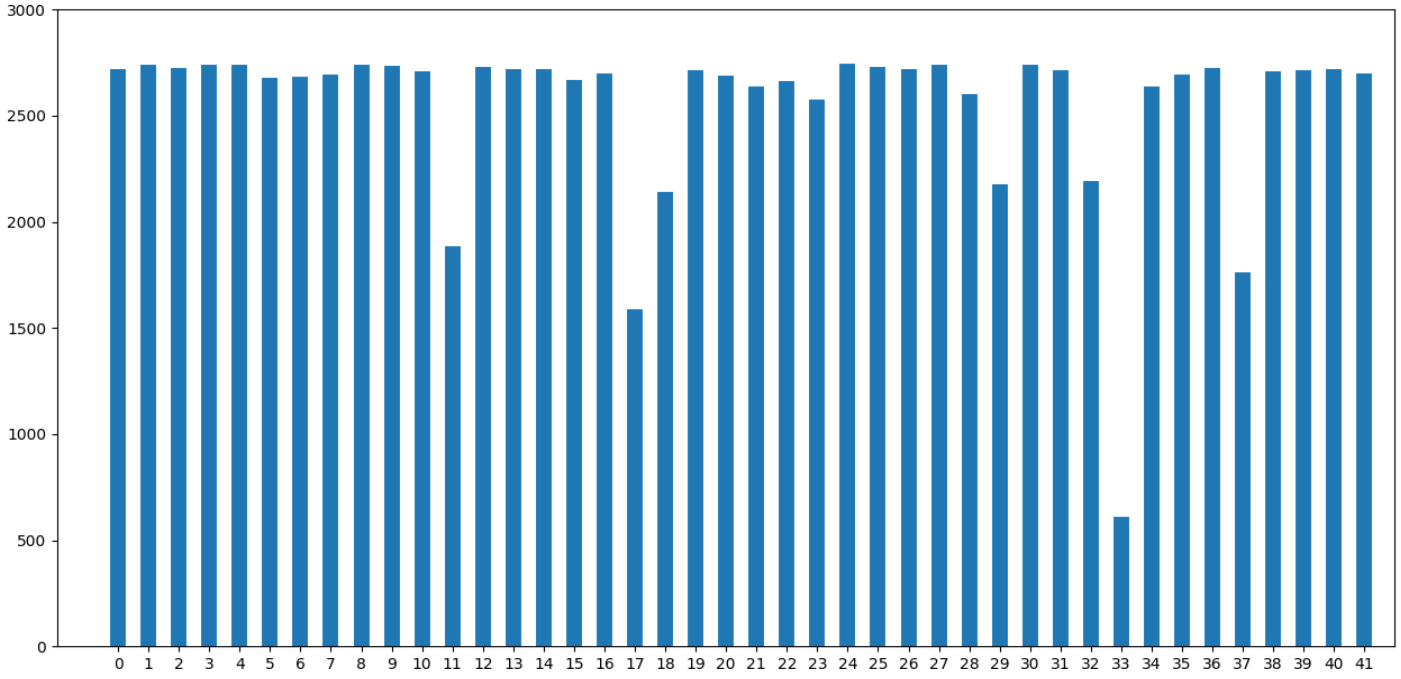
Việc phân loại này nhằm nhiều mục đích ví dụ như : việc hiển thị kết quả mong muốn cho người dùng, gợi ý các sản phẩm tương tự cho người dùng,…

1. Sơ lược về dữ liệu

a) Thông tin về dữ liệu

- Dữ liệu là dạng ảnh (có màu), được chia thành 42 label, với tổng số lượng khoảng 105000 ảnh là các ảnh quảng cáo về các mặc hàng bán trên shopee như : quần, váy, giày, nhẫn, mũ bảo hiểm,….

Dưới đây là biểu đồ thể hiện sự phân bố số lượng ảnh của từng label



Hình 1 : Phân bố số lượng ảnh trong từng label

- Từ hình 1 ta có thể thấy số ảnh trong các label tương đối đồng đều, riêng số ảnh ở các label : 11, 17, 18, 29, 32, 33, 37 là ít hơn so với các label khác

b) Đánh giá dữ liệu

Ưu điểm:

- Số lượng ảnh trong các label là khá nhiều
- Các ảnh trong label thể hiện được độ đa dạng : nhiều màu sắc, ảnh sắc nét, chụp với nhiều background (vật bớt phụ thuộc vào môi trường làm quá trình train tốt hơn), vật được chụp với nhiều góc độ,….

Nhược điểm:

- Vì đây là các mẫu ảnh quảng cáo nên có khá nhiều nhiễu trong một bức ảnh hoặc là cả bức ảnh là nhiễu
- ví dụ như các ảnh trong label 00(váy) dưới đây sẽ cho ta thấy rõ hơn về các kiểu nhiễu có trong các label

Hình 2 : có nhiều vật thể khác xuất hiện cùng vật trong label

Như trong hình 2 ngoài váy ta còn thấy chậu hoa, logo “minos design” trong ảnh (trong ảnh có nhiễu)

Hình 3 : ảnh không liên quan tới label

- Như trong hình 3 là một vật nằm trong label “00” là không liên quan gì đến váy (ảnh nhiễu)

Hình 4 : Có nhiều vật thể chính (váy) trong ảnh

- Trong quá trình train ta đưa ảnh vào model thì model sẽ hiểu cả tấm ảnh là váy, tuy nhiên như trong hình 4 thì trong ảnh có tới 4 cái váy điều này cũng sẽ gây ra khó khăn cho model khi học tập

2. Hướng tiếp cận

- Với lượng dữ liệu lớn khoảng 9Gb ảnh ta không thể load hết dữ liệu một lần trên colab nên sẽ cần đến file TFRecord. TFRecord là một định dạng Tensorflow được sử dụng để lưu trữ một chuỗi các bản ghi nhị phân trong trường hợp này ta sẽ sử dụng để lưu trữ hình ảnh. File TFRecord này trong tập dữ liệu "Shopee Code League 2020 Data Science" cũng đã có sẵn

- 2 model được sử dụng trong bài là ResNet và VGG, vì đây là 2 model dễ hiểu, dễ dàng sữ dụng và được tích hợp sẵn trong lớp applications của tensorflow và đều đã đạt các kết quả cao trên tập dữ liệu imagenet

- Sau khi đã train qua model ta sẽ predict lại trên toàn bộ tập train để đánh giá mức độ học của từng label có tốt hay không

- Trong bài viết này em sẽ cố gắng phân tích các vấn đề gặp phải và đã tìm hiểu được cũng đề xuất 2 phương án để tiếp cận bài toán

3. Các mạng được sử dụng trong bài

a) Sơ lược về mạng VGG

- Mạng VGG được phát triển vào năm 2014, có thể được phân chia thành hai phần: phần đầu tiên bao gồm chủ yếu các tầng tích chập và tầng gộp, còn phần thứ hai bao gồm các tầng kết nối đầy đủ.

- Hình 5 dưới đây là cấu trúc biến thể của mạng VGG khác nhau về số chiều sâu (được train trên tập Imagenet): như VGG11 có số Layer là 8Conv-3 FC, VGG16 là 13Conv-3FC,…. Ta có thể thay đổi lớp FC cho phù hợp với tập dataset của mình

- Chiều sâu của lớp Conv trong VGG bắt đầu với 64 filter ở lớp đầu tiên và tăng gấp đôi sau từng block cho đến khi đạt tới 512 filter

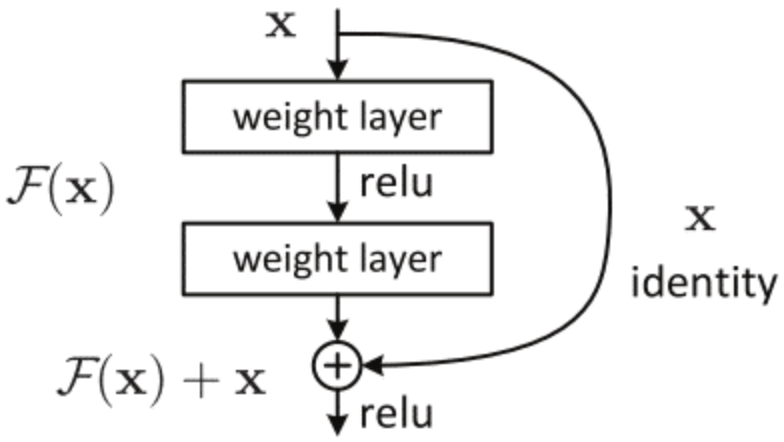
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Hình 5 : kiến trúc mạng vgg (trên tập datasets imagenet)

b) Sơ lược về mạng ResNet

- Mạng VGG và các mạng CNN nói chung là một mạng được thiết kế dựa trên việc sắp xếp chồng các lớp chập lên nhau tạo nên độ sâu của mạng Tuy nhiên, tăng độ sâu mạng không chỉ đơn giản là xếp chồng các lớp lại với nhau. Mạng sâu rất khó huấn luyện vì vấn đề vanishing gradient – vì độ dốc được truyền ngược trở lại các lớp trước đó, phép nhân lặp đi lặp lại có thể làm cho độ dốc cực nhỏ. Kết quả là, hiệu suất của mạng bị bão hòa hoặc giảm hiệu quả nhanh chóng.

- Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm hoặc hàng nghìn lớp chập. Ý tưởng chính của ResNet là sử dụng kết nối tắt đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một residual block như trong hình sau:



Hình 6 : khối residual block trong ResNet

- Theo hình 6 ta thấy input X sẽ được bổ sung vào đầu ra của layer, hay chính là phép cộng mà ta thấy trong hình minh họa, việc này sẽ chống lại việc đạo hàm bằng 0, do vẫn còn cộng thêm X. Với kiến trúc này, các lớp ở phía trên có được thông tin trực tiếp hơn từ các lớp dưới nên sẽ điều chỉnh trọng số hiệu quả hơn.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

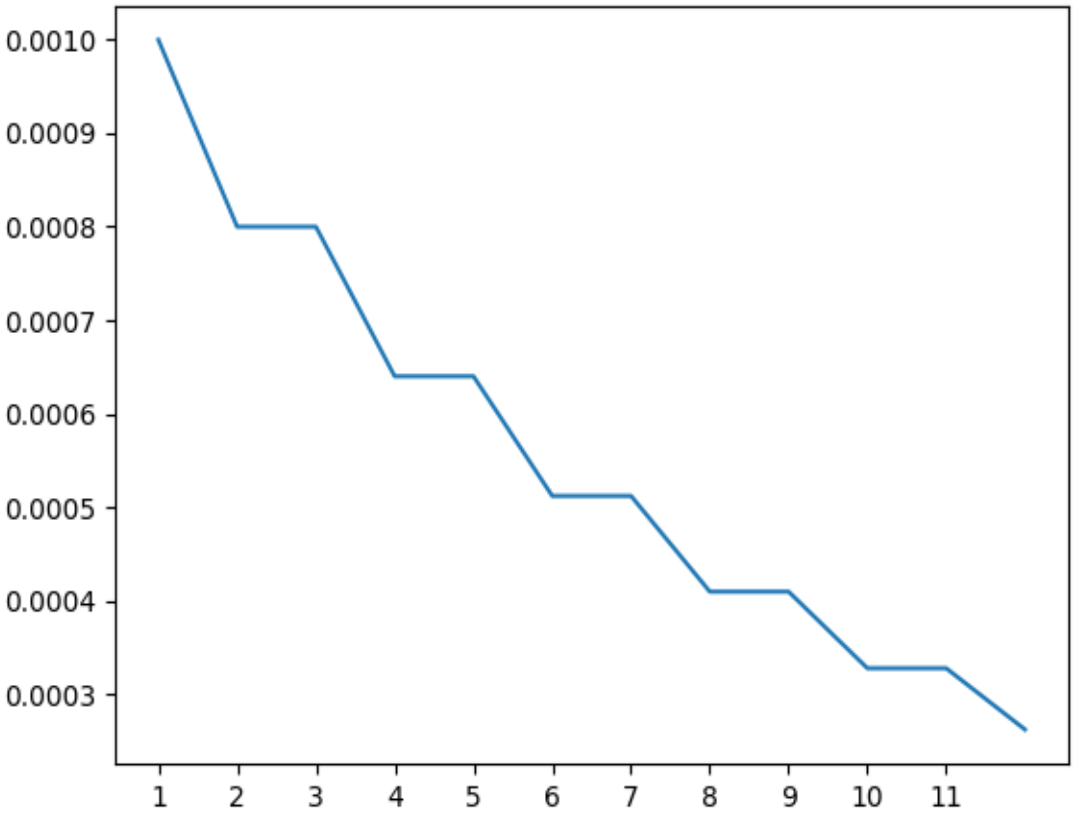
Hình 7 : Kiến trúc mạng ResNet (trên tập datasets imagenet)

- Từng khối nằm trong dấu ngoặc “[ ]” là từng khối Residual block

4. Triển khai bài toán

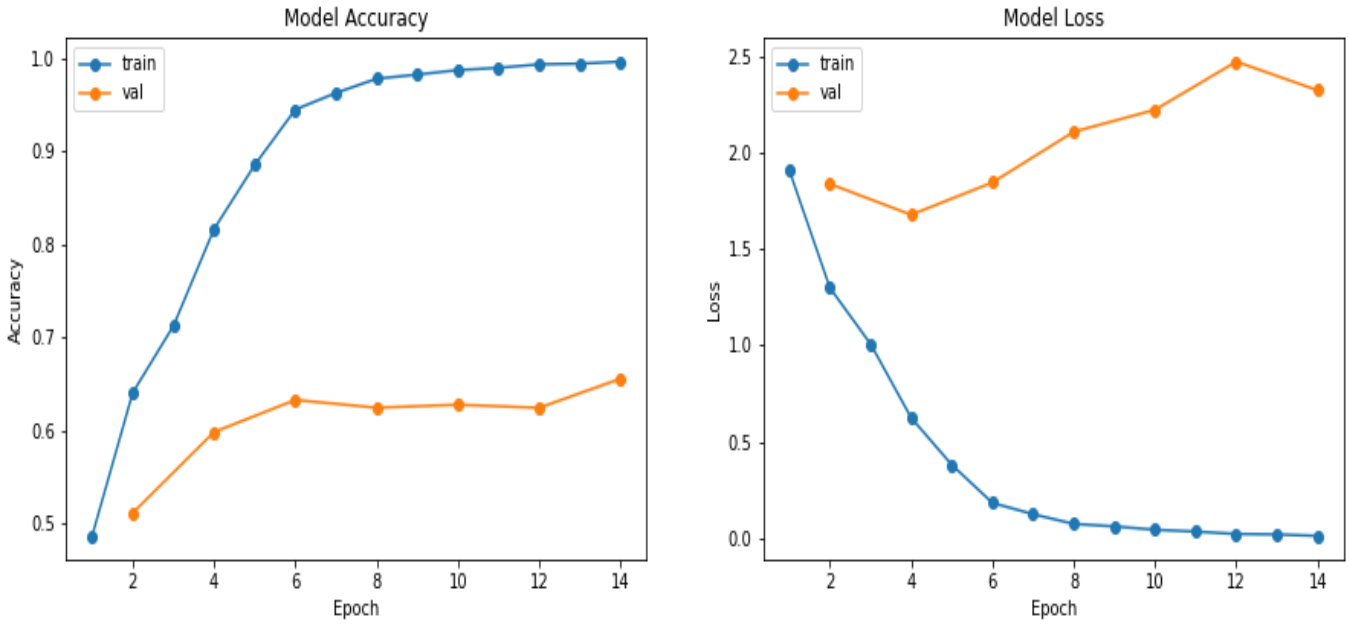
a) Hướng tiếp cận đầu tiên

- Dựa vào độ phức tạp của dữ liệu (nhiều nhiễu), và cũng như số lượng dữ liệu là rất lớn nên ta sẽ chọn model ResNet50(khoảng 23 triệu tham số chưa kể Fully Connected Layer) để phân loại 42 label
- Kĩ thuật ta sử dụng là dùng Transfer learning đã được train trên tập imagenet, và sau đó train lại toàn bộ trọng số, , kết hợp thêm với lớp Fully Connected Layer do ta tự tạo
- Learning rate sau nhiều lần khảo sát thì ta sẽ chọn ban đầu bằng 0.001 và cứ sau 2 epoch giảm đi 80% so với learning rate trước đó



Hình 8: đồ thị learning rate trong quá trình train

- Vì việc predict cho tập validation là khá lâu nên cứ 2 epoch ta sẽ cho tính accuracy trên tập validation (chia từ tập train theo tỉ lệ 4:1) , và thu được kết quả như sau:

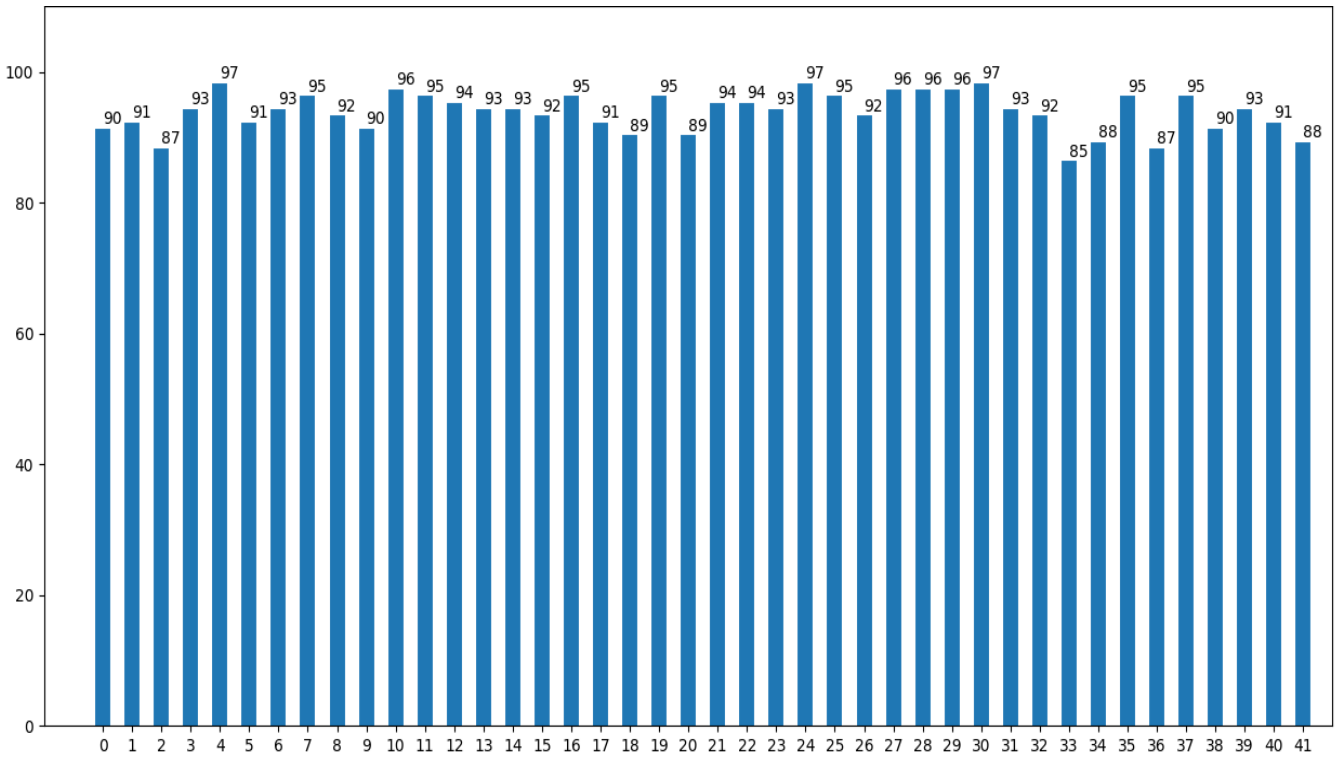


Hình 9: Model ResNet50 train cho cả 42 label

- Về accuracy : accuracy trên tập train học rất tốt gần 100% nhưng trên tập validation thì không được như vậy (chỉ được khoảng 65%)
- Về loss : loss trên tập train rất thấp gần về 0 nhưng trên tập validation thì ngược lại loss khá cao và có chiều hướng tăng kể từ sau epoch thứ 4

⇨ Dựa vào accuracy, loss trên tập train và validation ta có thể thấy model đã học tập không tốt có thể là vì 2 lý do sau:

- Model bị overfiting
- Vì dữ liệu có quá nhiều nhiễu nên dẫn đến việc phân bố ảnh trên tập train, validation là tương đối khác nhau => kết quả dự đoán trên tập validation là không tốt

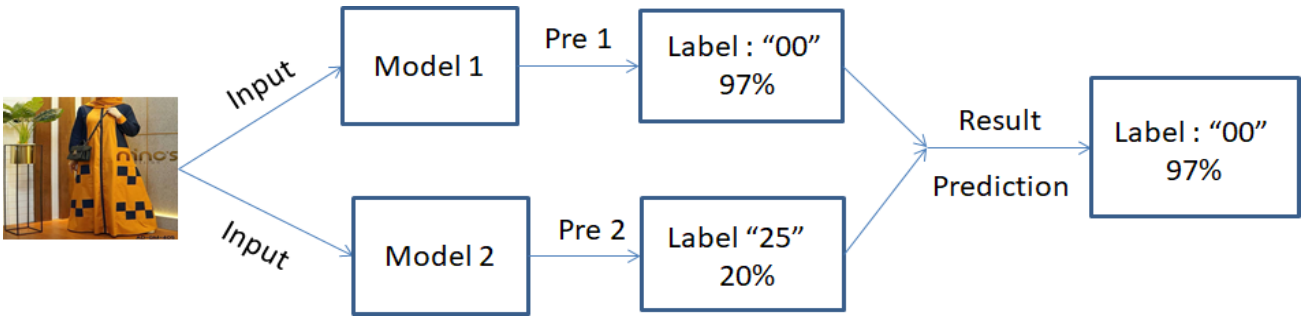


Hình 10: Kết quả dự đoán trên từng label (tính theo %)

- Ta có thể thấy model cho kết quả predict khá cao trên toàn bộ tập train(dự đoán đúng 97769 ảnh, đạt tỉ lệ 0.92) nhưng vì model đã có thể bị overfitting nên khi dự đoán trên tập dữ liệu mới kết quả khả năng cao sẽ không còn tốt như vậy
- Liệu có cách nào học tốt trên tập dữ liệu này không ? Ta sẽ cùng đến với phương án thứ 2 với hi vọng sẽ làm cái thiện accuracy và loss trên tập validation

b) Hướng tiếp cận thứ 2 :

- Lần này ta sẽ chia tập dataset ra làm 2 phần, mỗi phần sẽ gồm 21 label, phần đầu tiên gồm là các label từ “00” đến “20”, phần thứ 2 là các label từ “21” đến “41”
- Ta cho mỗi một model học trên 2 phần này (chuyển từ bài toán phân loại 42 label xuống chỉ còn 21 label)
- Khi dự đoán một tấm ảnh thì tấm ảnh sẽ được cho qua cả 2 model, kết quả dự đoán của model nào cho ra kết quả lớn hơn thì sẽ chọn dự đoán của model đó, để cụ thể hơn ta đến với hình minh họa bên dưới



Hình 11: mô hình dự đoán của phương án 2

- Khi dự đoán 1 tấm ảnh lúc này sẽ đi qua 2 model cho ra kết quả dự đoán tấm ảnh của mỗi model(model 1 : dự đoán là label "00" với xác suất là 97%, model 2: dự đoán là label "25" với xác xuất 20%) sau đó sẽ lấy kết quả dự đoán cuối cùng là kết quả có xác suất cao hơn (như trong ảnh kết quả cuối cùng là label "00")
- Model được lựa chọn ở trong phương án này là VGG16 cho cả 2 model 1 và model 2
- Kỹ thuật ta dùng trong phương án này vẫn là transfer learning, ta lấy lại toàn bộ trọng số của phần Conv, là cho train lại phần trọng số này, kết hợp thêm với lớp Fully Connected Layer do ta tự tạo

- Tại sao lại chọn VGG16?

➢ Từ phương án 1 ta có thể thấy model đã bị overfitting nên ta cần 1 model khác có số tham số nhỏ hơn (VGG16 có gần 15 triệu tham số chưa kể Fully Connected Layer) để train qua tập dữ liệu đã ít hơn 1 nửa so với ban dầu

➢ VGG là một model tương đối dễ hiểu cũng như dễ cài đặt, đã được tích hợp sẵn trong lớp application nên ta chỉ việc gọi lại và dùng

- Việc chọn cả model VGG16 cho cả 2 phần thì có khác gì so với việc train toàn bộ 42 label qua VGG16?

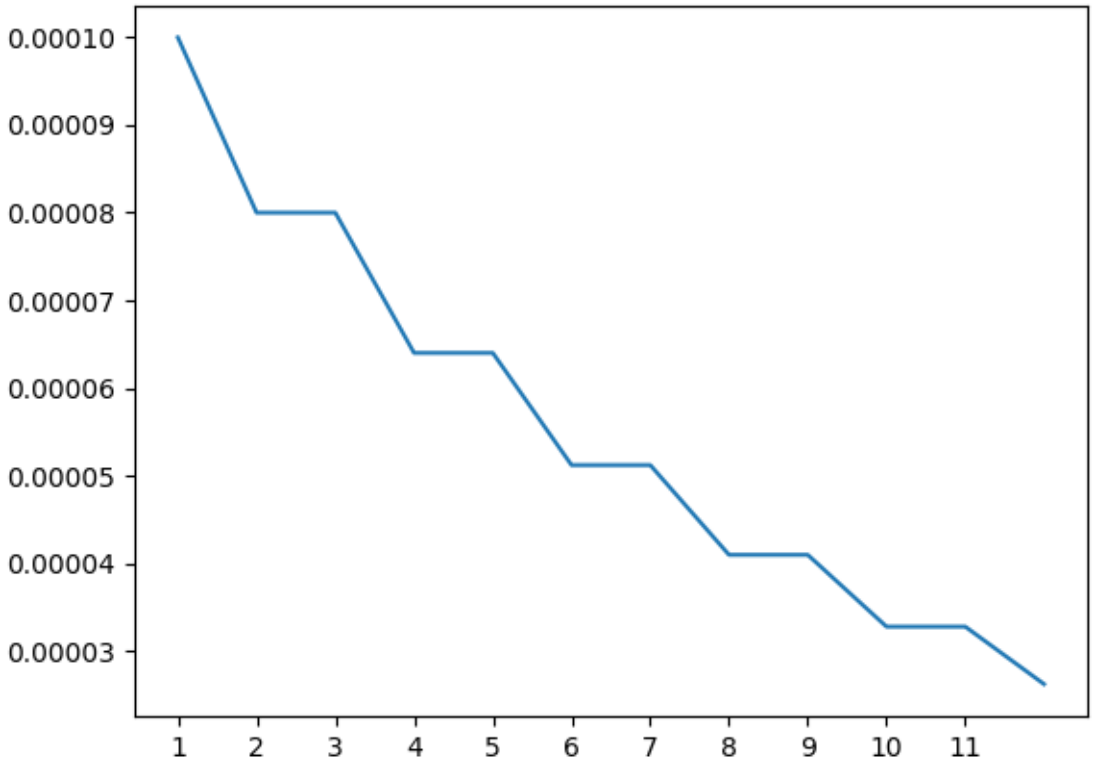
➢ Khi ta train trên từng phần trên VGG16 sẽ giúp cho model học kĩ hơn 21 label của mỗi phần, từ đó làm tăng accuracy và giảm loss trên tập validation

➢ Sau khi train trên mỗi phần ta sẽ được 2 tập trọng số khác nhau, điều này có được nhờ sự học tập "tập trung" hơn khi model chỉ cần học 21 label

➢ Với cách chia 42 label ở trên ra làm 2 nửa, ta mong muốn rằng mỗi model sẽ cho kết quả tốt (xác suất cao hơn) đối với label mà model đó đã học, còn model kia sẽ cho kết quả thấp hơn (xác suất thấp hơn). Như trong ví dụ ở hình 11 trên ta có thể

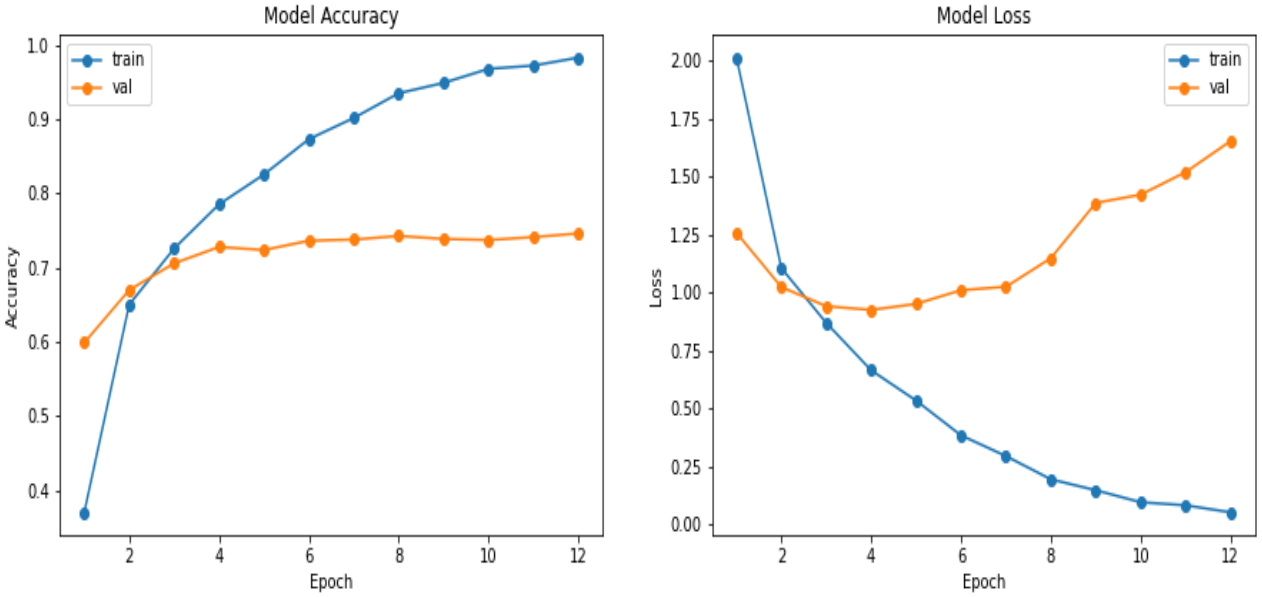
thấy đầu vào là 1 tấm ảnh của label “00” vì model 1 đã học label này nên sẽ cho ra kết quả tốt hơn so với model 2 chưa từng học ảnh trong label này

- Learning rate sau nhiều lần khảo sát thì ta sẽ chọn ban đầu bằng 0.0001 và cứ sau 2 epoch giảm đi 80% so với learning rate trước đó

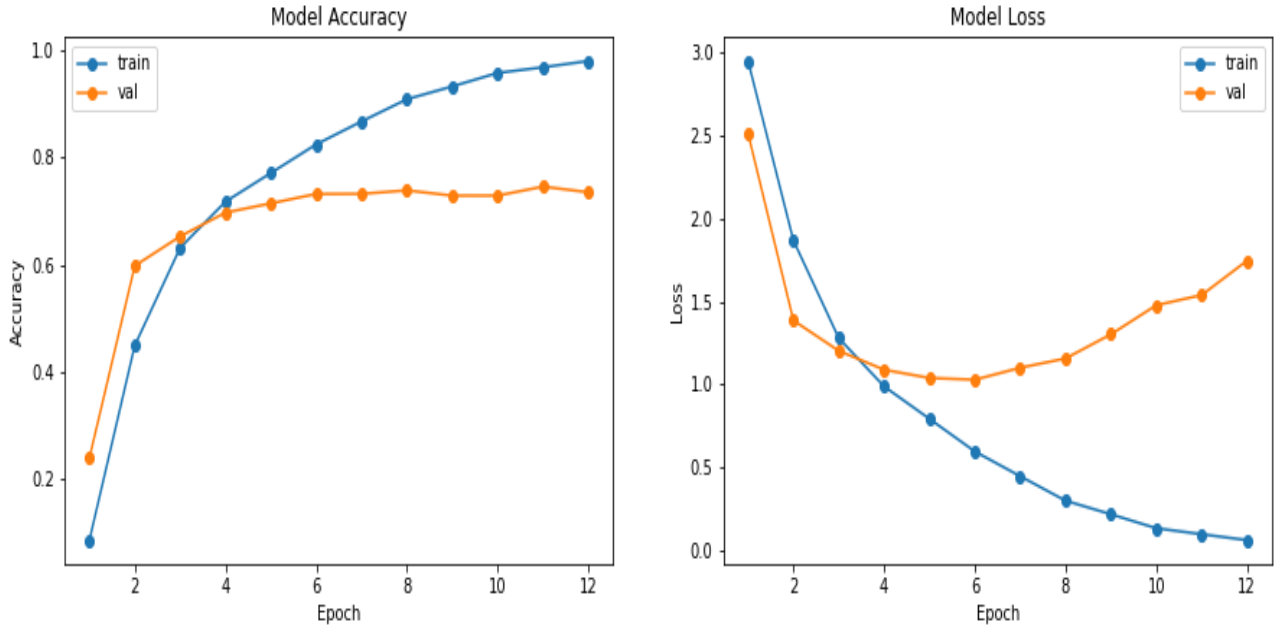


Hình 12: đồ thị learning rate trong quá trình train

- Dưới đây là kết quả thu được của 2 model trên từng phần



Hình 13: Model 1 train cho 21 label đầu



Hình 14: Model 2 train cho 21 label sau

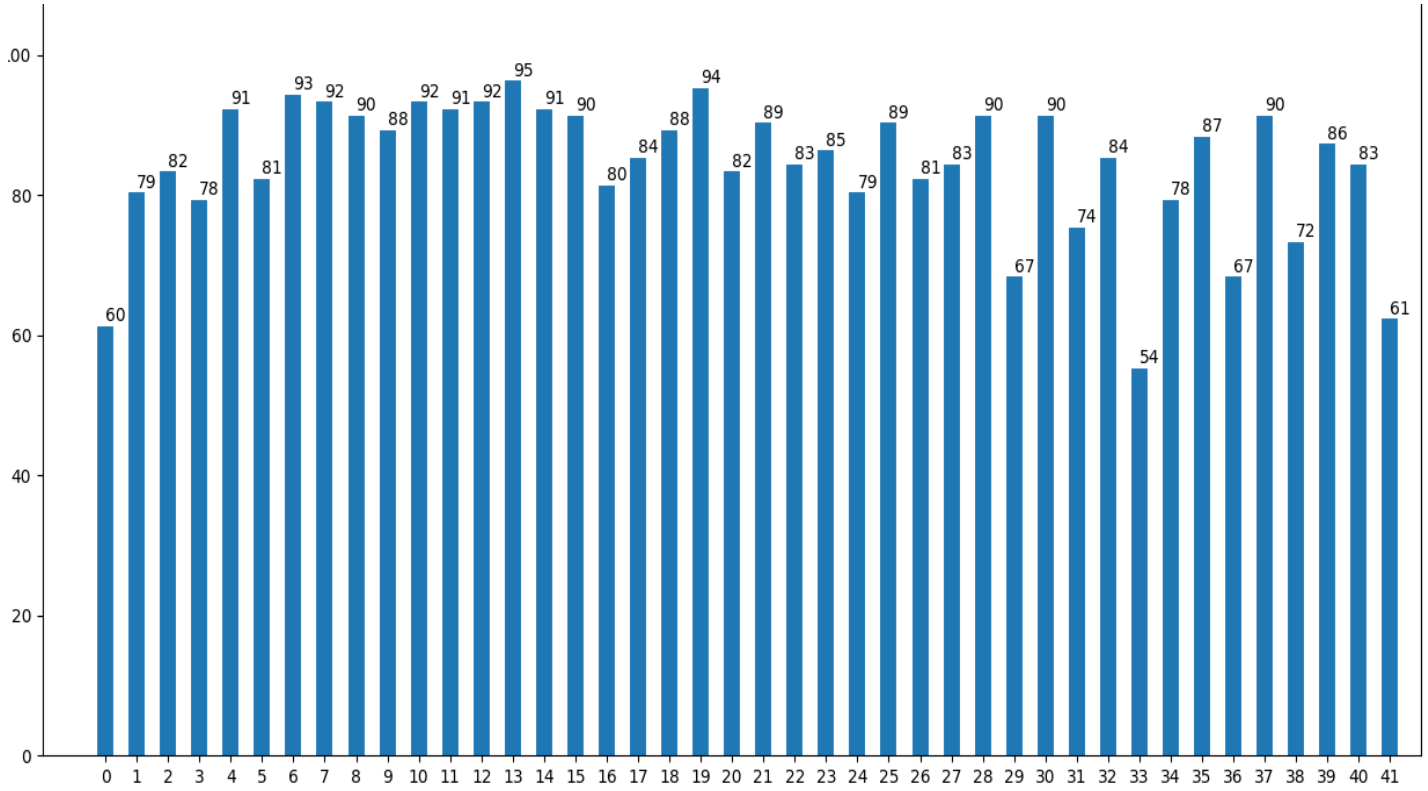
- Nhận xét về đồ thị :

➢ Tuy học trên 2 tập dữ liệu khác nhau nhưng hình dạng đường đi của đồ thị thì lại tương đối giống nhau

➢ Accuracy(cao nhất khoảng 98%), loss(thấp nhất cỡ 0.05) trên tập train rất tốt nhưng trên tập validation thì ta lại thấy val_loss khá cao và có xu hướng tăng từ epoch thứ 4, val_acc cũng không quá tốt (đạt lớn nhất khoảng 75%)

➢ Cả 2 model này đều cho val_acc cỡ 75% nên ta có thể kết luận rằng val_acc trên cả tập dữ liệu là 75%

⇨ Dựa vào accuracy, loss trên tập train và validation ta có thể thấy model đã học tập không tốt có thể là vì 2 lý do tương tự như với phương án 1



Hình 15: Kết quả dự đoán trên từng label (tính theo %)

- Ta có thể thấy model cho kết quả predict khá cao trên toàn bộ tập train(dự đoán đúng 90343 ảnh đạt tỉ lệ 0.86) nhưng có nhiều label lại cho kết quả dự đoán không tốt (dưới 70%)

c) So sánh 2 phương pháp:

	Val_Accuracy Max	Val_Loss Min	Số ảnh dự đoán đúng
Hướng tiếp cận 1	65%	1	97769 (92%)
Hướng tiếp cận 2	75%	1.6	90343 (86%)

Hình 16: Bảng so sánh 2 hướng tiếp cận

- Ta thấy rằng hướng tiếp cận thứ 2 đã cho kết quả val_acc cao hơn so với hướng tiếp cận thứ nhất (khoảng 10%), và val_loss cũng nhỏ hơn so với hướng tiếp cận thứ nhất (khoảng 0.6)
- Hướng tiếp cận 2 có val_acc cao hơn điều này giúp cho việc dự đoán trên một tập dữ liệu mới sẽ không còn nhiều sai sót
- Cả 2 phương án đều đã bị overfiting nhưng hướng tiếp cận thứ 2 lại cho ta thấy rằng khi chia nhỏ tập dữ liệu thành nhiều phần rồi tạo thành 1 mô hình dự đoán với sự tham gia của các model nhỏ như vậy sẽ cho val_acc cao hơn nhiều, val_loss cũng thấp đi

⇨ Điều này mở ra hi vọng khi ta chia tập dữ liệu thành những phần nhỏ hơn, sau đó tìm model phù hợp cho từng phần sẽ giúp val_acc, val_loss trên toàn tập dữ liệu cải thiện đáng kể