CS230

Transfer Learning for Multi-label Street Fashion Images

Yiying Hu
Stanford University
yhu1102@stanford.edu

Abstract

We trained a deep neural network to predict the clothing attributes of fashion images using transferred learning on top of pre-trained VGG-16. The motivation of training this model is to create "search engine" for fashion images, the possible use case will be for fashion journal editors to find the similar outlook from their own database of images and illustrate for blog readers. The challenge in training the model is that the outputs of the training model (i.e. clothing attributes) are high dimensional in the format of [0, 0, . . . , 1] for each image to indicate binary clothing attributes, and most of the elements in the vector are 0, which causes the label to be imbalanced. We tested the impact of changing number of clothing attribute as outputs, and putting high weights on important features such as gender, and adjusting the weights of the cross-entropy loss function when training the deep neural network. After these tests to obtain the best hyper-parameters, we compared our model with the original untrained VGG-16. The approach we used to make comparison could also work as a recommendation system - we put a new test image as well as every image in the existing database into our model and VGG-16 untrained, and extracted their results from a common intermediate layer of both models. For each model, we used cosine similarity to find pictures from our database whose extracted intermediate layers are closest to the test image. These most similar pictures from the image dataset are 'recommended' similar looks by the models. By manually looking at these images found by both models, we can evaluate models' performance.

1 Introduction

Streets fashion photos are becoming more and more popular on the social network nowadays. However, with the humongous amount of pictures online, many of them are missing description. By leveraging the deep neural network, we could label the image to see whether they meets certain attributes, such as color is black or not, or pattern is dot or not. We used transfer learning with the foundation architecture of VGG-16 whose first 12 layers are not trainable, and added a flattened layer on top of that so that the output vector's length is equal to the number of clothing attributes labels. To feed the model, every sample input is a color images of street fashion (224x224x3), and output is a vector of length (20,). Then we evaluated the performance of the model by extracting its VGG 17th layer which is fine-tuned, and find the image from the existing image database whose cosine similarity of the 17th layer is closest to the new test image's 17th layer, which guarantees the architectures of two models are the same so that it's an apple-to-apple comparison. By manually viewing the the images whose 17th layers are most similar to that of the new test image, we can determine how good the model is at performing the multi-label classification by eyeballing the similarity.

The commercial value from making the fashion photos able to 'talk' about their attributes is tremendous. Many online users arises interests in shopping for clothes when they see fashion bloggers' pretty street snapshots and get aware of the current season's trend. If we are able to label the street snapshots of their color pattern quickly, then the advertiser can quickly and automatically search for ones with the similar labels and display links for recommendation, thus improving the ads conversion rate.

2 Related work

There are many papers described the advantage of deep neural network in image classification[1]. Therefore, we think a pre-trained VGG-16 is a good foundation for transfer learning for image classification issue. There has been really advanced and accurate ML models for measuring similarity between items in images, such as deep ranking model [5]. But this kind of similarity might not necessary for fashion recommendation because we are looking for not images that most look like each other, and rather images with similar trendy elements is good enough. For fashion image related works, there is a dataset Fashion-MNIST which is predicting multi-class single output with input images of black/white; this study image might not be applicable to the real-world complicated images [2]. DeepFashion[3] does a great job at detecting clothing attributes and landmarks on online shopping clothing pictures - those with white and plain background. As for fashion recommendation, there is FashionNet that achieves personalized outfits recommendation by fine-tuning VGGNet stacked with FC[4]. But different from all these excellent works with high performance at labelling clothing attributes and recommendation, we want to study predicting on more noisy pictures where the fashion model stands in a more diverse background, and solve a multi-label classification problem.

3 Dataset and Features

Originally there are 1856 pictures [6], and each image is processed with:

1. Data augmentation for 10 times, using rotation, width and height shifts, etc. Some images only have 9 data augmentation copies because of generating conflict names, so final amount of processed images is 20412)
2. Resized to the resolution of 224x224x3
3. Normalized by dividing 225 and between 0 and 1

The training and testing split is 67 percent and 33 percent. Therefore, the shape for training is a tensor of (13676, 224, 224, 3) and test is (6736, 224, 224, 3). Sample input is figure 1 on page 3. For model output, is a 11-d or 20-d vectors, because we build two models - one with 20 labels and the other with 11 in order to see how the dimension of prediction output vector impact the model performance. The corresponding description are:

- 20 labels: Black, Blue, Brown, Collar, Cyan, Gender, Gray, Green, Pattern-floral, Pattern-graphics, Pattern-plaid, Pattern-solid, Pattern-spot, Pattern-stripe, Purple, Red, Scarf, Skin-exposure, White, Yellow
- 11 labels: Black, Blue, Brown, Cyan, Gender, Gray, Green, Purple, Red, White, Yellow

4 Methods

4.1 Loss Function

While training the neural network, the challenge is since this is a multi-label classification, and the actual data have most of labels to be 0 (84 percent 0s 16 percent 1s). Therefore, the model tends to predict everything as 0 to obtain a high performance (similar to the spam email problem). The prevent the labels to be all zeros, we modified the loss function by editing the weights and higher the weights on positive prediction result so the loss function becomes minimize.

Therefore, instead of using original binary cross-entropy, we put a weight K>1 on positive labels.

$$-[\sum K * y_t * log(y_p) + (1 - y_t) * log(1 - y_p)] \tag{1}$$

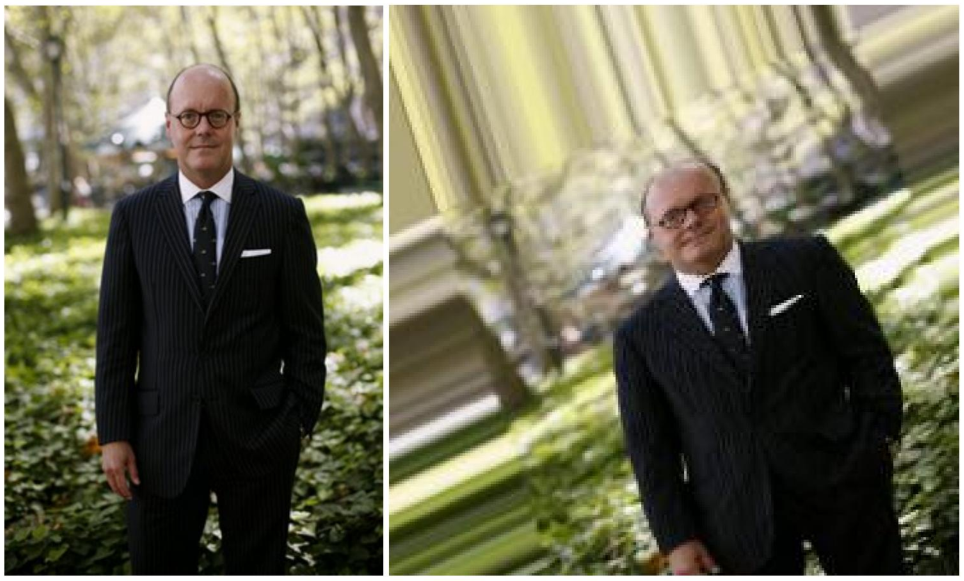


Figure 1: Left hand side is image from original dataset; right hand side is after data augmentation and resize to 224x224x3 (not normalized yet)

Here $\mathrm{y_p}$ is predicted output and $\mathrm{y_t}$ is actual output

4.2 Transfer Learning Architecture

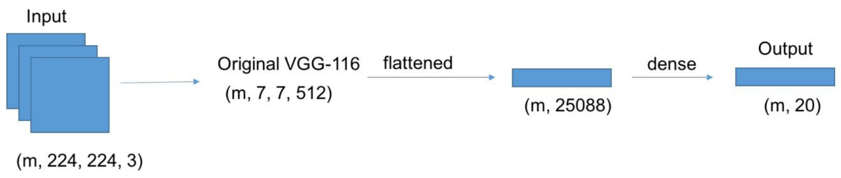


Figure 2: Transfer Learning Architecture

As you can see in the figure 2 on page 3, the network is stacking two layers on top of VGG-16 in order to obtain a n x 1 output (n is number of predicted attributes as needed), and the first 12 layers are fixed and the rest layers are allow to be fine-tuned.

4.3 Accuracy Metrics

When evaluating the performance, instead of using accuracy, we also calculated the F1 score so that the predictors needs to do well on both precision and recall. Under this accuracy measure, when the model blindly predicts everything as 0, the precision will be too low and the score won't be high, which could accurately reflect the biased prediction issue if the model does that.

4.4 Compare the Model Result with Original VGG-16

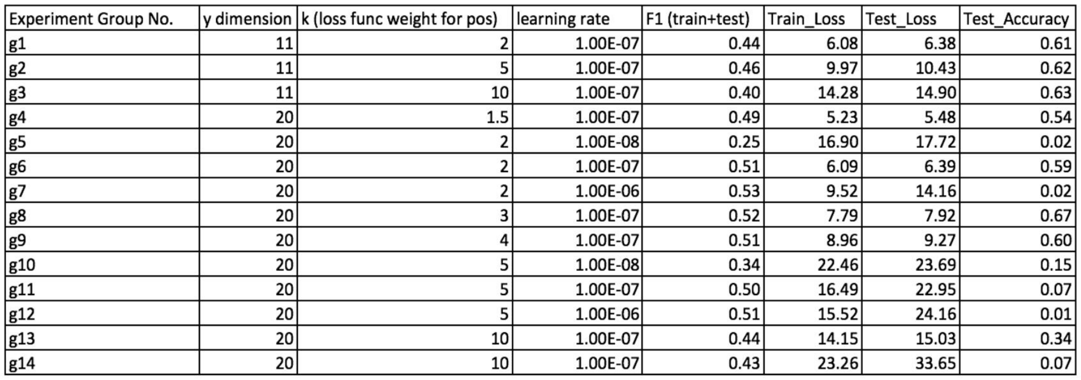
Search every image from the original training database (1856 pictures), and calculated their output from the 17th layer 'block5_conv3 (Conv2D)' (after flatten it changes from 14x14x512 to 100352x1) from each model and compared to that from the new test image's output using cosine similarity to find nearest 5 images from the database.

5 Results and Discussion

5.1 Result

We set batch to be 32 and epoch to be 5 because papers to deal with similar issue [4] uses epoch = 32 and we found after 5 epoch the loss function and accuracy become pretty stable, so there is no need to waste resource on further training.

The K from loss function equation (as explained above) and learning rate need to be fine-tuned. Other than that, we also found that some attributes of the output are more essential than others by intuition, and thus we tested the model result by putting more weights on them. We also tested the performance of model with 11 versus 20 outputs. All of these combinations are trained and evaluated using accuracy metrics (mainly look at F1 - score and test loss and test accuracy) and chose the hyperparameters whose metrics result is the best. See Figure 3 on page 4 for the experiment results.



Experiment Group No.	y dimension	k (loss func weight for pos)	learning rate	F1 (train+test)	Train_Loss	Test_Loss	Test_Accuracy
g1	11	2	1.00E-07	0.44	6.08	6.38	0.61
g2	11	5	1.00E-07	0.46	9.97	10.43	0.62
g3	11	10	1.00E-07	0.40	14.28	14.90	0.63
g4	20	1.5	1.00E-07	0.49	5.23	5.48	0.54
g5	20	2	1.00E-08	0.25	16.90	17.72	0.02
g6	20	2	1.00E-07	0.51	6.09	6.39	0.59
g7	20	2	1.00E-06	0.53	9.52	14.16	0.02
g8	20	3	1.00E-07	0.52	7.79	7.92	0.67
g9	20	4	1.00E-07	0.51	8.96	9.27	0.60
g10	20	5	1.00E-08	0.34	22.46	23.69	0.15
g11	20	5	1.00E-07	0.50	16.49	22.95	0.07
g12	20	5	1.00E-06	0.51	15.52	24.16	0.01
g13	20	10	1.00E-07	0.44	14.15	15.03	0.34
g14	20	10	1.00E-07	0.43	23.26	33.65	0.07

Figure 3: Experiment Result from Tuning Hyperparameters and Change Output Dimension

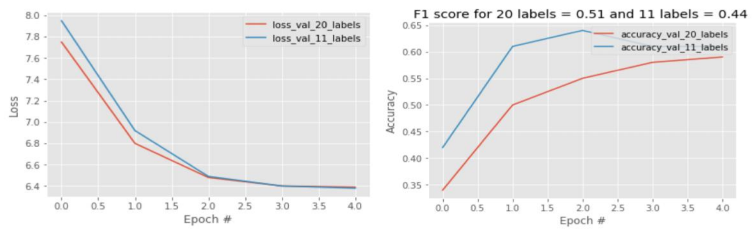


Figure 4: Dimension Change Model Performance; F1 score for 20 labels = 0.51 and 11 labels = 0.44

5.2 Discussion

Since the selection of the best model is pretty subjective because the F1 difference is not big, so does train and losses, we also tried g8 and g9. It turns out that the recommended images are the same 5 pictures, which means the difference of a parameter in the range of 2 to 4 for K is not significant.

Pictures recommended by our model is in Figure 7 on page 5 and recommended by VGG original is in Figure 8 on page 6. It is obviously that our model performs better than the VGG original in that 1. our prediction never labels gender wrong 2. it is better at extract an important fashion attribute of the input picture which is the red color. Therefore, it proves that trained VGG with a few front layers fixed is doing a better job than untrained.

The model result indicates that it is not overfit because Figure 5 on page 5 shows the loss function is always decreasing in both train and test data.

We selected g6 to be have the best performance and applied that model to extract the top 5 similar images from the database. The result is shown in Figure 4 on page 4.By comparing the group 1 and group 6, we can also conclude that higher dimension of classification output doesn't necessarily means worse performance. Figure 4 on page 4 shows for 20-label model, the accuracy, loss as well as

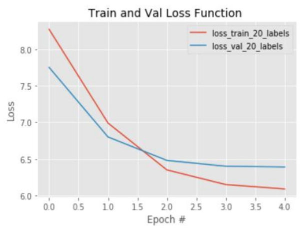


Figure 5: View Training and Testing Data Loss Functions

F1 score (which is calculated using the whole dataset to predict - including training and testing) is higher than that of 11-label. However, the intuition is that with more dimension, the output labels tend to have correlation with each other, which provides the training model more noise. It might worth testing the performance of reducing the output label dimension by PCA to see whether the model prediction result will change.

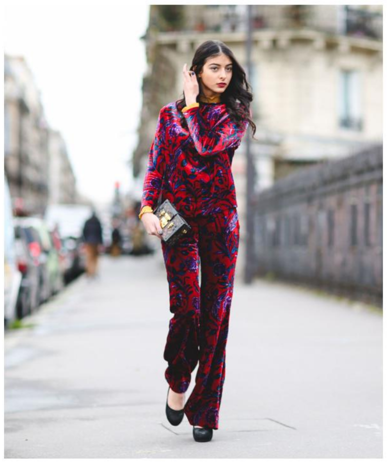


Figure 6: a new test image

6 Conclusion/Future Work

This study preliminary analyzes the power of transfer learning to performance multi-label classification and puts effort to evaluate the model result by F1 score, cost function and accuracy, as well as extracting the recommended images to subjectively evaluating the result. The result complies with the intuition that fine-tuned model works better than original VGG. In the future, it might be worthwhile to explore a more sophisticated method to perform multi-label classification by loss function variation, input data processing especially the output labels fed into the model, and etc, to prioritize the most important output attributes.

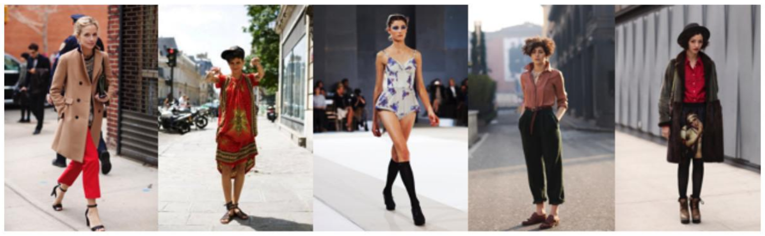


Figure 7: The recommended pictures from my model

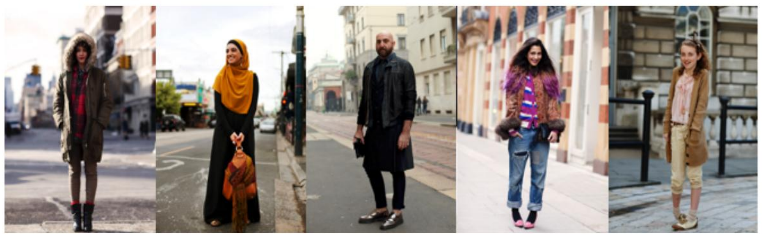


Figure 8: The recommended pictures from original VGG-16 model

7 Contributions

This project is done by only one person, Yiying Hu, from data processing, training infrastructure to evaluating the result. Thank Jay Whang for his helpful feedback during office hours.

Our code is on GitHub at [https://github.com/huyiying/Fashion-DL]

References

Code or libraries: scikit-learn for F1 score, Tensorflow and Keras for building deep neural network and downloading VGG-16

[1] Karen Simonyan: "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014; arXiv:1409.1556.

[2] Han Xiao, Kashif Rasul: "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", 2017; arXiv:1708.07747.

[3] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang: "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations", 2016.

[4] Tong He: "FashionNet: Personalized Outfit Recommendation with Deep Neural Network", 2018; arXiv:1810.02443.

[5] Jiang Wang, Yang song, Thomas Leung, Chuck Rosenberg, Jinbin Wang, James Philbin, Bo Chen: "Learning Fine-grained Image Similarity with Deep Ranking", 2014; arXiv:1404.4661.

[6] Huizhong Chen, Andrew Gallagher, and Bernd Girod, "Describing Clothing by Semantic Attributes", European Conference on Computer Vision (ECCV), October 2012.