CS230

Fashion clothing category classification

Shweta Karwa, Dibyajyoti Ghosh, Abhishek Rawat*
skarwa@stanford.edu, dibghosh@stanford.edu, arawat@stanford.edu

Abstract

In this project, we attempted to solve category classification of clothing items across large number of classes by using Clothing categories and Attributes dataset from DeepFashion [2] for training and validation. SOTA work for fashion uses DeepFashion [2] data along with CNN architectures like VGG-19. In our approach, we used VGG-16 and Resnet-50 architectures along with optimizations to improve performance. We got the best performance number of 75.4% accuracy with Resnet-50 based network. We also investigated visualization in order to analyze the efficacy of our network architecture.

1 Introduction

Online shopping for fashion items is a complex multi-step process. Part of the problem lies in incorrect annotations associated with a particular item like mismatches in style, fabric quality, color, etc. This problem can be solved by automating detailed attribute annotation and categorization of clothing items with high accuracy. The state of art approach for solving this problem uses Attention Network [2] based grammar for landmarks, which further refines the category classification along with attribute prediction.

We propose to achieve the performance as the state of art through experimentation with some popular base networks like VGG-16 and Resnet-v3,v4. The dataset we are using is called DeepFashion [2] which has over 290,000 images with rich attributes, landmark and category annotation.

2 Related work

After looking at various computer vision literature [1, 2, 3] in this space it becomes exceedingly clear that each of these problem areas can be modeled as various stages of a deep learning pipeline. In the paper titled "Attentive fashion grammar network" [1] authors have coded a novel deep learning pipeline for fashion related items. The present solution has attempted to address the problem with attention and grammar based networks, using various intuitive relationships between attributes. In the deepfashion paper, the authors have used clothing landmark and attribute prediction and further pool them to generate category classification prediction. The baseline network used in their FashioNet architecture is VGG-16.

3 Dataset and Features

The DeepFashion dataset [2] has around 290,000 clothing images. Each image is annotated with categories, attributes, bounding boxes and landmarks. Following is how the dataset has been split for training and testing:

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Total data samples = 289222
Training data samples = 209222
Test data samples = 40000
Validation data samples = 40000

3.1 Categories

There are 46 categories in the dataset like dress, tee shirt, coats, shorts, etc. Each category is of one of the 3 types: upper body clothing, lower body clothing and full body clothing. Category prediction is treated as a 1-of-K classification problem. Analysis of the training data (as can be seen from the histograms below) show that the there is immense class imbalance in the data. In order to address this data imbalance we randomly chose 600 images of each category for training. We also considered creating a model only for upper body garments. We also sub-sampled test and validation datasets 10000 each.

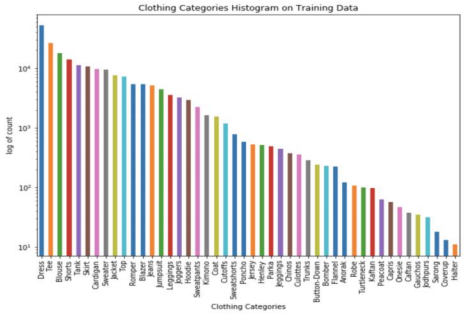


Figure 1: Clothing Categories Histogram on Training Data

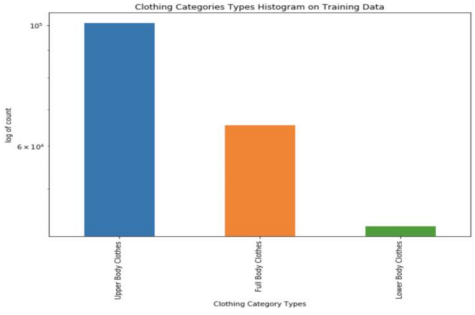


Figure 2: Clothing Category Types Histogram on Training Data

Note that the Y-axis in both category histogram and category type histogram above is log of the value.

3.2 Attributes

There are 1000 attributes that are annotated in the dataset like backless, animal print, knit, faux leather etc. These attribute are of 5 types: texture-related attributes, fabric-related attributes, shape-related attributes, part-related attributes, style-related attributes. Attribute prediction is treated as a multi-label tagging problem.

4 Methods

In order to solve the classification problem baseline VGG-16 model was used, as it is the base model used in DeepFashion [2] paper. We used pre-trained weights for VGG-16 and collected loss and accuracy numbers. We used categorical cross-entropy as a loss function as classification is done over 46 categories of clothing items. After VGG-16 numbers with pretrained weights were in line with DeepFashion [2] reference, we graduated to ResNet-50 as our baseline model. ResNet-50 network is qualitatively better than VGG-16 as it approximates more complex functions, because of its feed-forward branch and deeper architecture, which was a need given highly nuanced features of fashion items images determining categories.

The equation for categorical cross entropy is in Figure 3, where N is the number of training samples, C is the number of classes. Yi is 1, when it belongs to class Cc and p is the prediction probability of sample Yi to be in class Cc,

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}_{y_i \in C_c}\log p_{model}[y_i \in C_c]$$

Figure 3: Categorical Cross Entropy Loss

4.1 Implementation

Our end-to-end network is a modified version of Keras implementation of VGG-16 [3] and ResNet-50 [4]. This implementation includes code for Imagenet pretrained base models in Keras [5]. Our implementation is available at: https://github.com/souschefistry/cs230-midterm-proj-review. The most important modifications that we made were:

Gradient Clipping: We experimented with gradient clipping (clipnorm threshold = 0.5) for during a ResNet-50 finetuning run with last two residual block weights unfrozen.

Regularization: We added L_2 regularization for category embeddings both at kernel level (applied on weights) and activity level (applied on outputs) for fine tuning run on ResNet-50.

Early stopping: We measure validation accuracy for each task separately for each epoch, and use the test results from the best validation epoch for each task.

Learning rate decay: Our implementation uses step decay scheduler in Keras [6] to drop learning rate by a factor every few epochs. Mathematically it can be written as $lr = lr0 * drop^{floor(epoch/epochs_drop)}$.

4.2 Selecting the default values

Given the large number of hyperparameters (Non-trainable params: 53,120) for the network, it was useful to establish default values to use as the basis for experimentation. For the architecture, we set initial learning rate $lr0$ to 0.01 by borrowing recommendations from here [7] with lr dropping by a factor of 0.5 every 10 epochs. For a couple of runs we applied activity and kernel L_2 regularization values to 0.01 which are Keras defaults. As per our observation this didn't help in improving model performance, infact it had opposite effect. While it's difficult to capture exact interplays between various model hyperparameters one intuitive explanation for the drop was that kernel regularizer in Keras which applies to weights along with activity regularizer that applies only to output in last layer causes unintended damping of activations. Uneven distribution of available training samples per fashion category makes it even harder for the network to learn meaningful embeddings for all categories.

5 Experiments/Results/Discussion

This section gives the details of the results of our experiments with VGG-16 and Resnet-50.We used combination of offline / online training for VGG-16 baseline [3]. Learned category embeddings from bottleneck are saved on disk for all training samples and was used later for category prediction on validation / test dataset with RMSProp optimizer. While this works well for shallow networks with smaller dataset we could not scale it for larger training samples in VGG-16; ResNet-50 offline training was very slow along and disk space became a concern. Our ResNet-50 baseline [4] and all finetuning presented here are based on our online training implementation.

5.1 ResNet50 finetuning

One of the key finetuning hyperparameter we have played with is that which layers to freeze and which ones to finetune. Following what is described as residual learning [8] in the path breaking paper we finetuned by residual blocks rather than by going layer by layer. As per our results indicate 3 residual blocks retraining helped the most, gave better accuracy in predicting compared to 1 or 2 residual block(s) fine tuning over. L2 regularization didn't help with or without learning rate decay. We tried some popular optimizers e.g. SGD, RMSProp and Adam. Adam gives us the best result on the dataset. SGD run data haven't been reported since our experiment runs didn't converge.

Table 1: End-to-end ResNet-50 finetuning results.

Task	Optimizer	learning rate	epochs	Test Accuracy	Test Loss
1: ResNet50 finetuned after 153 layers	Adam	0.001	80	59.3536	1.49
2: ResNet50 finetuned after 153 layers + gradient clipping + l2 regularizer	Adam	0.001, 0.5, 0.01, 0.01	50	64.0125	1.68
3: ResNet50 finetuned after 143 layers + early stopping	Adam	0.001, 10	100	75.498	0.95
4: ResNet50 finetuned after 143 layers + early stopping	SGD	0.0001, 0.9	100	-	-
5: ResNet50 finetuned after 143 layers	RMSProp	2e-3, 10	100	66.3	1.69
6: ResNet50 finetuned after 143 layers	Adam	0.001	100	**82.1819**	**0.64**

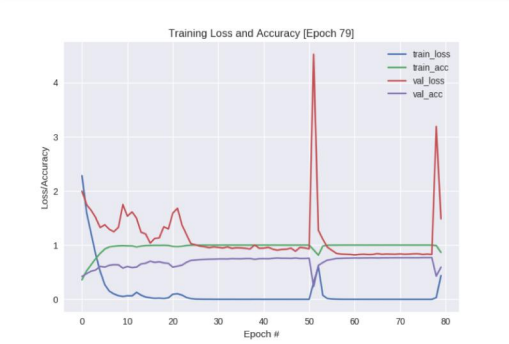


Figure 4: ResNet50 2 res blocks finetuned

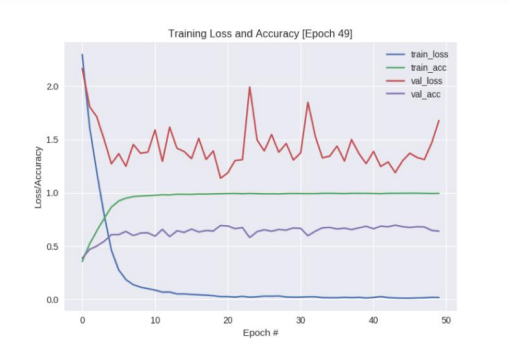


Figure 5: ResNet50 2 res blocks finetuned with L2 + grad clip

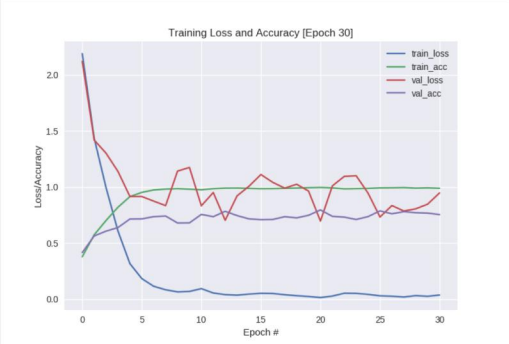


Figure 6: ResNet50 3 res blocks finetuned with early stopping

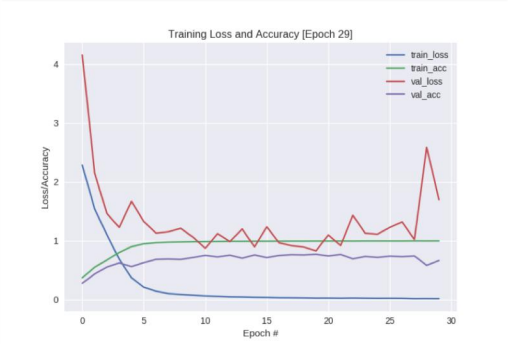


Figure 7: ResNet50 3 res blocks finetuned (RMSProp)

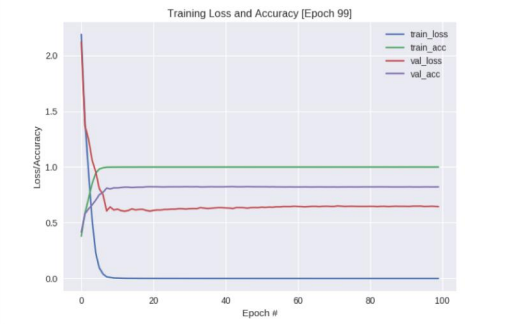


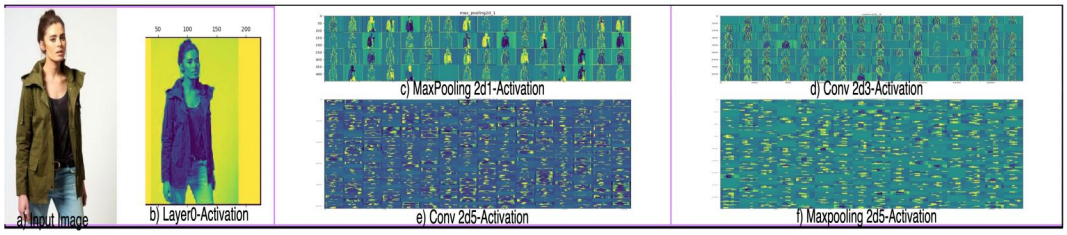
Figure 8: ResNet50 3 res blocks finetuned

6 Visualization

This section includes visualization of two kinds, a) Intermediate Activation Visualization, and b) Heat map of class activations [11].

6.1 Intermediate Activation Visualization

Activation Visualization tells us how various layers transforms an image input based on the function learned by activations at each layers. In the earlier layers of VGG-16 edge detection is done primarily, as seen in the diagram. In later layer, the activations learn more visually complex characteristics e.g. sleeve, neckline etc. with respect to the clothing item. Additionally we also see that the later layers have more sparsity in activations as visible by the all black segments in the activations map. In earlier layers, there are edges to be detected, some edge being detected. However in later layers the characteristics being detected may not be present in the part of the image where we see a black patch.



Visualization of Activations for VGG-16. a) Sample input image from DeepFashion, b) Activations with padding and image augmentation, c) #1 maxpool activations, d) #3 ConvActivations, e) #5 Conv Activations, f) #5 MaxPool Activations

Figure 9: Visualization of intermediate Activations

6.2 Heat-map of Class Activations

This visualization illustrates the heat-map of the class activations over input images. It is based on Class Activation Mapping (implementation based on Grad-CAM[13]), that highlights the important regions in the image and their contribution towards predicting a concept class. The approach computes the gradient of the input images with respect to the final convolution layer

Based on our analysis, we can see that there is marked difference in terms of the correctness of the weights of image segments in predicting the class. In the successful case, where the clothing category is correctly classified, the activations heat-map seem to be focusing on the intuitively better segment of the image. In the activation heat-maps below (Figure 9), we can see that (a), (c), (d) and (e), the predictions are as expected and seem to be focusing on the segments which define that clothing class. However in (b), a 'Jacket' is incorrectly classified as a 'TrenchCoat' and in (f) the 'Bomber-Jacket' is not classified at all.

VGG-16. Highest Predicted class with probability are: (a) TrenchCoat: 99.5%, (b) Trenchcoat: 26.5%, (c) Jeans: 58.8%, (d) Cardigan: 37.4%, (e) Suit: 81.2%, (f) Jeans: 44.6%

Figure 10: Visualization of Heatmap Activations

6.3 Visualization Analysis

A visual analysis of the heat-maps of activations illustrates that the discriminative segments on the clothing items are mainly at bust, waist, seams etc. In Figure 9(b), the missing belt and shorter length is not paid attention to. Additionally, in all these images some of the segments or specifically the clothing landmarks like sleeves, length of lower clothing item seem to be missing. For example in Figure 9(c), the item could have been a 'Jeans-Short', but there is no activation owing to the length. In Figure 9(f), there is hardly any activation for the upper body items, which points to spatial attention not being applied adequately. This analysis points to two missing techniques from our approach a) clothing landmarks features, b) attention mechanism.

7 Conclusion/Future Work

The key learnings of this work are threefold. Firstly our results confirmed how a deeper network which transfers activation across layers is better at learning visual concepts. Secondly, when you have lots of training data you can afford to go deeper in the network with unfrozen weights. DeepFashion clothing attributes dataset has 4000 images per class with uneven distribution; some classes have 0 or less than 100 images per class. This is quite less for end to end training from scratch. That's where transfer learning becomes vital. Training more layers allow pretrained model to learn better feature specific to the type of images it is looking at. Thirdly our visualization experiments indicated how the lack of visual attention and landmark inputs, leads to lack of accuracy in classifying the clothing items.

One of the possible next steps would be apply visual attention mechanism for category classification. Fine grained categorization for clothing attributes would be a good direction to explore. Based on our learning we would use visualization to do manual analysis of our future approaches.

8 Contributions

All team members were involved throughout all the phases of the project and discussions. Data analysis, dataset cleaning up and preparation was mainly done by Shweta. Experiments and model training were mainly done by Dibyajyoti. Visualization was done by Abhishek. Logistics, reports and poster creation were mainly handled by Shweta and Abhishek.

References

[1] Attentive Fashion Grammar Network for Fashion Landmark, Wenguan Wang, Yuanlu Xu , Jianbing Shen, and Song-Chun Zhu

[2] DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations, Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang, June 2016

[3] Keras VGG16: https://keras.io/applications/vgg16

[4] Keras ResNet50:https://keras.io/applications/resnet50

[5] Imagenet: http://www.image-net.org/

[6] Rate decay scheduler: https://keras.io/callbacks/learningratescheduler

[7] Practical Recommendations for Gradient-Based Training of Deep Architectures, Yoshua Bengio, 2012

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. (2015) Deep Residual Learning for Image Recognition, 2015

[9] Learning rate decay explanation and code: https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

[10] Visualization: https://github.com/keras-team/keras/blob/master/examples/`deep_dream`.py

[10] ResNet-50 code: https://github.com/keras-team/keras/blob/master/examples/`cifar10_resnet`.py

[11] Visualization: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/5.4-visualizing-what-convnets-learn.ipynb

[12] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. (https://arxiv.org/abs/1610.02391)

Appendix

8.1 VGG16 baseline results

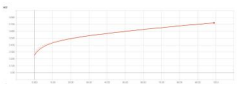


Figure 11: VGG16 baseline train accuracy

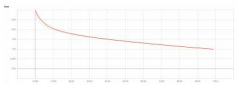


Figure 12: VGG16 baseline train loss

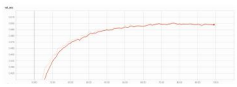


Figure 13: VGG16 baseline validation accuracy

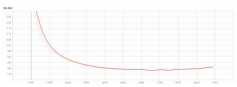


Figure 14: VGG16 baseline validation loss

8.2 ResNet50 baseline res

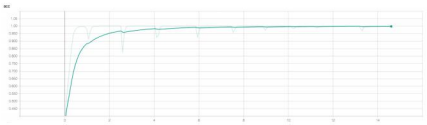


Figure 15: ResNet50 baseline train accuracy

Figure 16: ResNet50 baseline train loss

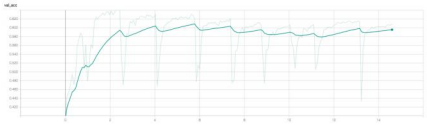


Figure 17: ResNet50 baseline validation accuracy

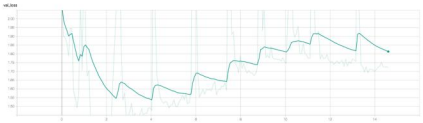


Figure 18: ResNet50 baseline validation loss