Apparel Classification using CNNs

Rohit Patki ICME Stanford University rpatki@stanford.edu

Abstract

Apparel classification from images finds practical applications in e-commerce and online advertising. This report describes our approach at classifying an apparel from a given image using convolutional neural networks which have been proven to perform remarkably well in field of image recognition. The later part of report describes the models we experimented with as well as the results we obtained. Lastly, we perform error analysis and discuss future work.

1. Introduction and Motivation

Fashion industry is always evolving and it is important to keep up with the latest trends. For example, if often happens that we like a particular type of apparel or clothing while watching a TV show. In such situations, one wants to know where they can buy a similar piece. With our project, we aim to lay the groundwork to facilitate such a system by which we can provide a set of similar apparels available for online purchase.

This requires us to be first able to classify a clothing with high precision. This task has its own challenges because very often the clothes are deformed, folded in weird manner and not exactly stretched to reveal its actual shape. If the picture only contains a clothing but no person wearing it, it can be hard even for a human to classify it accurately. Also, the pictures are not always taken from the front and this variation of angle can also add significant difficulty. We believe that with a good amount of data with many such variations, CNNs will do a good job at learning the features most indicative of their respective classes.

1.1. Problem Statement

The final aim of this project is to be able to start with an image containing one or more clothing items which may or may not be worn and be able to give a list of similar clothing items available to buy online. To achieve this, the problem statement can be broken into 4 sub-problems: Suhas Suresha ICME Stanford University suhas17@stanford.edu

- 1. **Style Classification**: Given images of apparels, we try to classify them into different classes (Example: shirt, blouse, undergarment etc.). For this particular problem, we assume that the input images are already cropped and one image contains only one clothing item. This input image is then passed through the CNN to generate one of the labels as the output.
- 2. Attribute Classification: We wish to identify clothing attributes in images. For example, given an image of a shirt, we wish to identify attributes like color, design, sleeve length etc. We will be looking at multilabeled images, where the labels identify the multiple attributes present in the clothing image. Here also, we would have liked to work with cropped images of single clothing items but no such labeled dataset was available. The data we work with has images which contain full images of people wearing clothes. The attributes generated from CNN thus will refer to multiple clothing items together.

After these two, it is important to be able to segment out clothing items from any image and search for closest items in a given database. We won't be addressing these two problems in this project but they are part of our future work.

2. Related Work

2.1. Style Classification

Lukas Bossard *et. al* [1] worked on the same dataset as ours to classify the images into clothing categories. Their focus was mainly to use **Random Forests**, **SVM** and **Transfer Forests** for the task. The core of their method consists of multiclass learner based on random forests that uses various discriminative learners as decision nodes. Their SVM baseline has an accuracy of 35.07 % and the best transfer forest model obtained an accuracy of 41.3 %.

3. Methods and Models

3.1. Style Classification

For this part, the input is a cropped image of a clothing item belonging to one of the following categories:

- 1. Blouses
- 2. Cloak
- 3. Coat
- 4. Jacket
- 5. Long Dress
- 6. Polo shirt or Sport shirt
- 7. Robe
- 8. Shirt
- 9. Short Dress
- 10. Suit, Suit of clothes
- 11. Sweater
- 12. Jersey, T-shirt
- 13. Undergarment, Upper body
- 14. Uniform
- 15. Vest, waist-coat

This image is first converted into a **numpy array** which stores individual pixel values. For example, if we choose a resolution of 64×64 , the input will have shape $64 \times 64 \times 3$ where the 3 refers to the **RGB** pixel values.

3.1.1 Baseline

As a baseline model, we used the convolutional neural network built during the second homework to predict the style of the clothing items. A batch of 32.64×64 inputs is passed through a convolution layer of 32 filters of size 7×7 followed by batch normalization, *relu* activation to bring in non-linearity and max-pooling over 2×2 region. The output from this is again passed into a exact same series of layers once again. Later, the output from that is passed into a dense fully connected layer of size 200 before going through another set of batch normalization and *relu* activation. The last set of layers consist of another dense layer of size 15 and a softmax layer which converts the outputs to probability scores for each class.

While training, the model weights are updated by backpropogation so as to reduce the **softmax loss** at each iteration. To achieve good performance on training set, the model was trained on 40 epochs. While testing, a forward pass is implemented on an image input and the label with highest score is predicted to be its clothing category. In summary, the architecture looks like:

[conv- batch - relu - 2x2 max pool]*2
- affine - batch - relu - affine softmax

A test accuracy of 31% was obtained and set as our baseline for future experiments.

3.1.2 Other models

Going forward, we first decided to assess the performance of one of the popular network architectures in image recognition. We implemented **VGG16**, the code for which was readily available in **Keras**. Also, instead of using 64×64 resolution, we opted for 128×128 .



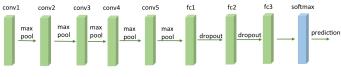


Figure 1. VGG 16 architecture

This model did slightly better than our baseline model in training but not any better on the test and validation accuracy.

Hence, instead of using the popular network architectures, we decided to modify and experiment on our baseline model. We started with a deeper version of our baseline with 5 convolutional layers instead of 2. The results seemed promising as the performance increased by 6% on validation set and around 5% on the test set. Next, we decided to perform hyper-parameter search on the same network. We could only do trial and error because the training takes around 12 hours and so grid search seemed infeasible.

While doing the hyper-parameter search we realized that the training accuracy never went over 55%. Hence we tried a deeper network with lesser dropout and pooling and **Adadelta** optimizer. We thought adadelta might be useful because it the fastest to reach near convergence even if it might take a long time to converge. This proved to be correct and we could observe a higher training accuracy and a slight improvement in validation and test accuracy.

In summary, the best performing model was:

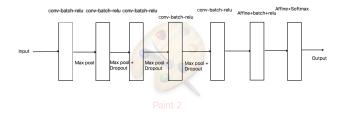


Figure 2. Best performing net architecture

3.2. Attribute Classification

In the **Clothing attribute** dataset, each image had at most 23 clothing attribute labels. For some of the images, certain attributes were not available because of the ambiguity among the human workers classifying them.

The images in the **Clothing attribute** dataset was also in the format of jpgs. For the classification, we only considered attributes having binary classes (most of them being a yes/no classification). We have 23 such attributes, which included 11 colors, 5 clothing patters, scarf and collar identification. To account for the varying image sizes, we decided to resize them to 100×100 images for the baseline test.

3.2.1 Baseline

For the baseline case, we only considered five attributes: **gender, necktie, skin exposure, wear scarf and collar**. As mentioned before, for many of the images certain attributes were not available because of the ambiguity among the human workers classifying them. Hence we have a varying number of attributes for each of the images. In order to account for the varying number of attributes in different images, we trained a different neural network for each of the chosen attributes. For each of the networks, we only chose the images having the corresponding attribute for the training set. For all the cases, we trained a simple 3-layer CNN. The CNN used was:

conv - relu - 2x2 max pool - affine relu - affine - softmax

We obtained a mean accuracy of 70% for the attributes considered in the baseline case. This was not really impressive considering the fact the labels were binary and the data set was unbalanced in most of the categories. The accuracy we obtained was not much higher than the guess accuracy for most of the selected attributes.

3.2.2 Other models

Going forward, we decided to build a single multi-label binary classification CNN instead of training multiple neural networks. This was a much faster and robust approach and gave us much better results compared to the baseline case. In order to account for the missing labels in the training set, we randomly assigned them one of the 2 classes (all being binary labels). This approach is acceptable considering that the fact that humans had difficulty identifying the classes themselves. For this step, we considered all the 23 binarylabel attributes.

We trained a CNN using **Keras** with **Theano** for doing multi-label classification. The architecture we used was:

[conv-batchnorm-relu-pool]*1
[affine-relu-pool]*2-affine-sigmoid

We used 1 convolution layer (with batchnorm, relu and dropout layers included) and 2 affine layers. We used a sigmoid activation fucntion instead of a softmax activation (which we used in the baseline) as we found much better accuracy. This is to be expected as all our labels have binary classes. We used binary crossentropy loss and Adam optimizer to train the CNN. We performed hyper-parameter tuning to improve the obtain the best model.

4. Dataset

4.1. Style Classification

We used the **Apparel Classification with Style** [1] dataset for this part. In total, it had 89,484 images split in 15 different clothing style categories listed in the previous section. We split this dataset into 3 parts for training, validation and testing.

Data	Images
Training	71,140
Validation	9,172
Testing	9,172

We shuffle the data first before splitting. This helps to maximize variety in the training examples.

The data has	.1 C 11	• • • •	.1	
The data hac	the tollow	una colit	among th	10 cotocorios
The uata has	the ronov	ving spin	amone u	ic calleones.

Label	Images	Label	Images
Long Dress	12,622	Sweater	6,515
Coat	11,338	Short dress	5,360
Jacket	11,719	Shirt	1,784
Cloak	9,371	T-shirt	1,784
Robe	7,262	Blouses	1,121
Suit	7,573	Vest	938
Undergarment	6,927	Polo shirt	976
Uniform	4,194		

As the dataset is decently sized, we did not find the need to perform any kind of augmentation.



Figure 3. Example of a jersey in the data set



Figure 4. Example of a long dress in the data set

4.2. Attribute Classification

We used the **Clothing Attributes** dataset for attribute classification. The dataset contains 1856 images, with each of the images having up to 26 ground truth clothing attributes. The statistics of the clothing attribute dataset is shown in the table.

Among the 26 attributes, we have 23 attributes with binary classes (Yes/No or Positive/Negative). We are going to identify only the binary-class attributes using our CNN. Since we have only 1856 images, we decided to augment the training data by performing rotation, vertical and horizontal shift, and flipping the images. Also, since some of the attributes were unbalanced in the data-set, we decided to augment the images having the lower-frequency labels.

We performed pre-processing on the data by applying ZCA whitening and normalization. As we mentioned before, certain attributes were not available for some of the images because of the ambiguity among the human workers classifying them. In order to account for the missing labels in the training set, we randomly assigned them one of the 2 classes (all being binary labels).

Clothing pattern	Solid (1052 / 441)	
(Positive/Negative)	Floral (69 / 1649)	
(1 Oshive/1 (egalive)	Spotted (101 / 1619)	
	Plaid (105 / 1635)	
	Striped (140 / 1534)	
Malan salan	Graphics (110 / 1668)	
Major color	Red (93 / 1651)	
(Positive / Negative)	Yellow (67 / 1677)	
	Green (83 / 1661), Cyan (90 / 1654)	
	Blue (150 / 1594), Purple (77 / 1667)	
	Brown (168 / 1576), White (466 / 1278)	
	Gray (345 / 1399), Black (620 / 1124)	
	Many Colors (203 / 1541)	
Wearing necktie	Yes 211, No 1528	
Collar presence	Yes 895, No 567	
Gender	Male 762, Female 1032	
Wearing scarf	Yes 234, No 1432	
Skin exposure	High 193, Low 1497	
Placket presence	Yes 1159, No 624	
Sleeve length	No sleeve (188), Short sleeve (323)	
	Long sleeve (1270)	
Neckline shape	V-shape (626), Round (465)	
	Others (223)	
Clothing category	Shirt (134), Sweater (88)	
	T-shirt (108), Outerwear (220)	
	Suit (232), Tank Top (62)	
	Dress (260)	

5. Experiments and Results

Here, we present the results of experiments we carried out and compare them with results from previous work.

5.1. Style Classification

Our best performing model correctly classified **41.1%** of the images in the test set. The training accuracy and validation accuracy with epochs is shown below:

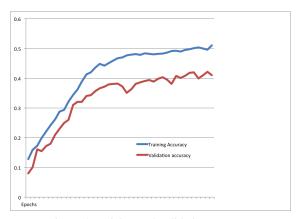


Figure 5. Training and Validation accuracy

We can see that the training performance plateaus around 51%. We tried deeper networks to increase the training performance. It did improve till 71% but validation accuracy was much lower.

The below plot shows the performance on each label:

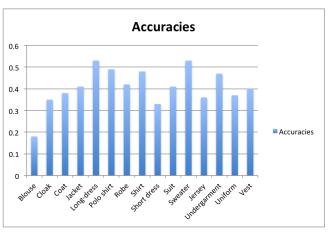


Figure 6. Accuracies for each label

Next, we compare our results for the Apparel Classification problem with Bossard and Dantone's [1] work with SVM, Random Forest and Transfer Forest:

Model	Test Accuracy
Our Baseline	0.311
SVM	0.350
Random Forest	0.383
CNN	0.411
Transfer Forest	0.413

Clearly, our CNN shows a lot of promise. It gives 41.1% accuracy even when the training accuracy isn't that great. With deeper networks and little more regularization, a higher accuracy should be achievable. We could not experiment further because training 40 epochs takes around 12 hours of computing and that gave us very little time to try out different models.

5.2. Attribute Classification

Our best performing model gave an average test accuracy of 84.35% among all the attributes. Since we have unbalanced data set, we have to compare the accuracy to the unbalanced percentage (Percentage of highest frequency label) for each of the attributes in order to truly judge how good our predictions were. The accuracies are compared with the unbalanced percentage in the following table.

Attribute	Test Accuracy %	Unbalanced %
Black	73.04	66.5
Blue	93.35	85.6
Brown	87.10	85.18
Collar	71.09	60.21
Cyan	93.75	91.8
Gender	64.84	55.78
Gray	71.48	75.62
Green	94.53	89.7
Many Colors	85.54	83.29
necktie	87.10	82.59
pattern floral	91.79	89.13
pattern graphics	97.26	92.66
pattern plaid	92.18	90.83
pattern solid	77.73	56.86
pattern spot	91.0	89.94
pattern stripe	85.15	85.22
placket	71.48	62.64
purple	93.35	90.01
red	90.62	89.24
scarf	73.82	75.10
skin exposure	81.64	79.50
white	77.34	70.21
yellow	94.92	90.64

From the table, we see that for 20 of the 23 attributes, we get higher accuracies compared to the unbalanced percentage. For 3 attributes (scarf, plaid pattern and Gray), we get lower accuracy than guess work. Since we have one CNN model trying to capture all 23 attributes, we are bound to get lower accuracy than unbalanced percentage for one or two of the attributes. On an average, we can see that our model identifies the attributes with a good accuracy. The plot below shows the accuracy for each of the attributes. Our model performance compares quite well with the results obtained by Chen and Girod [2]. They obtain a mean accuracy of around 85% compared to our mean accuracy of 84.35 %.

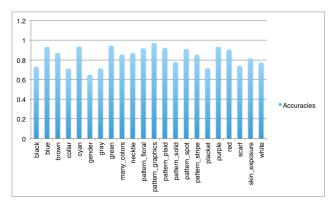


Figure 7. Accuracies for each attribute

6. Discussion and Analysis

6.1. Style Classification

Style classification can be challenging even for humans. The images in the dataset often have clothes worn by humans and the picture is taken at an angle which hides important features required to make prediction on the style. Also, sometimes the image is cropped such that the whole of the clothing item is not visible in the image. We, humans are able to recognize these but to a computer it is hard.

Another important fact is that there is no sharp distinction between some clothing style categories. For example, there is a good overlap between Sweater and a T-shirt. The below image shows a Sweater which was misclassified as a T-shirt



Figure 8. Example of a sweater misclassified as T-shirt

Below is another example of our model being confused about the label:



Figure 9. Example of a shirt misclassified as jacket

We can see that the way in which the shirt is kept makes is more similar to a jacket. These kind of issues come up quite frequently. Out of curiosity, we found out the top 2 accuracy of our best performing model to be **68.9%** which is decent in our opinion.

There are times when the model does remarkably well in identifying correct classes even when the clothing item is not full visible in the picture. Below is one such example.



Figure 10. Example of a long dress classified correctly even though it is not fully visible

6.2. Attribute Classification

We had to make the CNN model learn about 23 different attributes from just 1856 images, which is obviously a challenging task. The fact that some of the attributes had unbalanced classes made the task further challenging.

The model does well in identifying most of the attributes correctly. It does misclassify certain attributes either due to the unbalanced nature of our training set or due to certain confusing characteristics of the image. For example, it misclassifies Steve Jobs as a female probably due to the female class being unbalanced in the original data set. It does correctly identify certain attributes like black, blue, no collar and no skin exposure.



Figure 11. Attributes identified: Black, Blue, Female, solid pattern, no skin exposure, no collar, floral pattern

In Figure 12, we see that the model has misclassified color grey. This could be because the model identifies the shadows in the image, a confusing characteristic of the image. The model correctly identifies that there is a scarf and collar, which is pretty good. In figure 13, we see that the model has correctly identified the attributes being female and skin exposure. It does misclassify color as purple because pink was not among the color attributes.



Figure 12. Attributes identified: Black, Grey, Collar, Male, Solid pattern, Scarf, No skin exposure



Figure 13. Attributes identified: Female, skin exposure, purple

7. Future work

- Identifying similar apparels: Given a clothing image, identifying images having similar clothing from a large set of images.
- Object detection: Given an image, identifying the set of regions within the image that contain apparel objects.

8. Conclusion

To summarize, we successfully implemented CNNs to perform the task of apparel classification and attribute classification. The results obtained were decent and show great potential of doing better with higher resolution images and more sophisticated neural networks.

9. References

References

- Bossard, Dantone, Leistner, Wengert, Quack, and V. Gool. Apparel classification with style, computer visionaccv 2012.
- [2] H. Chen, A. Gallagher, and B. Girod. Describing clothing by semantic attributes. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 609–623, Berlin, Heidelberg, 2012. Springer-Verlag.