Fashion Classification and Detection Using Convolutional Neural Networks

Archit Arora and Prateek Srivastava

December 2, 2016

1 Motivation and Problem Statement

The objective of the project is to identify the apparel a person is wearing in a given image. Applications for this project are primarily in ecommerce [1] for providing targeted ads and product recommendations. Some of the other applications include identification of suspects through CCTV footage [2] and studying brand adoption trends for clothing companies via social media network data. Before describing our approach, we first discuss some of the challenges that exist in image classification problems. These challenges occur mainly due to different kinds of variations in images. These variations include intra-class variation, viewpoint and locational variation, scale variation and background clutter (refer to Fig. 1).

(a) Intra-class variation:
Images differ within a particular class

(b) Viewpoint and locational variation:
Different angles and orientations

(c) Scale variation:
Images zoomed in and afar

(d) Background clutter:
Images have different environments

Figure 1: Different types of variations in images

2 Methodology

The process of fashion classification can be summarized in three steps: (1) Data Collection, (2) Data Augmentation, (3) Learning from data. We explain these steps in greater detail in the following sections.

2.1 Data Collection

In order to generate a labeled dataset of images for our fashion classification problem, we used Scrapy to search Macy's website for images corresponding to each fashion category label. High-resolution versions of the images obtained from search results were then obtained by crawling the website and downloading them using the Urllib module. Using this procedure, we created a training dataset consisting of 3500 images belonging to 14 different apparel categories. These categories are listed in Table 1.

2.2 Image augmentation

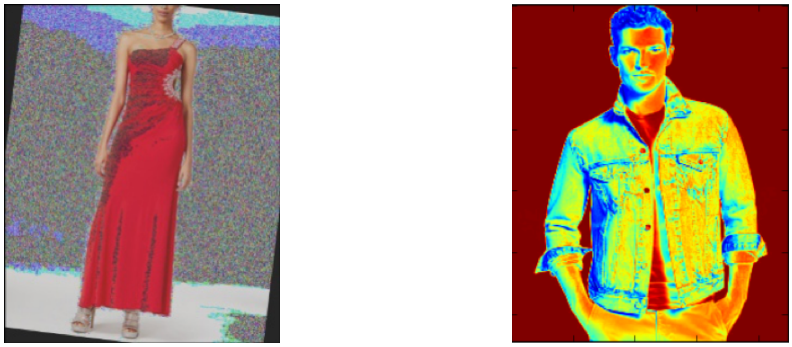
As described in [3], the CNN classifier can be made more robust to different variations described in Section 1 by explicitly incorporating these variations in the training dataset itself. This is because the CNN classifier is

Table 1: Fashion Categories

Jacket	Polo Shirts	Robe	Suit	Short-dress
T-shirt	Trousers	Sweater	Shoes	Long-dress
Shirt	Jeans	Swimsuit	Shorts	

able to learn from the examples in training dataset and adjust itself accordingly to recognize similar variations in the test dataset. In our code, we use Scipy and Scikit-image to introduce a combination of effects (with different probabilities) to each image in our training dataset. These changes included the addition of following effects (shown in Fig. 2) :

1. Background noise - to enable the classifier to recognize the important boundary features
2. Random jitter - to deal with potential noisy images
3. Rotation and contrast changes: to deal with rotational and contrast changes
4. Crop - to tackle scale variances
5. Resizing to tackle resolution variances
6. Image normalization- subtract mean from each RGB channel to center the images



(a) Example of background noise and rotation (b) Example of image normalization

Figure 2: Image augmentation

2.3 Learning from data

We use CNNs to classify the apparels in a given image. The input to CNNs are image volumes, consisting of length and breadth components which correspond to the spatial dimensions of the image and a depth component

Input: 224x224 size images with 3 channels
1. Convolutional (224x224x64)
2. Convolutional (224x224x64)

Max Pool (112x112x64)

3. Convolutional (112x112x128)
4. Convolutional (112x112x128)

Max Pool (56x56x128)

5. Convolutional (56x56x256)
6. Convolutional (56x56x256)
7. Convolutional (56x56x256)

Max Pool (28x28x256)

8. Convolutional (28x28x512)
9. Convolutional (28x28x512)
10. Convolutional (28x28x512)

Max Pool (14x14x512)

11. Convolutional (14x14x512)
12. Convolutional (14x14x512)
13. Convolutional (14x14x512)

Max Pool (7x7x512)

14. Fully Connected (1x1x4096)
15. Fully Connected (1x1x4096)
16. Fully Connected (1x1x1000)

Soft-max (1x1x1000)

Figure 3: Architecture of Modified VGG-16 model- the dotted box represents the modification in last 3 fully connected layers of original VGG-16 model.

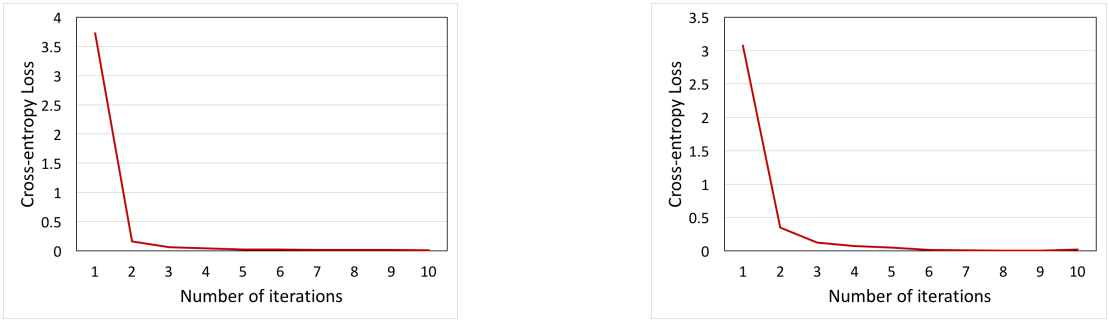
which corresponds to RGB channels. Stacks of convolutional layers, which form the backbone of a CNN, are obtained by sliding or convolving filters of much smaller sizes over the entire image volume. Each filter represents a pattern template that gets activated when it is passed over a volume of the image that resembles the template. Since we want to detect many different patterns in an image, we use many different filters resulting in several layers of activation map or layers.

During the learning process, the weights of the filter are optimized to minimize training error. However, in practice, these weights are only seldom optimized from scratch due to computational considerations. Instead, pre-trained CNN models that have been previously trained on some other dataset is used. This is because many of the low level features (e.g. detection of an edge at a particular orientation) are common to all image classification problems. For our experiments, we used the VGG-16 model [4] pre-trained on the Imagenet classification challenge. We assumed the weights for the first 13 convolutional layers to be fixed and optimized over the weights of the last 3 fully-connected layers of the model (see Fig.3).

3 Results

We implemented the modified VGG-16 CNN architecture using Theano and Lasagne (wrapper for Theano) Python libraries with CUDA GPU on Amazon EC2 server. We defined minimizing the cross-entropy or negative log-likelihood loss function as our objective for the problem. We then optimized over the weights using mini-batch Stochastic Gradient Descent (SGD) with Adam update rule and batch size of 16. We then conducted two different experiments- the first one consisted of test data from Macy's website, while the second consisted of a custom test data from Google Images.

- **Experiment 1- Macy's data with train-test split:** We first considered Macy's data without image augmentation and this was split into training (75%) and test (25%) datasets. This gave an accuracy of around 92% and the solution converged within 5 iterations using Adam update rule as shown in Figure 4a. Some of the results from the test data are shown in Figure 5.



(a) (b)

Figure 4: Convergence plots for Experiments 1 and 2

- **Experiment 2- Macy's augmented data with custom test data:** The test data consisted of around 60 images from Google Images comprising of real world photos. On using the previous model to predict this test data, the accuracy was exceptionally bad (around 20% accuracy). This was because Macy's training dataset was devoid of many variations as discussed in Section 1. Hence, there was a need for image augmentation in the training set.
  We achieved an accuracy of around 71% after image augmentation. In this case too, the solution converged within 5 iterations as shown in Figure 4b. Some of the results from the test data are shown in Figure 6.

4 Conclusions and Future Work

In this project, we developed a framework for creating a custom fashion image dataset and classifying apparels in image dataset using CNNs. We conducted experiments to understand the effect of data augmentation on the prediction performance of a CNN classifer and achieved state-of-the-art results using our model. Future work would be to extend the approach to multiple clothing object detection in images. This can be done by using a sliding window approach with fast linear classifiers to identify regions of interest and apply CNN to those ROIs. Another extension could be determine specific object location in images. For this, a regression head with L2 loss function can be used as modification to VGG-16 model.[3]

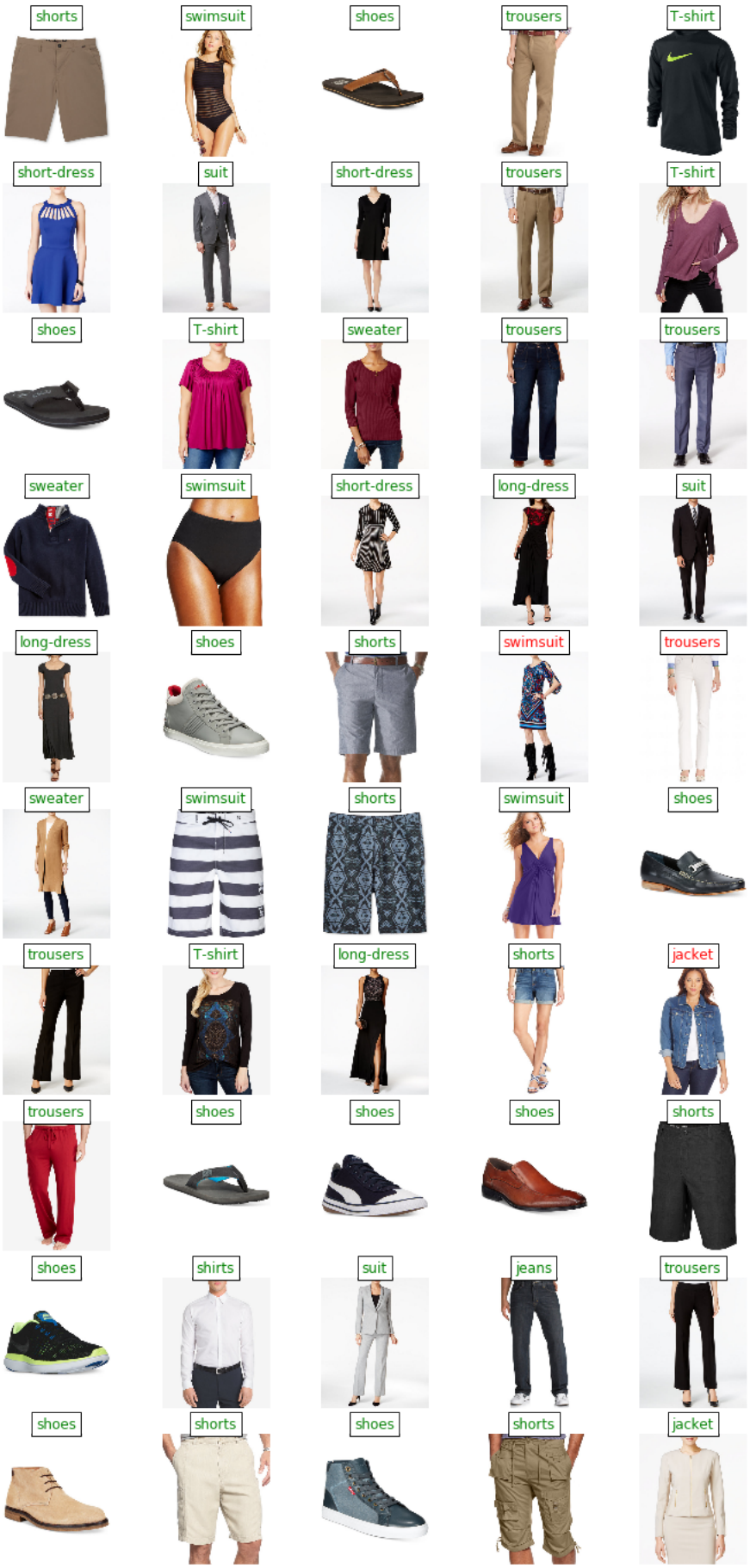


Figure 5: Experiment 1- Some results from test set with predicted labels on top, green denotes correct classification and red denotes error in classification

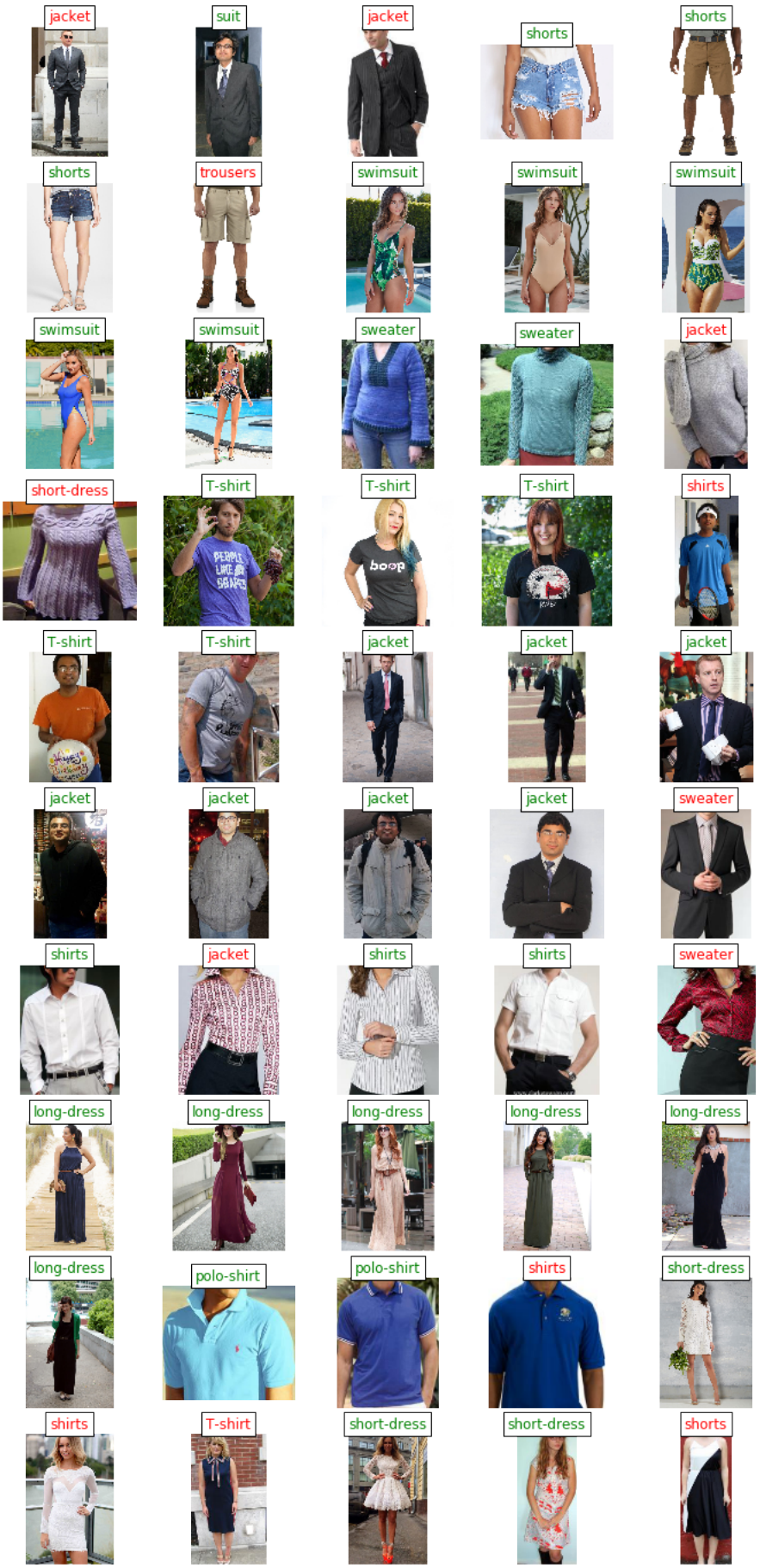


Figure 6: Experiment 2 - Some results from test set with predicted labels on top, green denotes correct classification and red denotes error in classification

References

[1] Si Liu, Zheng Song, Guangcan Liu, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3330–3337. IEEE, 2012.

[2] Ming Yang and Kai Yu. Real-time clothing recognition in surveillance videos. In *2011 18th IEEE International Conference on Image Processing*, pages 2937–2940. IEEE, 2011.

[3] Justin Johnson Fei-Fei Li, Andrej Karpathy. Convolutional neural networks for visual recognition.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.