REPORT

1. Understanding the problem:

* Upon looking at the dataset and reading the description provided it was clear that it was a multiclass classification problem where an image (of a shoe) has to be categorized into one of the types and colors. There were 28 classes for type of shoes and 21 classes for color of the shoe. I decided to training two separate classifiers one for type and another for color.

1. Dataset Curation:

* I relied upon **deep learning** for extracting features from image and for that I had to train the **Neural Network** again and again to see if it is converging or not. That is why I decided to store all the images on the disk rather than fetching them from **url** every time. So this was my first step downloading all the images to disk and save the images for each category to specific folder such that there were 28 folders with their names representing type of shoe.

* During the phase of fetching images from **url** some of the image links were broken and for some images there were no labels. Images with no labels were of no use as I was working with *supervised learning* task of classifying images. So I skipped the images with broken **url** and wrong or no labels.

* After taking care of cleaning the data only 9338 out of 15000 entries were clean i.e. which had labels for types and 5898 images which had their color label correct. For training the **type classifier** I split image dataset into 8000 images for *training* and 1338 for *validation set*. For **color classifier** I decided not to split data as the number of images were already very less. To assess the performance of **color classifier** I used unlabelled images from the dataset provided.

1. Literature Review:

* A considerable amount of time was spent for this phase as the classification task for images is quite hard. Two approaches were finalized based on **Transfer Learning** after reading through all the blogs and papers.

1. **Fine Tuning** final layers of CNN
2. **Feature extraction** from last dense layer of CNN

* Here is the link mentioning both the approaches: <http://cs231n.github.io/transfer-learning/>

1. Design and Approach:

* Firstly, I tried fine tuning the **VGGNet** which is 16 layers deep. I freeze all convolutional layers so that only last dense layers were trainable. The **jupyter notebook** named (**fine\_tuning.ipynb)** can be referred to see the accuracy drops while training. The results were really bad due to less data and redundancy inside the data (5 similar images for each shoe means 15000/5 => 3000 unique images and again some of them have no labels). So afterwards I relied upon the second approach of using the **VGGNet** as **Feature Extractor**.
* A **multiclass SVM classifier** was used to train upon the image features extracted from images as SVMs are proven to show the best results in cases with this under 100 classes. I also used **PCA** for extracting most significant features from the 4096 length vector as more features can sometimes hurt the accuracy of SVM classifiers.
* After all the extraction and transformation, I trained 1024 dimension and 512-dimension vector for type and color classifier respectively. 1024 for type as it needs much more information about the image but color information can be generated using few features.
* The libraries I used for this task were

- **keras** (with tensorflow backend)

- **scikit-learn**

- **PIL** and **cv2** for image manipulation (resizing etc.)

* Hardware:

- Intel i5 4690

- 8GB ram

- nvidia 970 gpu (for fine tuning)

1. Experimentation:

* Almost 70% of my time was spent on this phase. I have done extensive experimentation with number of features to be passed to the classifier, fine tuning the model by freezing different number of layers each time and the *hyper-parameters* to be chosen.
* I experimented with the transformation dimension for **PCA** and trained **SVM** classifier for each of them to find a sweet spot where accuracy is maximum.
* I also used grid search as provided by **sklearn** which is a mechanism through which best parameter values can be found for your classifier. As **grid search** check all the combinations of parameters it took a lot of time even when multithreading (all 4 cores) was used.
* After all the look up I found that 4096-dimension vector performs poorly than 1024 length vector which was obtained after applying **PCA**. So the final classifier that I am using is based on **PCA** transformed vector.
* All the above steps were also applied for color classifier and finally it was trained on 512 length **PCA** transformed vector.
* I was able to get 31% accuracy for 28 classes even on such redundant and unlabelled data.

1. Failures:

* First failure was, unable to fine tune the model even on last layer I tried various combinations of *layer freezing* and *dataset manipulation* but results were very poor
* Second major failure was the accuracy obtained without applying **PCA**. More features must result in better results but in this case the features were irrelevant and were acting like noise. And that is why **PCA** performed really good.
* Some small failures were because of broken **urls** and unlabelled entries.

1. Business Impact:

* According to me such classification task will best suit recommendation problem. An image based **Recommender System** can be constructed which can generate recommendation based on the clicked image. Features can be extracted from the clicked image and its type and color can be predicted. Based on type and color other images lying in these categories can be recommended.
* Second application can be *searching the products* based on the image uploaded by the user. With some image pre-processing on uploaded image its category can be determined and similar products can be searched.

1. Conclusion:

* Image classification is a difficult task but recent researches in **Deep Neural Networks** have made it possible to get state-of-the-art results for such task. **CNNs** have proved to work best for image classification tasks but with less data linear classifiers like **SVM** are still our best option.
* Really good results can be obtained for this task first, if the redundancy in the data can be removed and second if, unlabelled data entries can be labelled.
* More data is always better, collecting more data can increase accuracy up to 90%.
* Also keeping same image five times in a dataset do not help much. It is just like running 5 extra epochs on same data.

1. References:

* Transfer Learning: <http://cs231n.github.io/transfer-learning/>
* Keras Documentation: <https://keras.io/>
* sklearn SVC: <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
* Pretrained weights for VGGNet16: <https://github.com/fchollet/deep-learning-models/releases>
* Fine tuning in keras:<https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>