# Final Project Report

**Ajay RR (IMT2017502) Ronak Doshi (IMT2017523) Ram S (IMT2017521)**

Visual Recognition Course
International Institute of Information Technology Bangalore

## Contents

# 1 Problem, Data, Model and Solution Architecture

## 1.1 Problem Statement

*A camera is placed at the entrance of a retail shop. Your company supplies the video analytics solution to this shop. You are given a snap i.e. a image frame sampled every 5 seconds*
*Assuming people enter one-by-one at the entrance, you have to find if there is a person in every snap, and if so what are they wearing - "Formal Shirt/ T-Shirt/ Saree/ Kurti / None of the above". Build such a system.*

## 1.2 Overview

A dress recognition system was proposed to recognise the dress that a person is wearing in an image. The system was designed to first detect a person in the image and then classify the dress the person is wearing. The person detection was performed using a pre-trained YOLO object detection network and a classifier was trained on the dataset of dresses.

However, the performance of such a system is limited. A 'recognition' system would only be able to recognise one category of dress that a person is wearing. A more effective and robust solution for a retail store would be if the system could detect all the dresses that a person is wearing. Thus, this was also treated as an object detection problem. In other words, each dress is considered an 'object'. A YOLO object detection network was trained on these categories of objects (dresses) and the system was evaluated. A comparative study revealed that the object detection approach provides more accurate, reliable and useful results.

## 1.3 Data

The dataset was created from the 'Deep Fashion' database [3]. DeepFashion is a comprehensive fashion dataset. It contains 491K diverse images of 13 popular clothing categories. It totally has 801K clothing clothing items, where each item in an image is labeled with category, style, bounding box, dense landmarks and per-pixel mask.
We modified the DeepFasion database into 5 simple categories of dresses - 'Jacket', 'Skirt', 'T-Shirt', 'Shorts', and 'Top'. This formed our customised dataset [4]. The custom dataset was balanced to contain a total of 55000 images with 11000 images for each category.

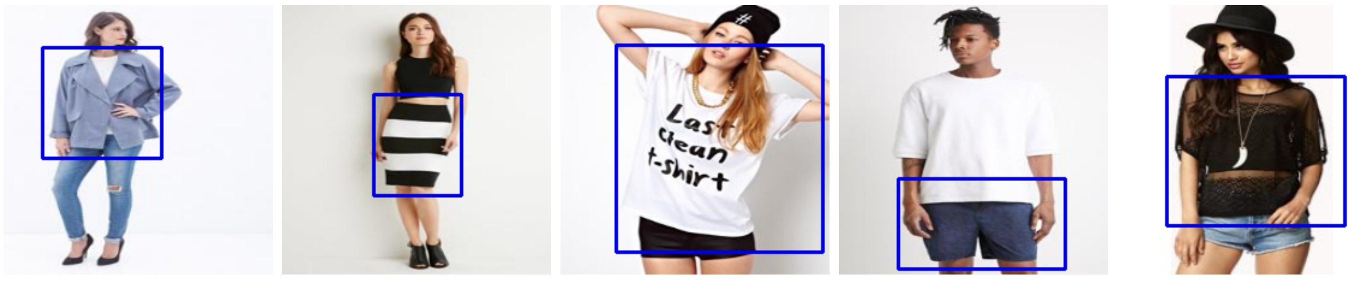


Figure 1: DeepFashion Modified Dataset, (a) Jacket (b) Skirt (c) T-Shirt (d) Shorts (e) Top

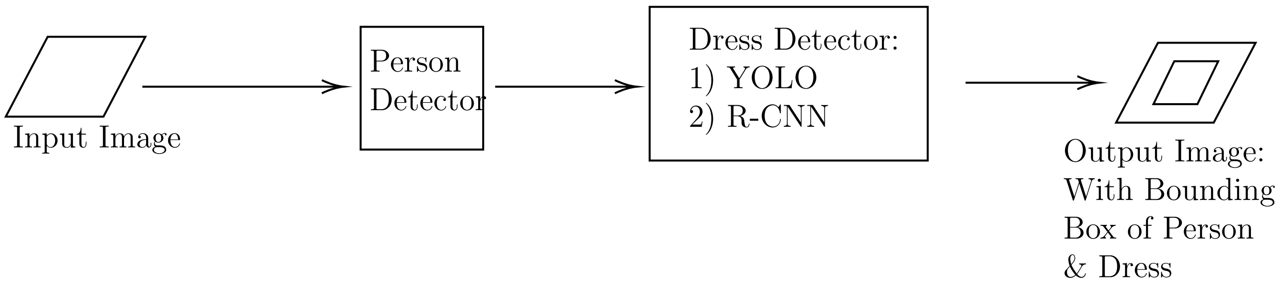
## 1.4 Deep Learning Based Object Detection

Within Deep Learning based Object Detection models, there are two primary object detection models we have used:

1. RCNN and it's variants such as Fast RCNN and Faster RCNN - RCNN's were one of the first deep learning-based object detectors and are an example of a two-stage detector. In the first RCNN publication, *Rich feature hierarchies for accurate object detection and semantic segmentation*, [] it proposed an object detector that required an algorithm such as Selective Search to propose candidate bounding boxes that could contain objects. These regions were then passed into a CNN for classification, ultimately leading to one of the first deep learning-based object detectors.
   The main problem with these detectors was that they were very slow and not a complete end-to-end object detector. So, the same authors published a second paper in 2015, entitled *Fast RCNN*. This had considerable improvements compared to the original, but still relied on an external region proposal algorithm. It wasn't until the follow up paper, *Faster RCNN: Towards Real-Time Object Detection with Region Proposal Networks* that RCNN's became a true end-to-end deep learning object detector by removing the Selctive Search requirement and instead relying on a Region Proposal Network (RPN) that is fully convolutional and could predict the object bounding boxes and "objectness " scores. These outputs would then be passed into the RCNN component for final classification and labelling. More on this topic is elaborated in Section 4.
2. YOLO - You Only Look Once - While RCNN's tend to be very accurate, they are also very slow. To increase the speed of deep learning based object detectors, both SSD's (Single Shot Detectors) and YOLO use a one-stage detector strategy.
   The algorithms treat object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and the corresponding class label probabilities. In general, single stage detectors tend to be much faster, but less accurate than two stage detectors. YOLO itself has gone through multiple iterations and further details will be discussed in Section 3.

## 1.5 Solution Architecture

**Solution Architecture**



Our Solution Architecture consists of two components: Human Detection and Dress Detection. We created a pipeline where an image is first passed through the Human Detection which detects whether a human is present or not. Then, if a human is present, a bounding box for the human is drawn and the image is cropped to showcase only this bounding box. This ensures that the system does not waste time trying to detect dresses when there is no person in the image.

The image is then passed through our dress detector to detect if a particular dress is present in the image. For this particular system we have considered five categories - Jacket, T-Shirt, Shorts, Skirt, Top.

## 2 Person Detection

### 2.1 The Model

Now that we have a general grasp of RCNN and YOLO, to solve the problem of Person Detection, we used the YOLOv3 model trained on the COCO data set. The COCO dataset consists of 80 different labels such as People, Bicycles, Airplanes and many others, however, we are going to use the model only to perform object detection for a Person and get bounding box coordinates.

We use a pre-trained version of this model (on the COCO data set), as we only need to perform Person Detection and there is no real need to train it on our data. The results of running this model are shown in Figure 2.

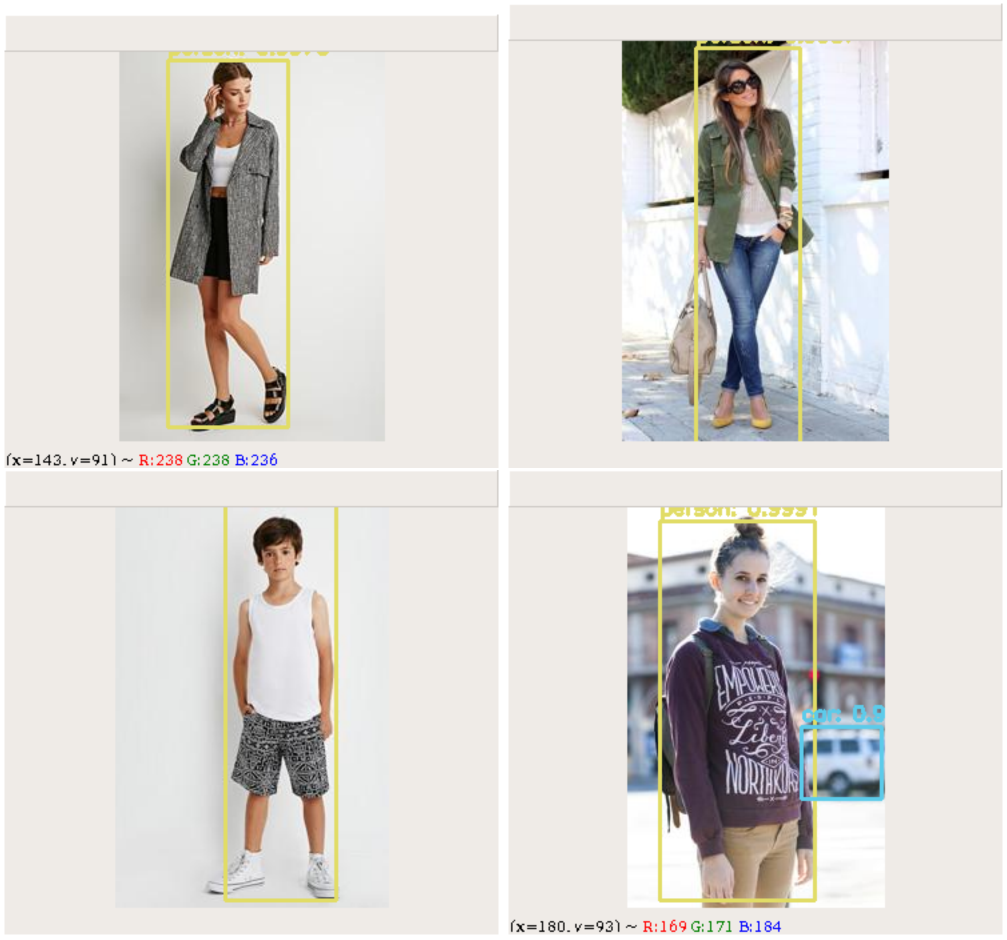

(x=143, y=91) ~ R:238 G:238 B:236
person: 0.9991
car: 0.9
(x=180, y=93) ~ R:169 G:171 B:184

Figure 2: Result of Person Detector

Now that we have identified where the person is present in the image, we simply crop out the image present within the bounding box and feed it to the Dress Detector, the next component of our pipeline. If a person is not detected, then we simply output that no person was detected and hence no dress was detected.

## 3 Dress Recognition

After generating new image data-set by cropping out the image present in the bounding box around the person, we feed this to a classification model.

## 3.1 Model

This classification model (or recognition model) will classify the dress (or image) from either one of the above mentioned classes ('Jacket', 'Skirt', 'T-Shirt', 'Shorts', and 'Top').
We are using an already existing CNN architecture called ResNet-50 [1]. ResNet-50 is a deep residual network. The "50" refers to the number of layers it has. It's a subclass of convolutional neural networks, with ResNet most popularly used for image classification. The main innovation of ResNet is the skip connection. As you know, without adjustments, deep networks often suffer from vanishing gradients. The skip connection allows you to stack additional layers and build a deeper network, offsetting the vanishing gradient by allowing your network to skip through layers if it feels they are less relevant in training.
Architecture of the ResNet-50 can be seen in Figure 3

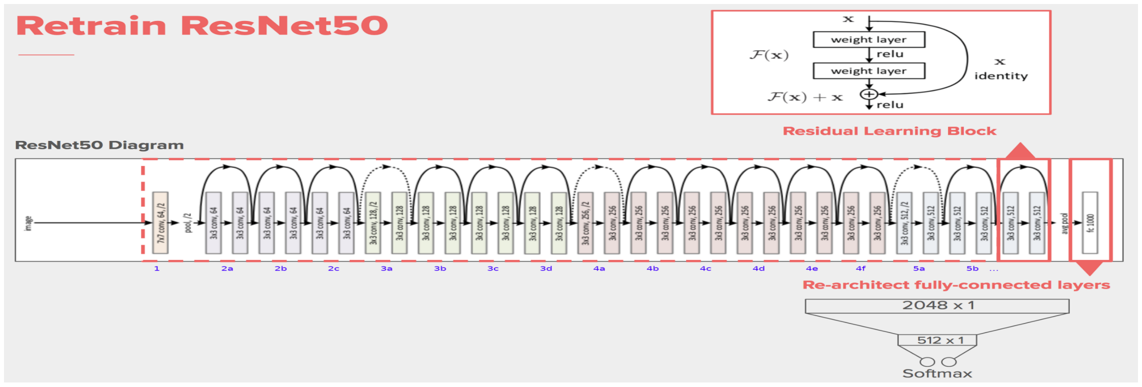


Figure 3: ResNet-50 Architecture

To complete the archeticture, the ResNet-50 was followed by a fully-connected network. The output of ResNet-50 is flattened and feeded to a fully connected network with two Dense layers containing 1024 neurons and two Dropout layers for regularization.

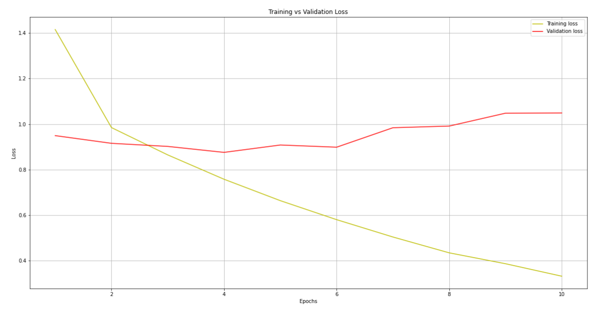
## 3.2 Training and Initialization Weights

For training, instead of using random weights during initialization we are using pre-trained weights during the weight initialization. We are using pre-trained weights of ResNet50 which was trained on ImageNet [2].
ImageNet is an image dataset organized according to the WordNet hierarchy. There are more than 100,000 synsets in WordNet. In ImageNet, there is on average 1000 images to illustrate each synset.
For training we used **Adam** as the optimizer with **learning rate = 0.00001**. Entire network was trained using **10 epochs**.

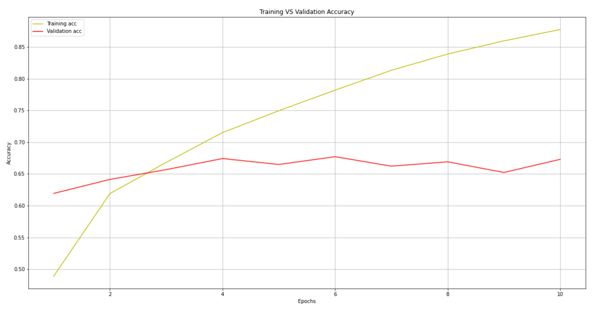
## 3.3 Results and Observation

We trained the ResNet-50 model using the ImageNet weights during initialization. We firstly trained the model on all the 5 classes ('Jacket', 'Skirt', 'T-Shirt', 'Shorts', and 'Top') and got the validation accuracy as **67.31%**. You can see the loss and accuracy values over each epoch in Figure 4.

When analysed the cause of less accuracy, and we observed that there is possibility of two classes being



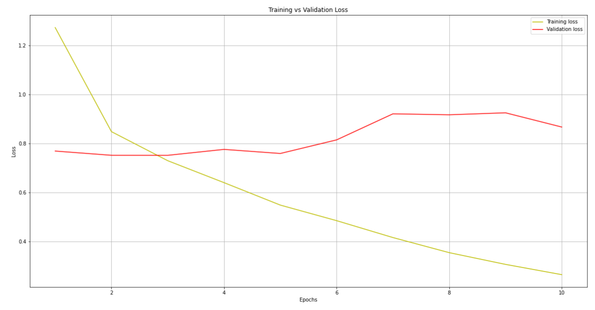
(a) Loss



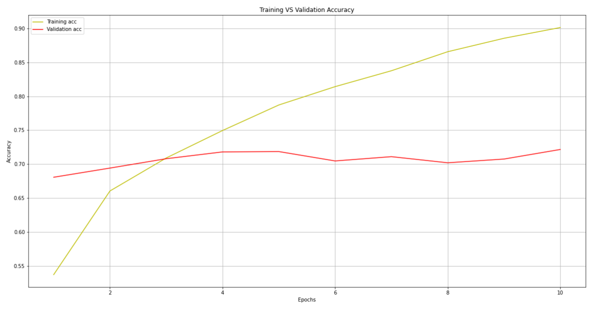
(b) Accuracy

Figure 4: Comparison of Training and Validation

in one image causing the classifier to get confused. Like an image in the dataset of a person wearing a 'T-Shirt' who is also wearing 'Shorts', is labeled as 'T-Shirt' but the classifier can get confused and recognise it as 'Shorts' which is correct but is wrong according to the known label, leading to a decrease in accuracy.
So we re-trained the model with dataset without the class 'Shorts'. We found that our accuracy increased to **72.17%**. You can see the loss and accuracy values over each epoch in Figure 5



(a) Loss



(b) Accuracy

Figure 5: Comparison of Training and Validation

Hence we concluded that using a detection model will be more proficient for this system than the classification (recognition) model as classification model can't properly classify if an image contains two or more categories.

## 4 Dress Detection using YOLO

The You Look Only Once object detection approach re-frames object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Unlike other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in a image, the Yolo architecture passes the image once through the FCNN and the output is the prediction. This makes Yolo extremely fast and an ideal detector for our purpose. Yolo divides the input image into an S  S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. Formally the confidence score is defined as $Pr(Object) \times IOU$. The confidence scores reflect how confident the model is about the bounding box it predicts and the class of the object it predicts. The Intersection Over Union (IOU) is the area of intersection divided by the area of union of the predicted and original bounding boxes [5].

The output of the detector is the class prediction, bounding box prediction and a confidence score. The confidence is given as :

$$Pr(Class_i/Object) * Pr(Object) * IOU = Pr(Class_i * IOU)$$

We used the Yolo-tiny network which is a lighter version of the Yolo network. It consists of 23 Convolutional layers and 2 yolo layers.

## 4.1 Data Augmentation

A large amount of data is crucial for training a deep learning model. Data augmentation techniques can be used to artificially expand the dataset by creating modified versions of it. These techniques create variations of the dataset and improve the ability of the model to generalise to new images. The following augmentation techniques were applied at random :

- **Hue :** Randomly changed to a value between $90 - 110\%$.
- **Saturation :** Randomly changed to a value between $50\% - 150\%$.
- **Brightness :** Randomly changed to a value between $50\% - 150\%$.

## 4.2 Training the network

The customised dataset consisting of five categories with 11000 images each was split into 30% test and 70% train. The starting weights were initialised with the pre-trained weights of the yolo-tiny network trained on the COCO dataset. This was done to obtain faster convergence which in turn reduces the training time. Dropout of 0.25 and 0.5 was used for regularization. The optimizer was chosen (after hyperparameter tuning) to be SGD with 0.9 momentum and 0.0005 decay. The configuration file for the network can be found at [6]. The training loss is shown in Figure 6.

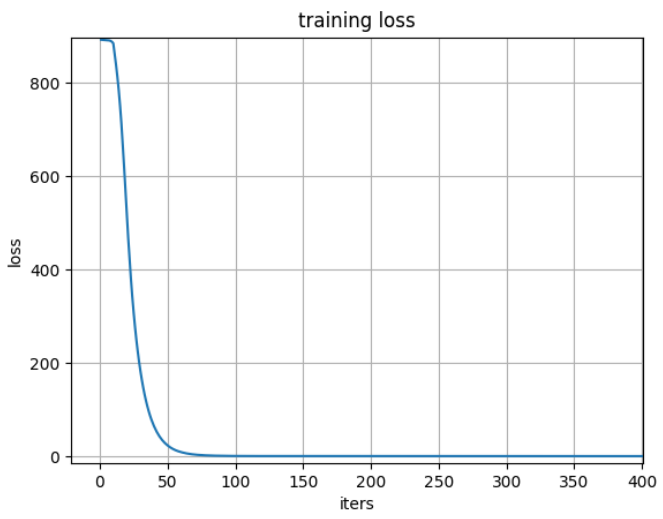


Figure 6: Training Loss of Yolo-tiny network

## 4.3 Results and Observations

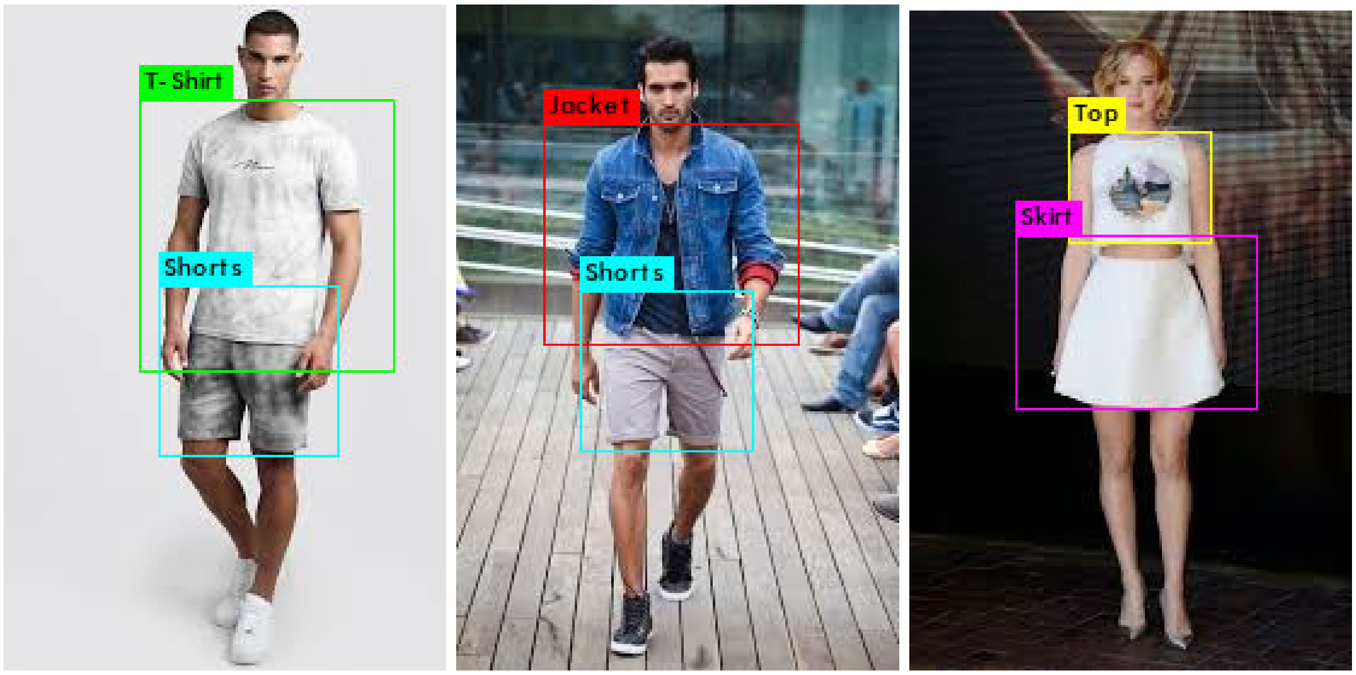
The results of evaluating the trained detector on the test set are shown in Table 1. The IOU threshold was set to 0.5. However, the deepfashion dataset did not have multiple annotations for all images. For example, an image containing a person wearing a skirt and a T-Shirt was annotated for skirt but the T-shirt the person was wearing was not annotated. Thus, if such an image were to be a part of the test set, the detector will also detect the T-shirt since it has been trained on T-shirts as well. However, since the true label does not contain T-shirt, this prediction will be considered false. This leads to a decreased map. Thus, we believe that the actual map for detection is much higher than computed from the Deepfashion test set. The results of detection for three images randomly taken from the internet are shown in Figure 7.

*The average prediction time for an image is 8ms using NVIDIA 1050 Ti GPU*

The dress detector is able to detect multiple dresses that are worn by a person in the image. This is a good improvement compared to the recognition system which could detect only one dress. Moreover, the average prediction time for the detector is $8ms$ and it is only $33MB$ is size. It can thus be deployed onto an edge computing device to perform reliable dress detection.

| **Category** | **ap@0.5IOU(%)** |
|---|---|
| **T-Shirt** | 77.89 |
| **Jacket** | 57.77 |
| **Top** | 81.64 |
| **Short** | 86.22 |
| **Skirt** | 71.44 |
| **Total (map)** | 75 |

Table 1: Mean Average Precision



(a) T-Shirt : 74%, Shorts : 75%  (b) Jacket : 72%, Short : 82%  (c) Skirt : 92%, Top : 52%

Figure 7: Dress Detection Results

## 5 Conclusion

We have proposed a dress detection system which can be used in retail shops to detect the different kinds of dress that a person wears. We initially developed a recognition system which first detects a person in the image and then proceeds to classify the dress. However, such a system has many drawbacks and hence we adopted a direct object detection approach. The major advantage of the object detection approach is the ability to detect multiple dresses. In addition, the YOLO-Tiny architecture is extremely fast and can be used for real time applications. This served as the motivation to train a dress detector using the YOLO-tiny architecture.

## 6 Future Work

The dress detector can be deployed onto an edge computing device in order to completely build a dress detection solution for a retail shop. Moreover, such a system will also generate a lot of data regarding the dresses that customers are wearing.
Real time data collection and performing dress detection will help in understanding the current fashion trends. Understanding fashion trends will greatly help the retail shops in interpreting people's choice and accordingly change the display clothes on the showcase.
With this we can also try understand the change in fashion trends over time in-order to generate a model that predicts the future trends so that fashion designers and retailers can plan/design accordingly.

## References

[1] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[2] *ImageNet*. (accessed May 2020). URL: `http://www.image-net.org/`.

[3] Ziwei Liu et al. "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[4] Ajay R R, Ronak Doshi, and Ram S. *Custom Dress Dataset*. URL: `https://drive.google.com/open?id=1-7dqKOlJbO5p8R6alIhzNlTN9bp14Fae`.

[5] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. URL: `https://ieeexplore.ieee.org/document/7780460`.

[6] Ajay RR. *Dress detection yolo tiny configuration file*. URL: `https://github.com/AJ-RR/darknet/blob/master/cfg/dress.cfg`.