greatlearning

Image Based Search

Using

Convolution Neural Network (YOLOv3)

Object Detection and Localization

Project Report

[15/02/2020]

Team "DeepMinds" (Group-4)

1. Gauravski Ahuja
2. Ankush Belorkar
3. Suhail Mukadam
4. Joseph Raju

Mentor

Rajeev Kumar

Program Director

Dr. Narayana

Table of Contents

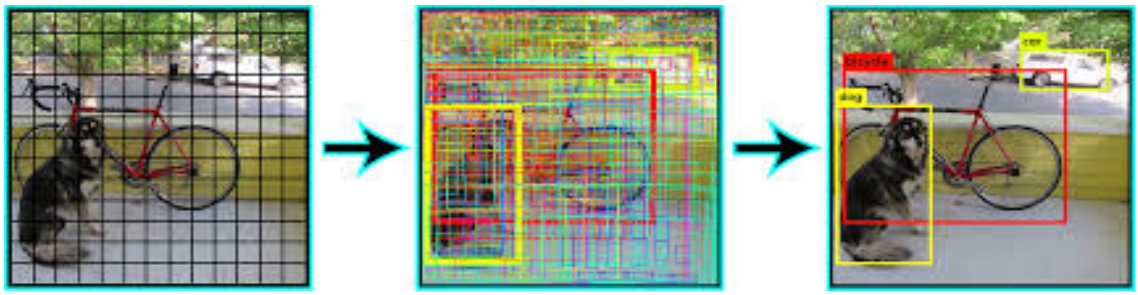
Abstract

Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance. In this project, we use a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the publicly available dataset (Deep Fashion 2). The resulting system is fast and accurate, thus aiding those applications, which require object detection in Fashion domain.

1. Introduction

1.1 Problem Statement

Many problems in computer vision were saturating on their accuracy before a decade. How- ever, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the object in the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box. An overview of all these problems is depicted in below Figure.



1.2 Application of Solution

Traditional way of searching any product on ecommerce application is using text and applying filter criteria manually. To provide better user experience this system can be integrated with ecommerce application running on handheld device with camera which will detect fashion object and extract specific features such as type of fashion object(Shirt, Trouser, T-Shirt etc) which can be used to present similar object. Current system can be further use to detect brand name, color, size which will reduce end user turnaround time to buy any product.

1.3 Challenges

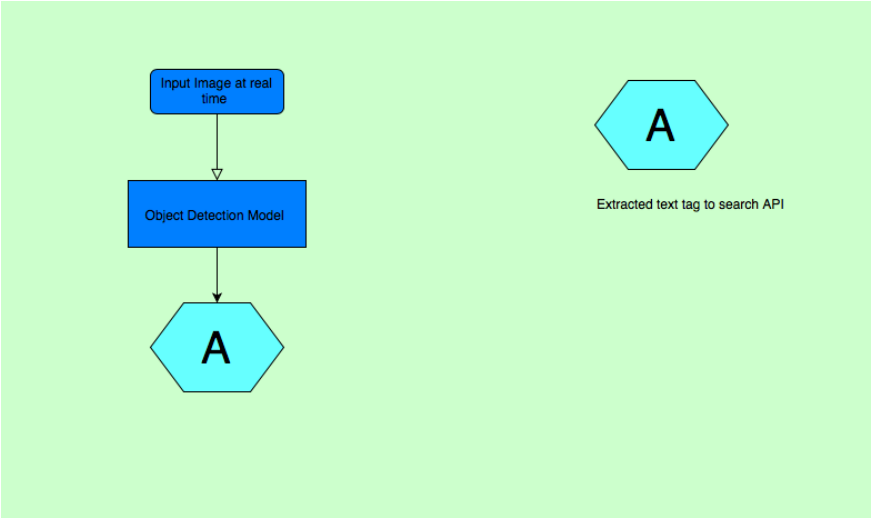
The major challenge in this problem is that of the variable dimension of the input image which is caused due to the different resolution devices used to capture image. Any general Image processing task requires a fixed dimension of input and output for the model to be trained. Another important obstacle for widespread adoption of object detection systems is the requirement of real-time (**30fps**) while being accurate in detection. The more complex the model is, the more time it requires for inference; and the less complex the model is, the less is the accuracy. The problem involves classification as well as regression, leading the model to be learnt simultaneously. This adds to the complexity of the problem.

1.4 Proposed Solution

To solve this problem, we are proposing search mechanism based on image provided by user. This system will be image based search mechanism for Fashion industry.

The solution we started working on is to do Object detection on images for fashion objects (shirts , trousers etc…) and mark the bounding box and display label. The detected objects will then be passed as search string elements to the search api resulting in a near accurate search.

Please refer to the solution flow as below:



1.5 Related Work

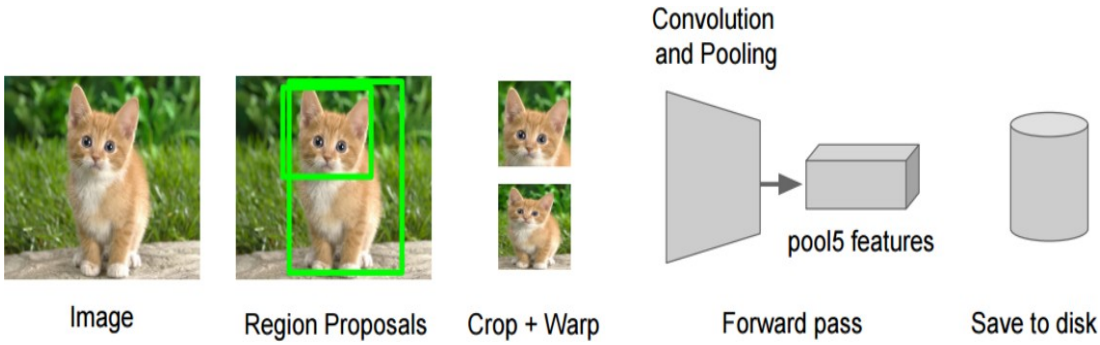
Among the deep learning based techniques, two broad class of methods are prevalent: two stage detection (RCNN [1], FastRCNN [2], FasterRCNN [3]) and unified detection (Yolo, SSD).

The high-level concepts involved in these techniques have been explained below.

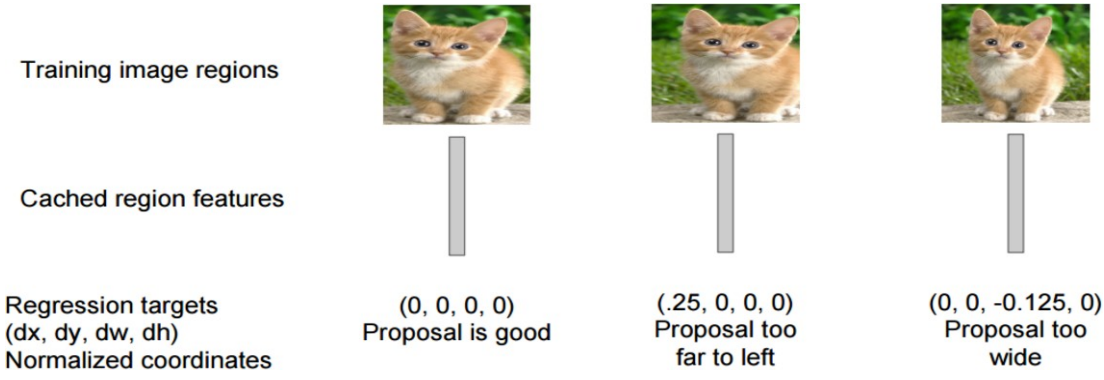
1.5.1 Two-stage Method

In this case, the proposals are extracted using some other computer vision technique and then resized to fixed input for the classification network, which acts as a feature extractor. Then an SVM is trained to classify between object and background (one SVM for each class).

Also a bounding box regressor is trained that outputs some correction(offsets) for proposal boxes. The overall idea is shown in below image. These methods are very accurate but are computationally intensive (low fps).



Stage 1



Stage 2

Figure:Two-StageMethod

1.5.2 Unified Method

The difference here is that instead of producing proposals, pre-define a set of boxes to look for objects. Using convolution feature maps from later layers of the network, run another network over these feature maps to predict class cores and bounding box offsets. The broad idea is depicted in below image. The steps are mentioned below:

1. Train a CNN with regression and classification objective.
2. Gather activation from later layers to infer classification and location with fully connected or convolution layers.
3. During training, use jaccard distance to relate predictions with the ground truth.
4. During inference, use non-maxima suppression to filter multiple boxes around the same object.

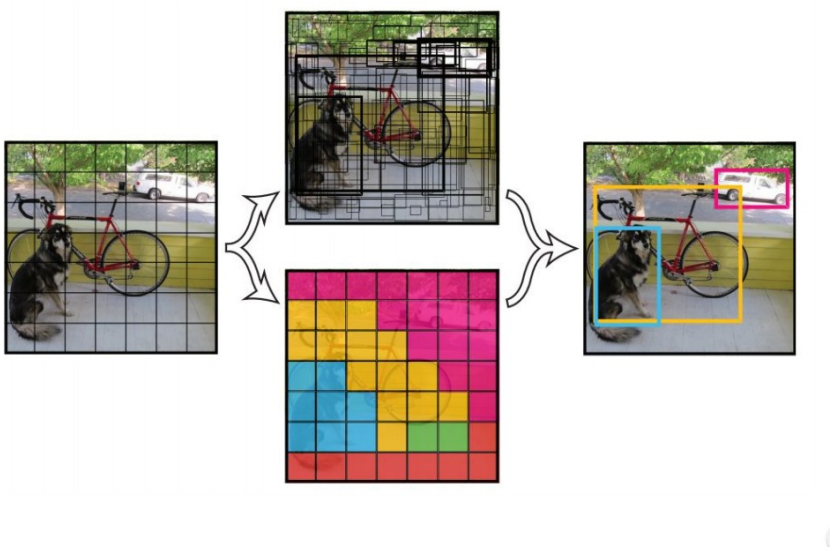
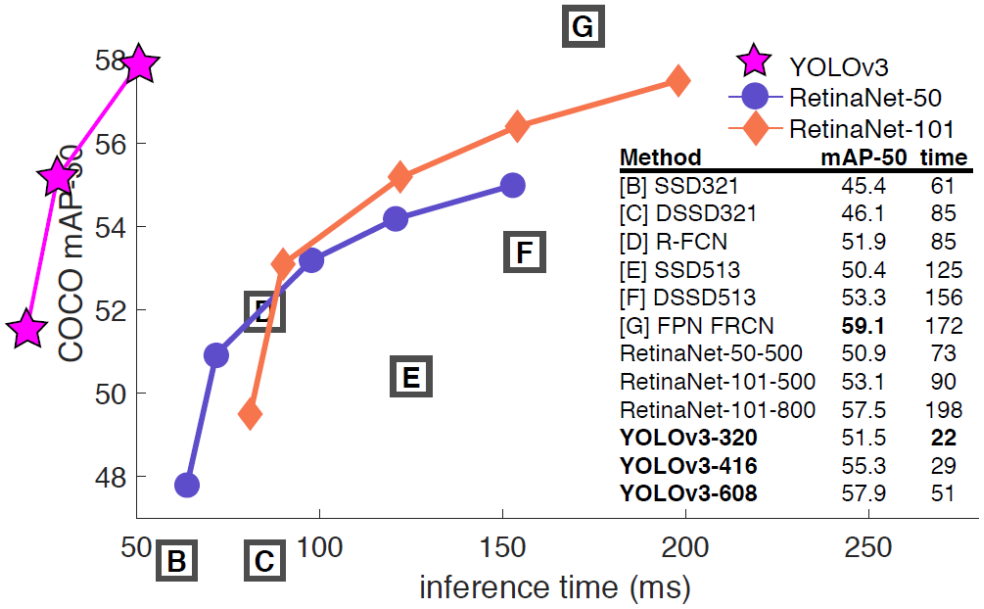


Figure : Unified Method

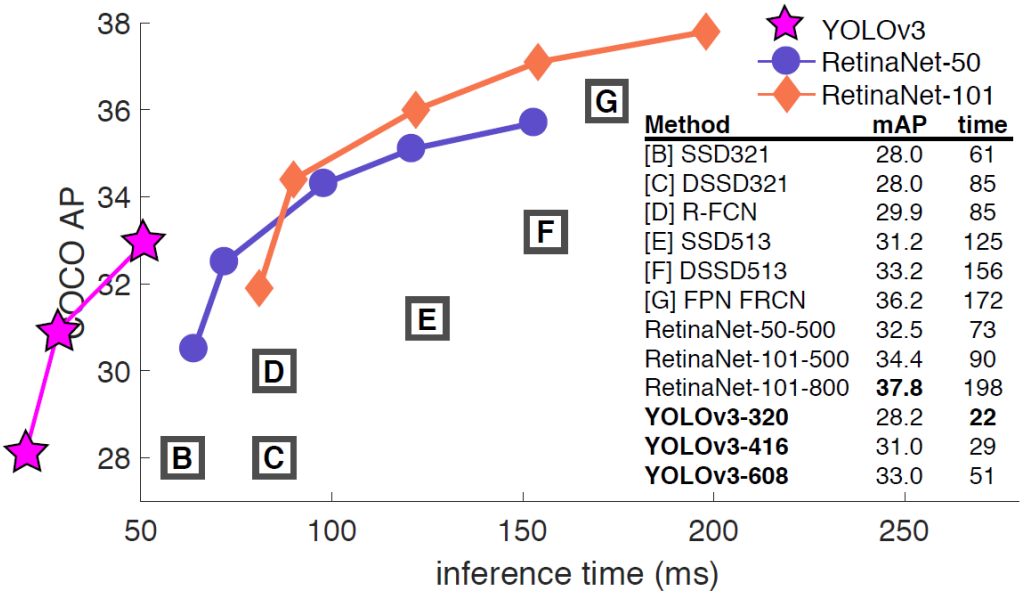
(The major techniques that follow this strategy are, Yolo (uses a single activation map for prediction of classes and bounding boxes) and SSD (uses different activation maps (multiple-scales) for prediction of classes and bounding boxes).

2. Benchmark Comparison of performance based on Coco mAP survey



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

As shown above, compared with RetinaNet, YOLOv3 got comparable mAP@0.5 with much faster inference time. For example, YOLOv3–608 got 57.9% mAP in 51ms while RetinaNet-101–800 only got 57.5% mAP in 198ms, which is 3.8× faster.

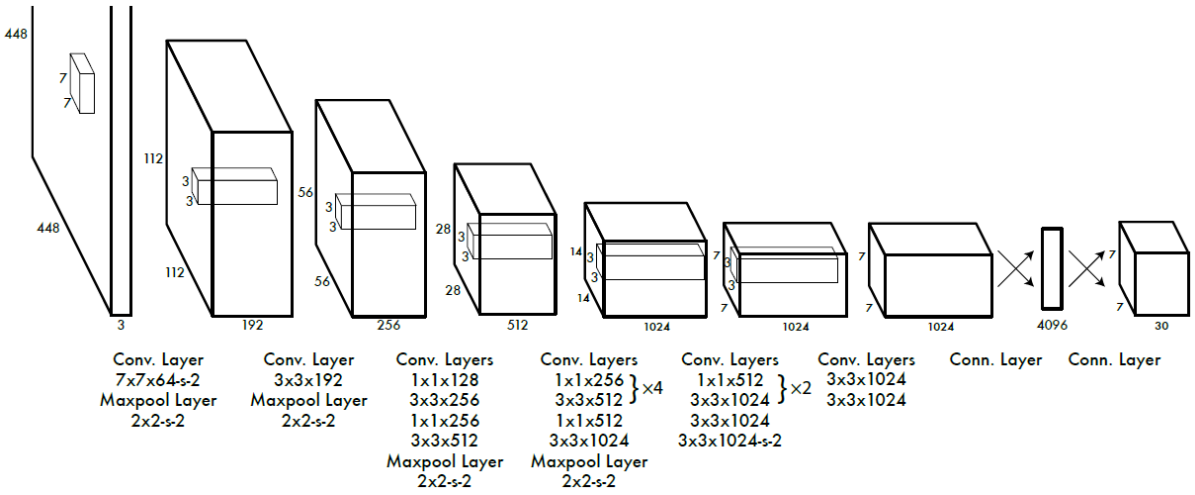


Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

For overall mAP, YOLOv3 performance is dropped significantly. Nevertheless, YOLOv3–608 got 33.0% mAP in 51ms inference time while RetinaNet-101–50–500 only got 32.5% mAP in 73ms inference time and YOLOv3 is on par with SSD variants with 3× faster.

3. YOLO Approach

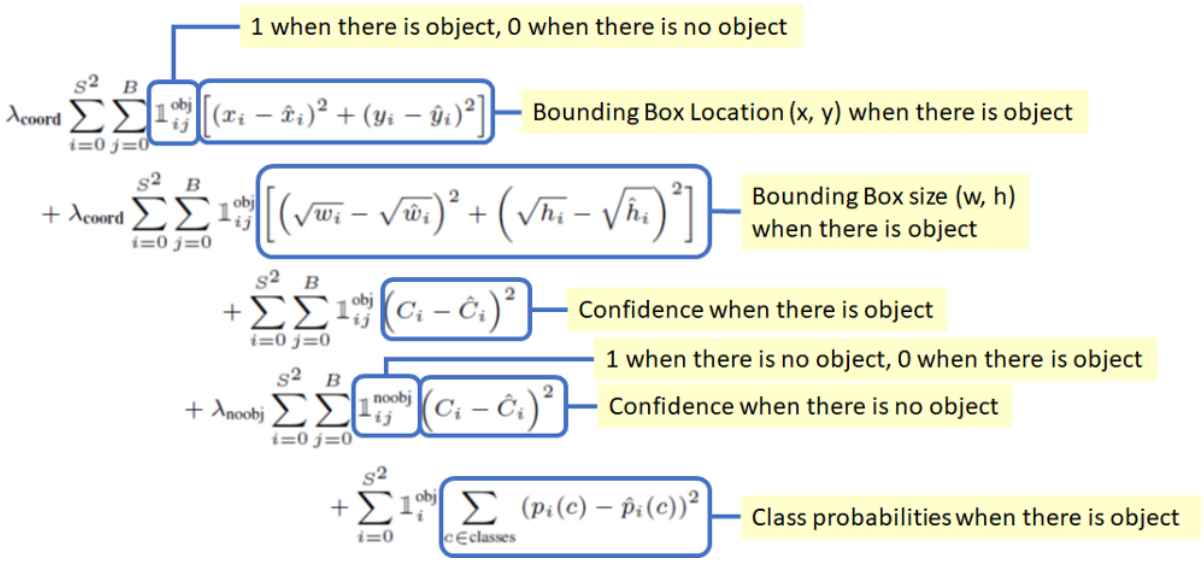
The Network used in this project is based on ***You Only Look Once (YOLO***). The YOLO architecture is shown in fig below.



- In YOLO, input image is divided into a grid of say, 13*13 cells (S=13)
- Each of these cells is responsible for predicting 5 bounding boxes (B=5)

  (A bounding box describes the rectangle that encloses an object)
- YOLO for each bounding box
  - outputs a confidence score that tells us how good the shape of the box is
  - the cell also predicts a class
  - The confidence score of bounding box and class prediction are combined into final score -> probability that this bounding box contains a specific object

3.1 Loss Function



- 1st term (x, y): The bounding box x and y coordinates is parameterized to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

  Sum of square error (SSE) is estimated only when there is object.

- 2nd term (w, h): The bounding box width and height are normalized by the image width and height so that they fall between 0 and 1.

  SSE is estimated only when there is object.

  Since small deviations in large boxes matter less than in small boxes. Square root of the bounding box width w and height h instead of the width and height directly to partially address this problem.

- 3rd term and 4th term (The confidence)
  - ✓ The IOU between the predicted box and any ground truth box
  - ✓ In every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects, and makes the model unstable.
  - ✓ The loss from confidence predictions for boxes that don't contain objects, is decreased, i.e. λnoobj=0.5.

- 5th term (Class Probabilities): SSE of class probabilities when there is objects.

  During prediction, non-maxima suppression is used to filter multiple boxes per object that may be matched.

4. Experimental Results

4.1 Exploratory Data Analysis

For the purpose of this project, the publicly available **DeepFashion2 dataset** is used. It is comprehensive fashion dataset containing **491K diverse images** of **13 popular clothing categories** from both commercial shopping stores and consumers. It totally has 801K clothing items, where each item in an image is labeled with scale, occlusion, zoom-in, viewpoint, category, style, bounding box, dense landmarks and per-pixel mask. There are also 873K

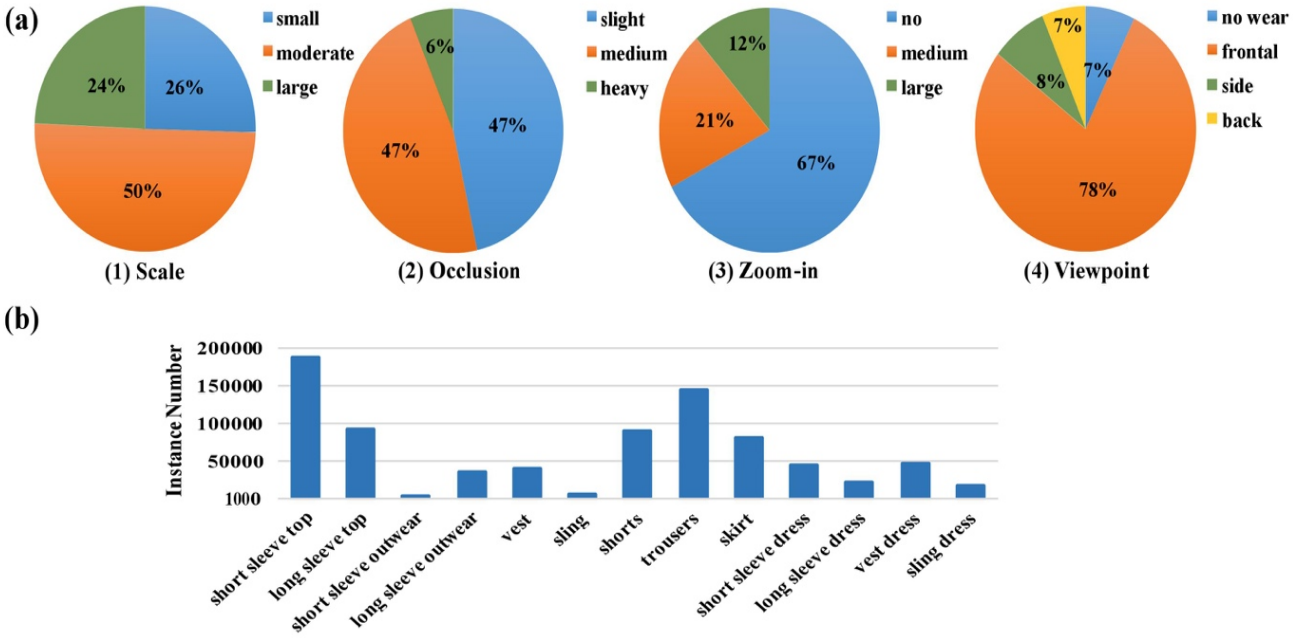
Commercial-Consumer clothes pairs. The dataset is split into a training set (391K images), a validation set (34k images), and a test set (67k images).

4.2 Dataset Statistics

	Train	Validation	Test	Overall
Images	390884	33669	67342	491895
BBboxes	636624	54910	109198	800732
Landmarks	636624	54910	109198	800732
masks	636624	54910	109198	800732

4.3 Exploratory Visualization

Depictions below shows the statistics of different variations and the numbers of items of the 13 categories in DeepFashion2. Data was found to be highly biased towards few classes (short sleeve top, trousers, skirt, and shorts)



5. Implementation Details

The project is implemented in python 3. Darknet implementation was used for training the deep network and OpenCV was used for image pre-processing. The system specifications on which the model is trained and evaluated is mentioned as follows:

CPU - AMD 3900x 3.90 GHz, RAM - 16 GB, GPU - Nvidia 2080Ti.

6. Pre-Processing of Data

The Deepfashion2 dataset obtained was clean enough. The following steps were taken to Normalize the data:

- Resizing of images was done converting into 416*416 dimensions
- Based on the height and weight of images – normalized the bounding boxes of objects
- We created text file from the image label data (.json files) to extract class identifiers and respective image bounding box coordinates.
- Formatting of data files required for Darknet configuration

7. Initial Training on minimal image samples

- We started by extracting **200** random images from Deepfashion2 dataset.
- We explored on how to use Darknet Yolo3 architecture as this is proven faster and accurate for object detection on images.
- We performed the EDA on the dataset
- We started building a code to read the images in a clean way
- Installed Darknet in Google Collab as Collab provided a better environment for executing the initial proof of concept faster.
- Modified the darknet configurations as per the DeepFashion2 requirements. For example, the number of class outputs expected etc.…
- Trained the images on Darknet
- Tested few images for Object Detection with bounding boxes and labels displayed

8. Final Training done on Full dataset

- Google Collab did not come to our rescue while training the full dataset as the session terminated every ***90 minutes*** with a weight file being generated. Every time the training was re-started, the training started from scratch.
- We realized we required larger memory in order to execute the training on full dataset.
- We arranged for a GPU connected to one of our desktop pc.
- The next challenge we faced was installing Darknet on Windows.
- Once we successfully installed the Darknet, we started training the full dataset. The training extended upto 36+ hours to finally get completed.
- For testing the trained model on Deepfashion2 dataset, we provided real images of our own clothing.

- The results came out very quickly with fairly good accuracy and greater IOU

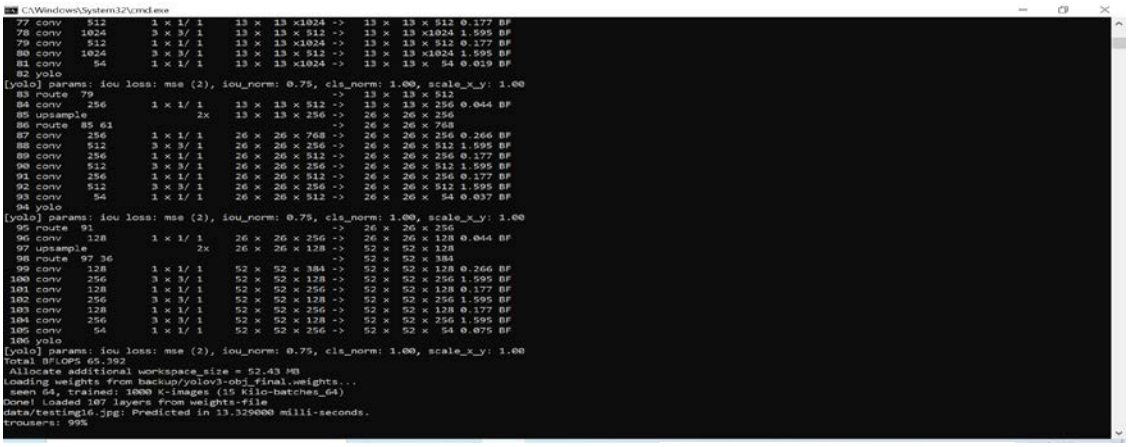
9. Extracting the Test outputs for objects detected into Search API

- The test outputs provided the objects detected with information on the object class labels.
- The object class labels were extracted after being parsed in the output file and fed to the search engine.
- The search engine displayed the matching clothes images as an outcome of this solution.

10. Model Evaluation

Darknet Yolov3 was used as the final model to train the Deepfashion2 dataset. The architecture consists of Total **106** total layers (**75** Convolution layers with **3 YOLO layers** for scaling at varying sizes of clothing objects). Objective of the Model was to accurately classify the objects per the problem statement with less turnaround time. The Accuracy and speed are prominent parameters. On a small test samples of around 15 images/ clothing category **accuracy was observed to be closer ~90%** although the model was not able to detect objects accurately on low quality images or objects were classified incorrectly.

Test Output→ 1



Test Image

Predicted as "Trousers" with **99% accuracy**
(in 13.32 mili-seconds)

Test Output→ 2

```
Select C:\Windows\System32\cmd.exe
  78 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  79 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  80 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  81 conv     54       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x  54 0.019 BF
  82 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  83 route  79                                ->   13 x  13 x 512
  84 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
  85 upsample                 2x    13 x  13 x 256 ->   26 x  26 x 256
  86 route  85 61                             ->   26 x  26 x 768
  87 conv    256       1 x 1/ 1     26 x  26 x 768 ->   26 x  26 x 256 0.266 BF
  88 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  89 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  90 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  93 conv     54       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x  54 0.037 BF
  94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  95 route  91                                ->   26 x  26 x 256
  96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
  97 upsample                 2x    26 x  26 x 128 ->   52 x  52 x 128
  98 route  97 36                             ->   52 x  52 x 384
  99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
 100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 105 conv     54       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x  54 0.075 BF
 106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.392
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3-obj_final.weights...
 seen 64, trained: 1000 K-images (15 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
data/testmycrop.jpg: Predicted in 13.216000 milli-seconds.
short sleeve top: 76%

D:\darknet-master\darknet-master>
```

Test Image

Predicted as "short sleeve top" with **76% Accuracy** (in 13.21 mili-seconds)

Test Output→ 3

```
Select C:\Windows\System32\cmd.exe
  79 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  80 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  81 conv     54       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x  54 0.019 BF
  82 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  83 route  79                                ->   13 x  13 x 512
  84 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
  85 upsample                 2x    13 x  13 x 256 ->   26 x  26 x 256
  86 route  85 61                             ->   26 x  26 x 768
  87 conv    256       1 x 1/ 1     26 x  26 x 768 ->   26 x  26 x 256 0.266 BF
  88 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  89 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  90 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  93 conv     54       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x  54 0.037 BF
  94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  95 route  91                                ->   26 x  26 x 256
  96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
  97 upsample                 2x    26 x  26 x 128 ->   52 x  52 x 128
  98 route  97 36                             ->   52 x  52 x 384
  99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
 100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 105 conv     54       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x  54 0.075 BF
 106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.392
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3-obj_final.weights...
 seen 64, trained: 1000 K-images (15 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
data/testimg6.jpg: Predicted in 13.797000 milli-seconds.
shorts: 86%
vest: 86%

D:\darknet-master\darknet-master>
```

Test Image **Predicted as "vest and shorts"** with **86% Accuracy** (in 13.79 mili-seconds)

<u>Test Output</u>→ 4

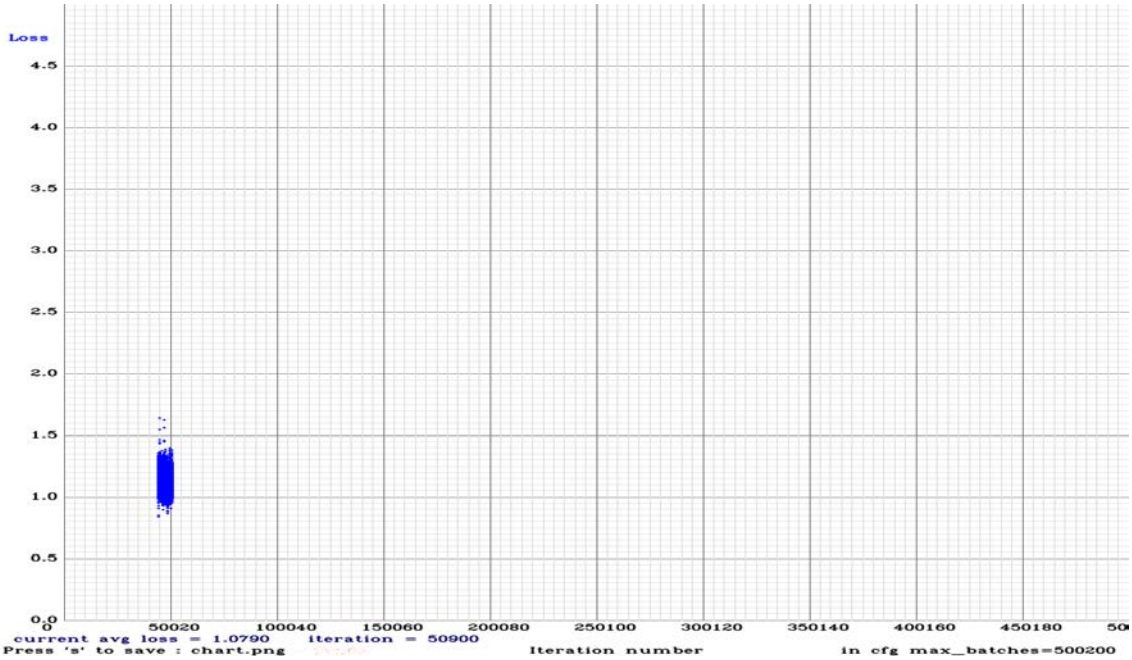
```
Select C:\Windows\System32\cmd.exe
  78 conv    1024       3 x 3/ 1     13 x  13 x 512 ->    13 x  13 x1024 1.595 BF
  79 conv     512       1 x 1/ 1     13 x  13 x1024 ->    13 x  13 x 512 0.177 BF
  80 conv    1024       3 x 3/ 1     13 x  13 x 512 ->    13 x  13 x1024 1.595 BF
  81 conv      54       1 x 1/ 1     13 x  13 x1024 ->    13 x  13 x  54 0.019 BF
  82 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  83 route  79                                   ->    13 x  13 x 512
  84 conv     256       1 x 1/ 1     13 x  13 x 512 ->    13 x  13 x 256 0.044 BF
  85 upsample                 2x     13 x  13 x 256 ->    26 x  26 x 256
  86 route  85 61                                ->    26 x  26 x 768
  87 conv     256       1 x 1/ 1     26 x  26 x 768 ->    26 x  26 x 256 0.266 BF
  88 conv     512       3 x 3/ 1     26 x  26 x 256 ->    26 x  26 x 512 1.595 BF
  89 conv     256       1 x 1/ 1     26 x  26 x 512 ->    26 x  26 x 256 0.177 BF
  90 conv     512       3 x 3/ 1     26 x  26 x 256 ->    26 x  26 x 512 1.595 BF
  91 conv     256       1 x 1/ 1     26 x  26 x 512 ->    26 x  26 x 256 0.177 BF
  92 conv     512       3 x 3/ 1     26 x  26 x 256 ->    26 x  26 x 512 1.595 BF
  93 conv      54       1 x 1/ 1     26 x  26 x 512 ->    26 x  26 x  54 0.037 BF
  94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  95 route  91                                   ->    26 x  26 x 256
  96 conv     128       1 x 1/ 1     26 x  26 x 256 ->    26 x  26 x 128 0.044 BF
  97 upsample                 2x     26 x  26 x 128 ->    52 x  52 x 128
  98 route  97 36                                ->    52 x  52 x 384
  99 conv     128       1 x 1/ 1     52 x  52 x 384 ->    52 x  52 x 128 0.266 BF
 100 conv     256       3 x 3/ 1     52 x  52 x 128 ->    52 x  52 x 256 1.595 BF
 101 conv     128       1 x 1/ 1     52 x  52 x 256 ->    52 x  52 x 128 0.177 BF
 102 conv     256       3 x 3/ 1     52 x  52 x 128 ->    52 x  52 x 256 1.595 BF
 103 conv     128       1 x 1/ 1     52 x  52 x 256 ->    52 x  52 x 128 0.177 BF
 104 conv     256       3 x 3/ 1     52 x  52 x 128 ->    52 x  52 x 256 1.595 BF
 105 conv      54       1 x 1/ 1     52 x  52 x 256 ->    52 x  52 x  54 0.075 BF
 106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.392
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3-obj_final.weights...
 seen 64, trained: 1000 K-images (15 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
data/testimg3.jpg: Predicted in 13.585000 milli-seconds.
trousers: 99%

D:\darknet-master\darknet-master>
```

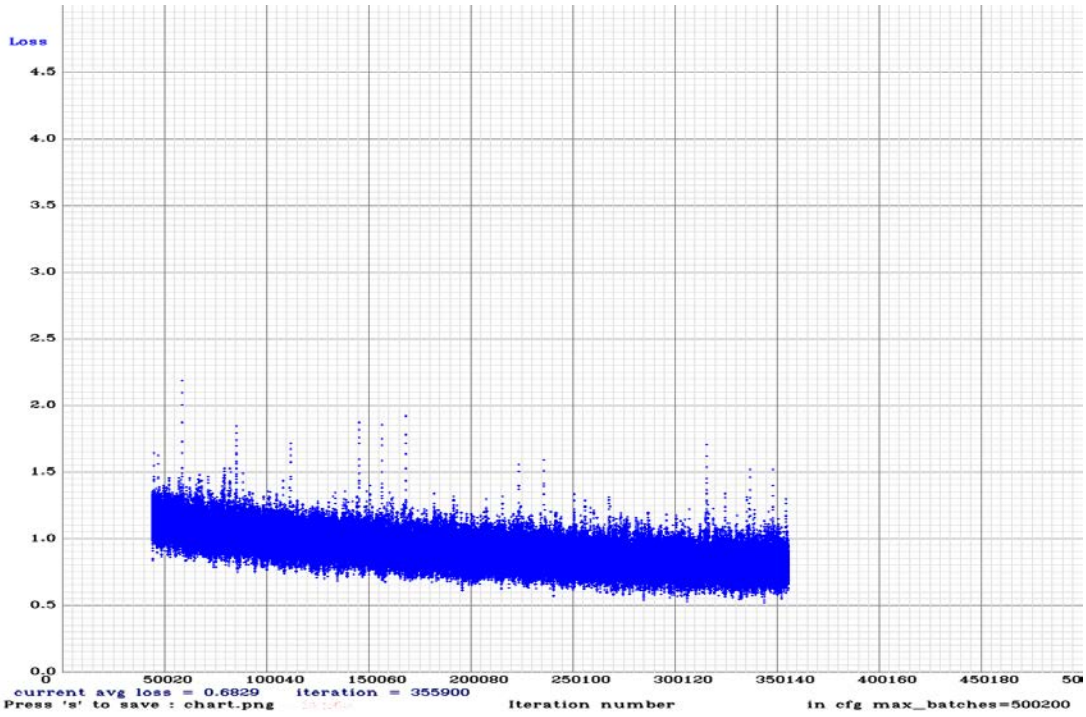
Test Image **Predicted as "trousers"** with **99% Accuracy** (in 13.79 mili-seconds)

11. Loss Function Trends Observed During Model Training

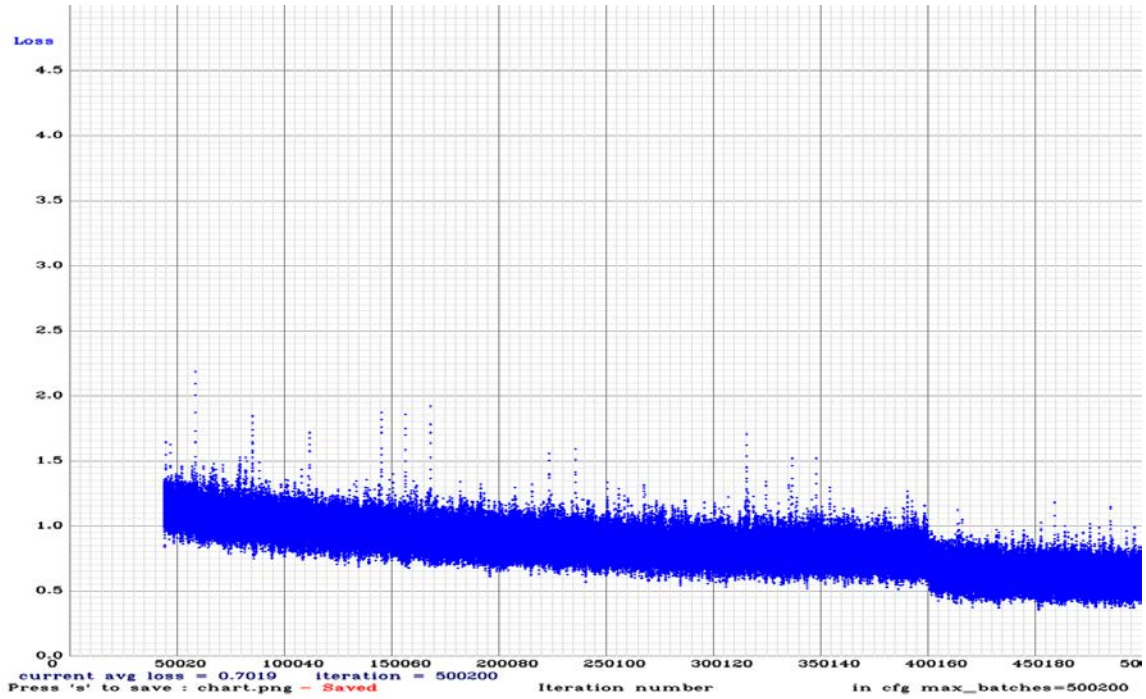
Loss Trend at Start of the training→



Loss Trend at Mid training→



Loss Trend at End of training→



12. Implications by the Current Model

The problem statement is related to fashion industry and particularly w.r.t shopping of clothes from online e-commerce portal. In this situation, the main focus should really be displaying the correct matches to the user searches for specific clothing / styles / colour etc.Often the user sees some clothing of his interest and would like to explore the e-commerce sites to either inquire / buy. An instant response to the search with an accurate match would increase the chances of the sale.

Any Niche clothing article with unique styles and cannot be described in simple text language could now be easily identified in searches on e-commerce sites using this solution. As fashion industry grows year over year – new Clothing categories are consistently been introduced and the solution can really make a difference in the user experience and sale of clothing products.

13. Limitations of the Current Model

Currently the solution is scoped only to 13 clothing categories. In addition, the current training accomplished on the Yolo3 Model provides classified clothing object labels. There is an opportunity to add training for extracting colour and size information and various details on the clothing articles.

The Enhancements to the existing model would scope to cover the extraction of colour, size and other features of clothing articles in order to build the search by image more robust and accurate. In addition, we should be able to integrate the model into an application running on any latest handheld device to capture images at run time and provide inputs to the image search on the Internet.

14. Closing Reflections

Yolov3 is a good object detector. It's fast enough and accurate!!!

Needless to mention there was a tremendous learning from this project. We faced real world Challenges in building the model and train the images. Our realization increased towards the need for getting more labelled data to achieve a more robust solution. Not to forget – the model requires state of the art memory hardware to train model on larger volumes of images.

With these, learning has and experiences, we are better equipped now to deal with any similar real world object detection problem. This should help us approach the business problems with a better solution focussing more on value outcomes.

References

#	Links	Comments
1	. https://github.com/AlexeyAB/darknet?files=1	Darknet Installation and Training
2	https://github.com/switchablenorms/DeepFashion2	DeepFashion2 Dataset
3	https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/	Reference to Image Search techniques
4	https://towardsdatascience.com/calculating-loss-of-yolo-v3-layer-8878bfaaf1ff?_branch_match_id=756534430818647193	Calculate Loss Functions for YOLOv3
5	https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b	YOLO Object Detection technique
6	https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774	Feature extraction and image search
7	https://medium.com/coinmonks/shopping-with-your-camera-visual-image-search-meets-e-commerce-at-alibaba-8551925746d0	Shopping with your phone camera
8	https://github.com/ice609/Object-detection-coding	Reference code Implementation
9	https://github.com/experiencor/keras-yolo3/blob/master/yolo.py	Reference code Implementation

Thank You!!!