DRESS DETECTION AND CLASSIFICATION

Project Report

Submitted To

Dr. RAVI KUMAR JATOTH

Associate Professor

Department of Electronics & communication Engineering

Submitted by-

MANDA ABHINAY REDDY (204242)

PAMMI VIJAY HEMANTH (204255)

Department of Electronics & Communication Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

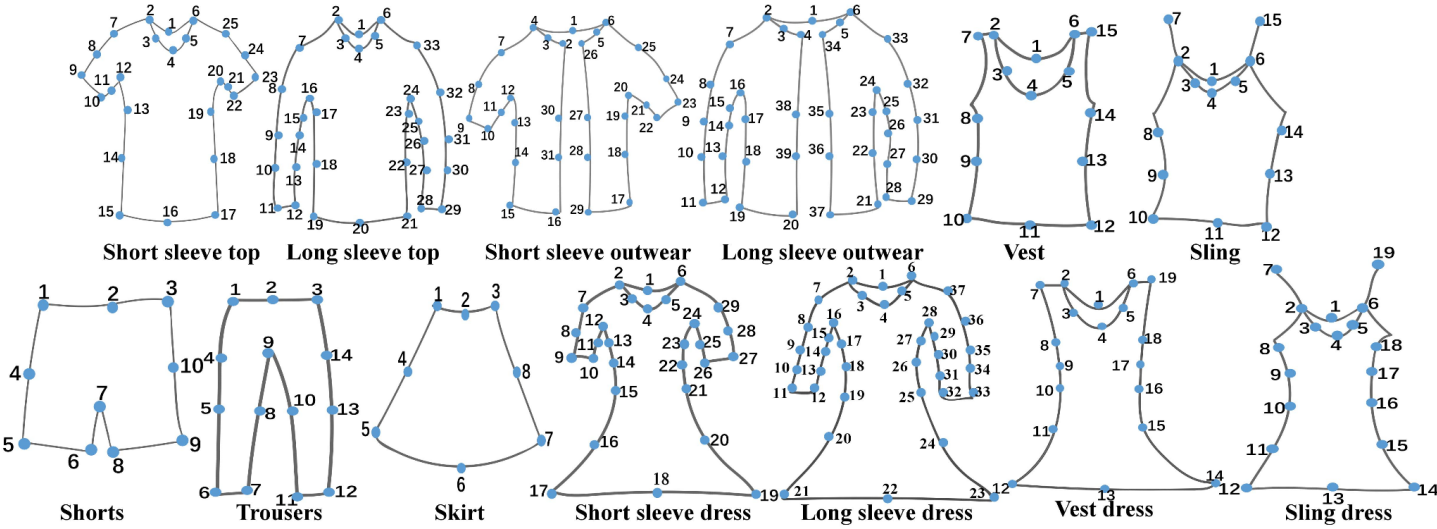
Telangana-506004

ABSTRACT

Dress detection and classification is a machine learning model that is able to detect the dress from an image and can classify it into different types of the categories.In this model we used ‘Make Sense’ a online tool for creating the dataset (bounding boxes and labels of images) and the ‘YOlOv5’(You Look Only Once) algorithm for the training dataset and the compilation of the code is done with the use of pytorch, for further help in compilation of the code and in training the dataset visit :
https://github.com/ultralytics/yolov5

OBJECTIVE

The objective of the project is to generate a model which has clear and precise detection of the dress from an image and classify it into one of the following category ().



CONTENT

YOLOv5 :

YOLO, an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. YOLO reframes object detection as a single regression problem instead of a classification problem.This system only looks at the image once to detect what objects are present and where they are, hence the name YOLO.Each cell in the grid is responsible for detecting objects within itself.An input image is divided into S × S grid cells and some grid cells are responsible for detecting an object present in the image, i.e., only the ones where the centre of the bounding box is in the cell.

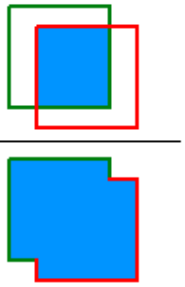
The confidence score indicates how sure the model is that the box contains an object as also how accurate it thinks the box is that predicts.

$$C = Pr(object) * IoU$$

IoU: Intersection over Union

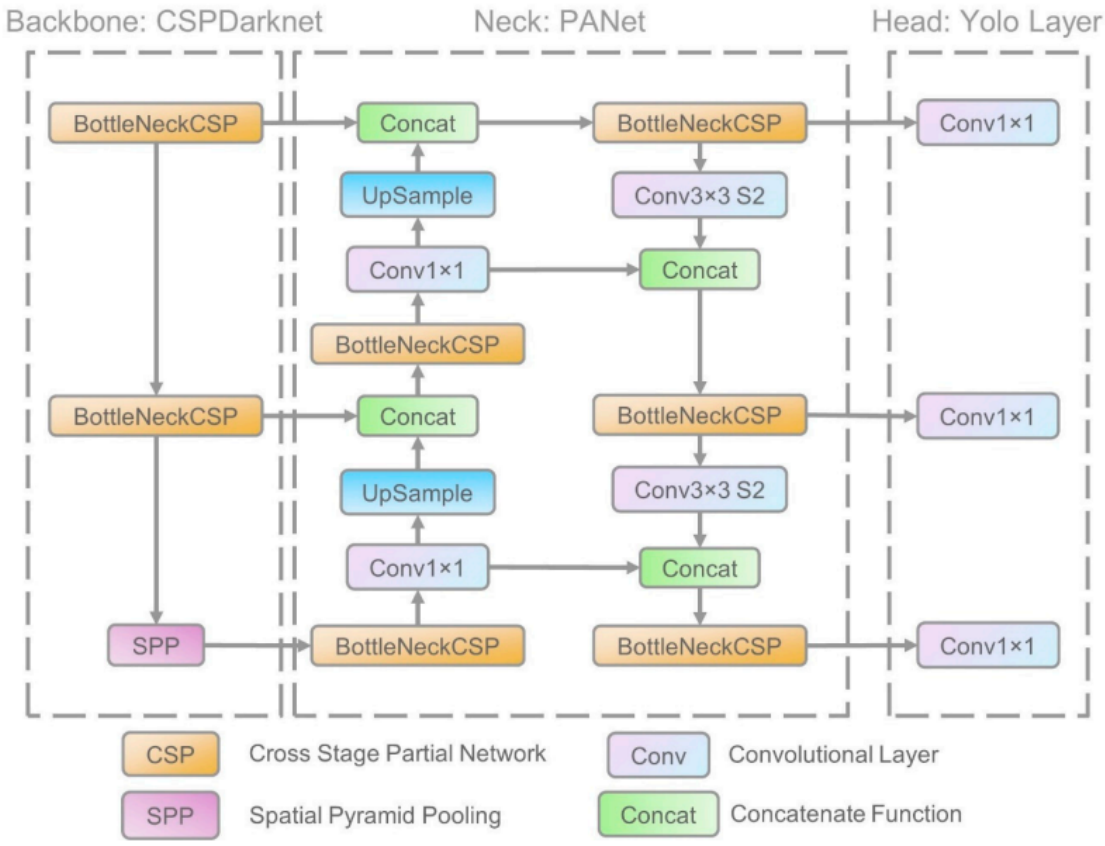
Intersection over union is an evaluation metric used to measure the accuracy of an object detector on a particular data.the accuracy is calculated between the predicted box and the ground truth.

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



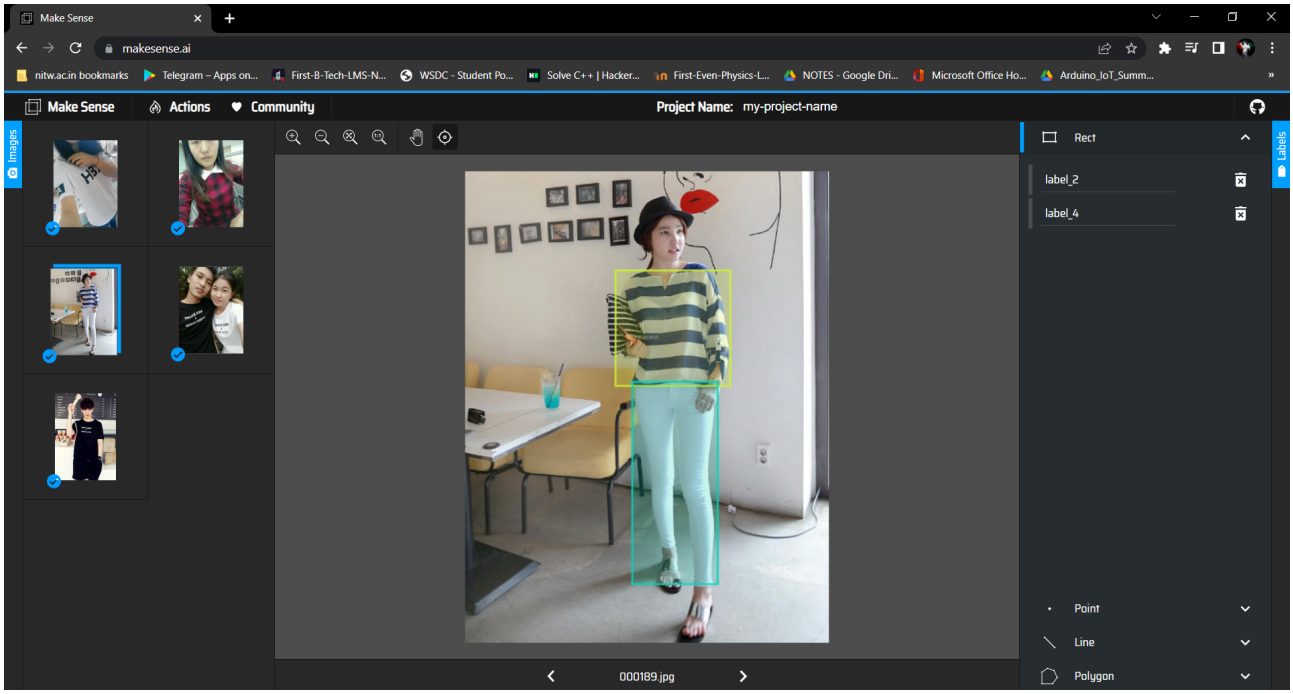
Yolov5 incorporated a cross stage partial network (CSPNet) into Darknet, creating CSPDarknet as its backbone. CSPNet solves the problems of repeated gradient information in large-scale backbones, and integrates the gradient changes into the feature map. Secondly, the Yolov5 applied path aggregation network (PANet) as its neck to boost information flow.PANet adopts a new feature pyramid network (FPN) structure with enhanced bottom-up path, which improves the propagation of low-level features. Thirdly, YOLOv5 developed the final detection part (model head) for feature aggregation. It is responsible for generating the final output vectors including bounding boxes, confidence scores, and class probabilities.YOLOv5 is done with the help of PyTorch.PyTorch is open source machine learning framework based on torch library used for image classification and computer vision.

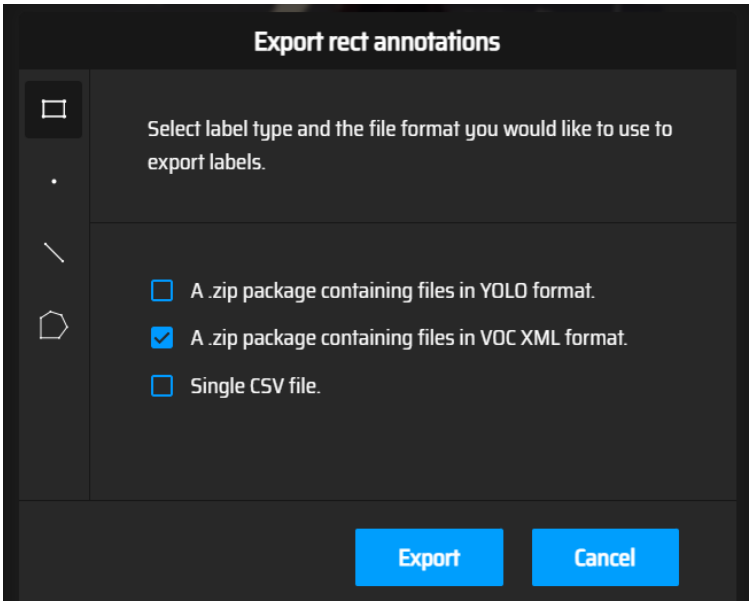
The Architecture of the YOLOv5



MAKE SENSE :

makesense.ai is a free to use online tool for labelling photos.This website can be used for both object detection and the Image classification.It is perfect for small computer vision deep learning projects,the process of preparing a dataset is much easier and faster. Prepared labels can be downloaded in one of multiple supported formats(YOLO, VOC XML, VGG JSON, CGG).


Make Sense
makesense.ai
Make Sense
Actions
Community
Project Name: my-project-name
Images
Labels
Rect
label_2
label_4
Point
Line
Polygon
000189.jpg


Export rect annotations
Select label type and the file format you would like to use to export labels.
A .zip package containing files in YOLO format.
A .zip package containing files in VOC XML format.
Single CSV file.
Export
Cancel

PROCESS

CREATING THE DATA SET :

For the preparation of the dataset we had given a input of () images for testing data and () images for the validation data.The model consists of () classes/categories.The dataset is created with the help of makesense.ai.Each image is uploaded into the website and we made the bounding boxes for the images and exported the annotations into a zip file containing the files in YOLO format,the un-zip file contains the .txt files which has the bounding boxes and the labels/classification of the respected image.

The train images and the validation images should be in the separate folder and both the folders be in the images folder, similarly for the annotations.

The format of the dataset be :

1. train_data (folder)
    a. Images (folder)
        i. train (.jpg images)
        ii. val (.jpg images)
    b. Labels (folder)
        i. train (.txt file)
        ii. val (.txt file)

THE YOLOv5 MODEL :

Clone repositories, install dependencies and check PyTorch and GPU.

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

After running this we get the yolov5 folder in the files.

TRAINING THE DATASET

upload the train_data(dataset) in the colab.download the coco.yaml file and rename as custom_data.yaml and keep the paths of the train , validation of the original dataset.and also change the number of classes/classifications and names of the classification according to the dataset.now upload the custom_data.yaml file in the ./yolov5/data

```
# Train YOLOv5s
!python train.py --img 640 --batch 2 --epochs 60 --data custom_data.yaml --weights yolov5s.pt --cache
```

We train the model is compiled with 60 epochs and of 10 classifications.the train data consists of 312 training images and 45 validation images and 10 test images.

```
  Epoch    gpu_mem       box       obj       cls    labels  img_size
   0/59     0.499G   0.07666   0.03418   0.05212         4       640: 100% 156/156 [00:18<00:00,  8.36it/s]
             Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:00<00:00, 16.15it/s]
               all         44         61    0.00366      0.662     0.0267    0.00607

  Epoch    gpu_mem       box       obj       cls    labels  img_size
   1/59     0.684G   0.05603   0.03256   0.04801         8       640: 100% 156/156 [00:16<00:00,  9.64it/s]
             Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:00<00:00, 16.78it/s]
               all         44         61     0.0319      0.517      0.103      0.038

  Epoch    gpu_mem       box       obj       cls    labels  img_size
   2/59     0.684G   0.05183   0.02823   0.04388        12       640: 100% 156/156 [00:15<00:00,  9.85it/s]
             Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:00<00:00, 17.03it/s]
               all         44         61      0.548     0.0365     0.0906     0.0344
```

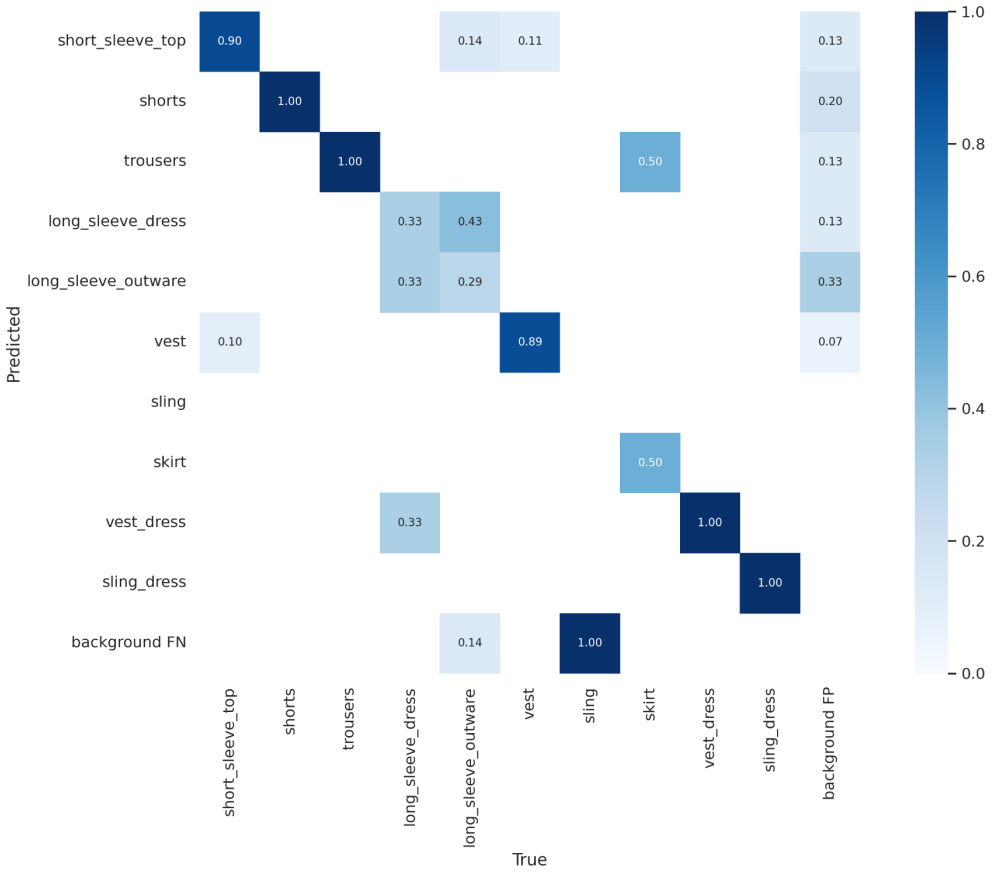
```
      Epoch    gpu_mem       box       obj       cls    labels  img_size
      57/59     0.684G   0.02078   0.01671   0.02009         2       640: 100% 156/156 [00:16<00:00,  9.65it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:00<00:00, 17.41it/s]
                  all         44         61      0.756      0.843      0.898      0.716

      Epoch    gpu_mem       box       obj       cls    labels  img_size
      58/59     0.684G   0.01959   0.01659   0.01966         5       640: 100% 156/156 [00:16<00:00,  9.72it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:00<00:00, 17.41it/s]
                  all         44         61      0.752      0.864      0.901      0.717

      Epoch    gpu_mem       box       obj       cls    labels  img_size
      59/59     0.684G   0.01907   0.01544   0.01804         8       640: 100% 156/156 [00:15<00:00,  9.77it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:00<00:00, 17.91it/s]
                  all         44         61      0.746      0.845      0.896      0.729
```

We are getting an average accuracy of 85% for the training data.

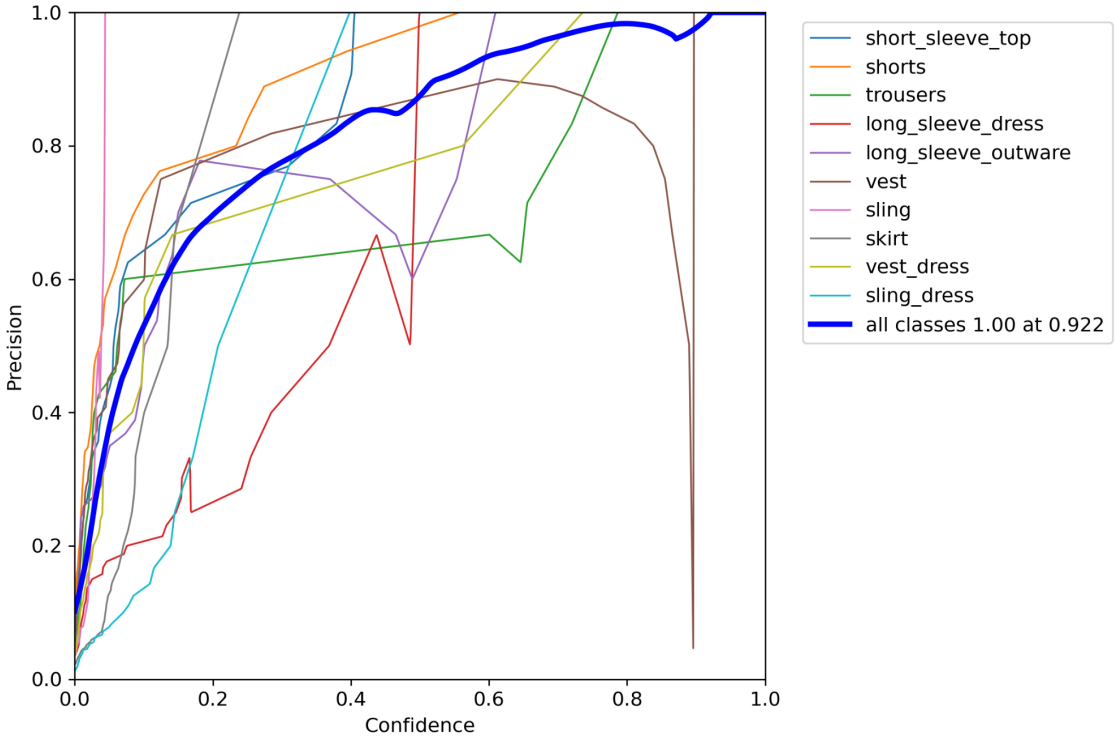
MODEL SUMMARY

```
Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7037095 parameters, 0 gradients
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 11/11 [00:01<00:00, 10.41it/s]
                 all         44         61      0.765      0.836      0.918      0.729
   short_sleeve_top         44         10      0.758          1      0.995      0.796
              shorts         44         16      0.892          1      0.995      0.734
            trousers         44          6      0.626          1      0.942      0.671
  long_sleeve_dress         44          3      0.392      0.667      0.665      0.588
long_sleeve_outware         44          7      0.763      0.924      0.868      0.756
                vest         44          9      0.817          1      0.895      0.679
               sling         44          3          1          0       0.83      0.664
               skirt         44          2          1       0.77      0.995      0.846
          vest_dress         44          4      0.711          1      0.995      0.759
         sling_dress         44          1      0.693          1      0.995      0.796
Results saved to runs/train/exp
```

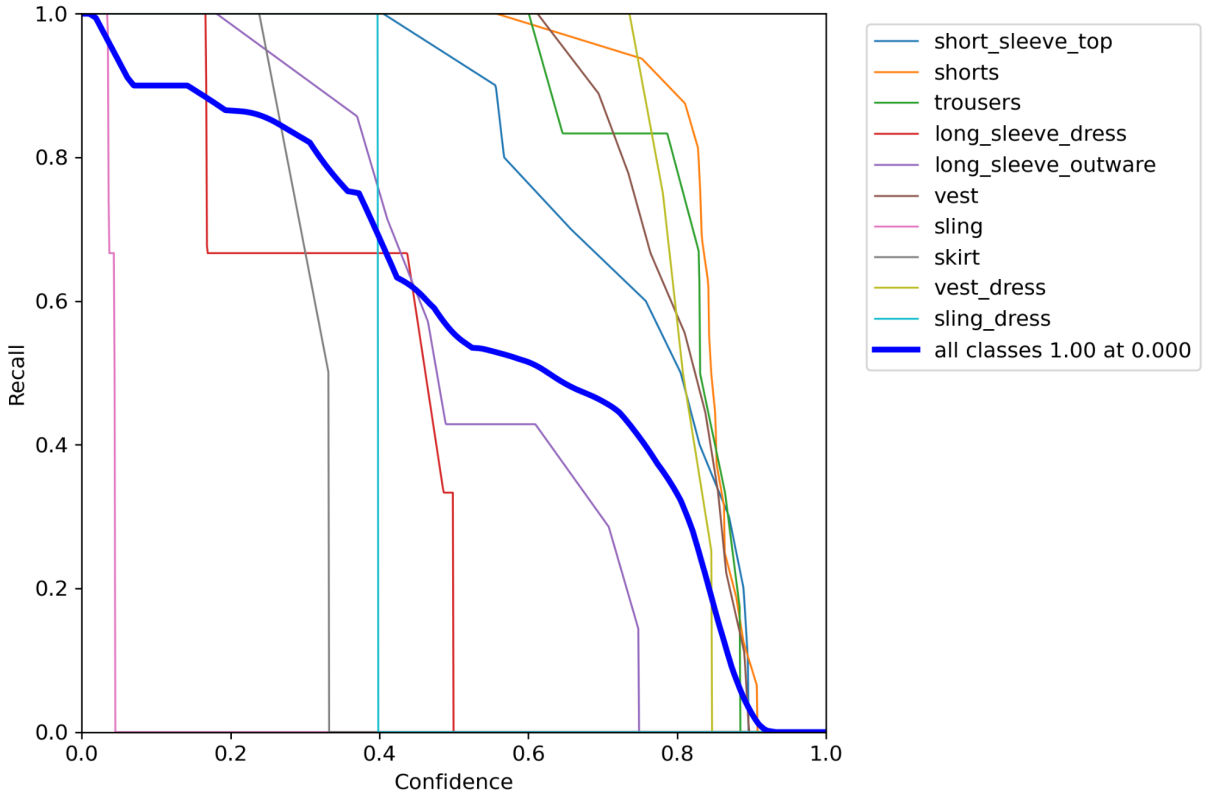
Confusion matrix :



Precision Curve :

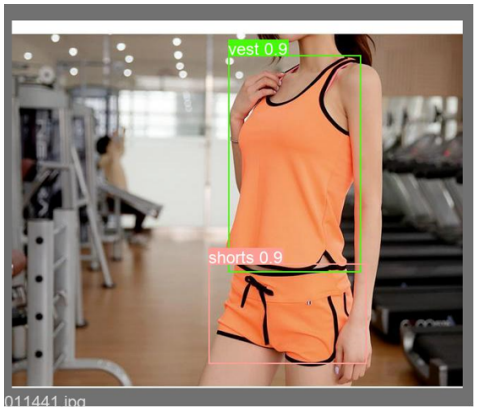


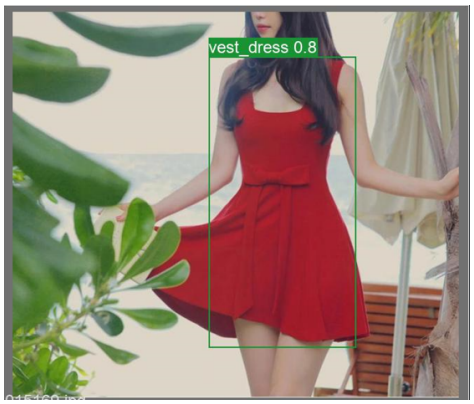
Recall Curve :



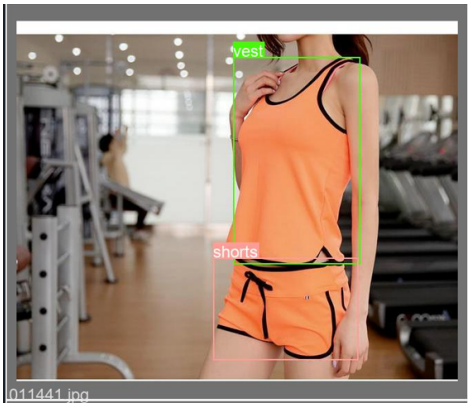
OUTPUTS

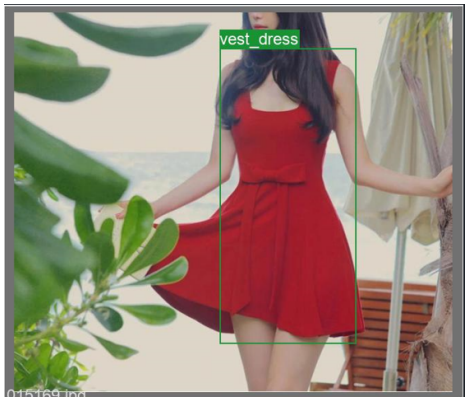
Predicted:





Real:





OUTPUTS FOR TEST INPUTS :

```
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
```

After training the data and running all epochs we get the weights of the trained data. We use the weights of the trained data for testing data.

```
!python detect.py --weights runs/train/exp/weights/last.pt --img 640 --conf 0.25 --source ../clothes_test.mp4
```

Colab Notebook :
https://colab.research.google.com/drive/1judGmEj60DXUXUzYJonkxoyjWgLbiriV#scrollTo=4vWUBHz_UD5U

References :

- GitHub link for YOLOV5 tutorial : https://github.com/ultralytics/yolov5
- Make Sense website for labelling images and creating dataset : https://www.makesense.ai/
- Youtube Tutorials for YOLO :
    1. YOLOv5 training with custom data
    2. What is YOLO algorithm? | Deep Learning Tutorial 31 (Tensorflow, Ker...
    3. TFOD 2.0 Custom Object Detection Step By Step Tutorial