CCE 2ND YEAR
AMRITA VISHWA VISYAPEETHAM,
CHENNAI CAMPUS

13 DECEMBER- 2022
THURSDAY
19CCE201

CAPSTONE PROJECT REPORT

on

DETECTION OF HUMAN INTERACTIONS USING YOLOV5

Submitted by

NITHIN NARAYANAN P B (CH.EN. U4CCE21037)

SNEHALAKSHMI S (CH.EN. U4CCE21057)

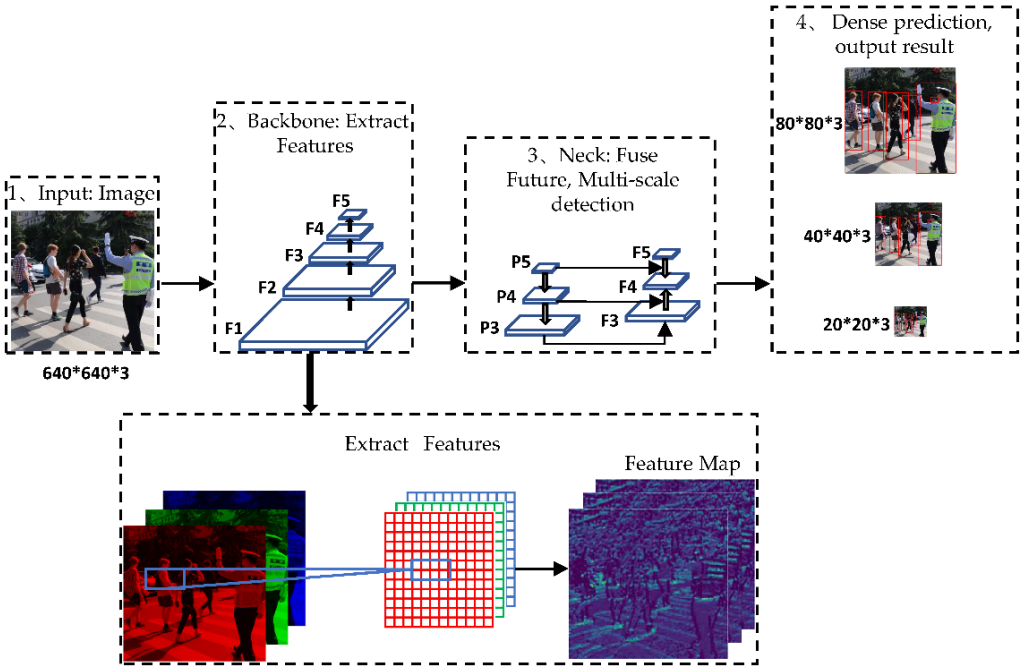
MAYANK MAITI (CH.EN. U4CCE21033)

Abstract

Human activity recognition has been widely studied and has recently gained significant advances with deep learning approaches. Through this experiment, we expect to understand how yolov5 uses input for training and testing and that later analysis the human interactions by fusing information from local images where the interaction actively occurred, primitive motion with the posture of individual subject's body parts. Human interactions change, depending on how the body parts of each human interact with the other. Here we capture the subtle difference between different interactions using interacting body part attention and classify them. The experimental results on six public datasets to recognise human interactions. The obstacle faced during gesture detection is unavailability of Quality Dataset.

Introduction

Our project is on building a gesture detector using Deep Neural Networks (DNN). The DNN employed here is a Convolutional neural network (CNN). Our model is trained using YOLOv5 which is written in the Pytorch framework. As YOLO v5 is a single-stage detector, it has three important parts like any other single-stage detector: Model Backbone, Model Neck and Model Head as shown below:



Model Backbone is mainly used to extract important features from the given input image. In YOLO v5 the CSP—Cross Stage Partial Networks are used as a backbone to extract rich in informative features from an input image. CSPNet has shown significant improvement in processing time with deeper network. Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalized well on object scaling. It helps to identify the same object with different sizes and scales. Feature pyramids are very useful and help models to perform well on unseen data. In YOLO v5 PANet is used for as neck to get feature pyramids. Path Aggregation Network (PANet) aiming at boosting information flow in proposal-based instance segmentation framework. We present adaptive feature pooling, which links feature grid and all feature levels to make useful information in each feature level propagate directly to following proposal subnetworks. The model Head is mainly used to perform the final detection part. It applies anchor boxes on features and generates final output vectors with class probabilities, accuracy and bounding boxes.

We are working on a Human interaction recognition framework based on interacting body part attention, the aim is to train the model Yolov5 with 6 different classifications ***{Hugging, Pointing, Handshaking, Kicking, Punching, Pushing},*** The interactions can be done by more than two members even then our model can classify. We are going to take extra Classification for our next steps. Gestures are numerous; For our convenience, we have considered 6 classes for our model. On average, we have taken 230 images for every class and classified them as training, validation and testing data – 80%, 19% and 1%, respectively. So, we have taken 184 images for training, 44 images for validation and 2 images for testing from each class. The LabelImg software is used to create annotations in YOLO format. The project's main aim is to attain an accuracy of > 91%. The working of a Gesture Detection Model based on Deep Learning is Feature extraction and detection based on the Extracted features. The Feature extraction process is done by stacks of Hidden Neural Network which extracts features at different levels by scaling each input with a corresponding weight which makes it significant or not. The output we receive after the Feature extraction is the Weights File. This is indeed the optimized weights which performs the Feature extraction efficiently. In any custom object detection model, the layers in the neural network that changes are the input layers and the Output layers. The feature extraction can be used multiple times. The optimized weights of the Features extraction layers are frozen which can be called for the user's wish using frameworks like TensorFlow and PyTorch. YOLOv5 does online augmentation during training, so we do not apply any augmentation steps in LabelImg for training with YOLOv5. But we will auto-orient and resize as the pre-processing steps.

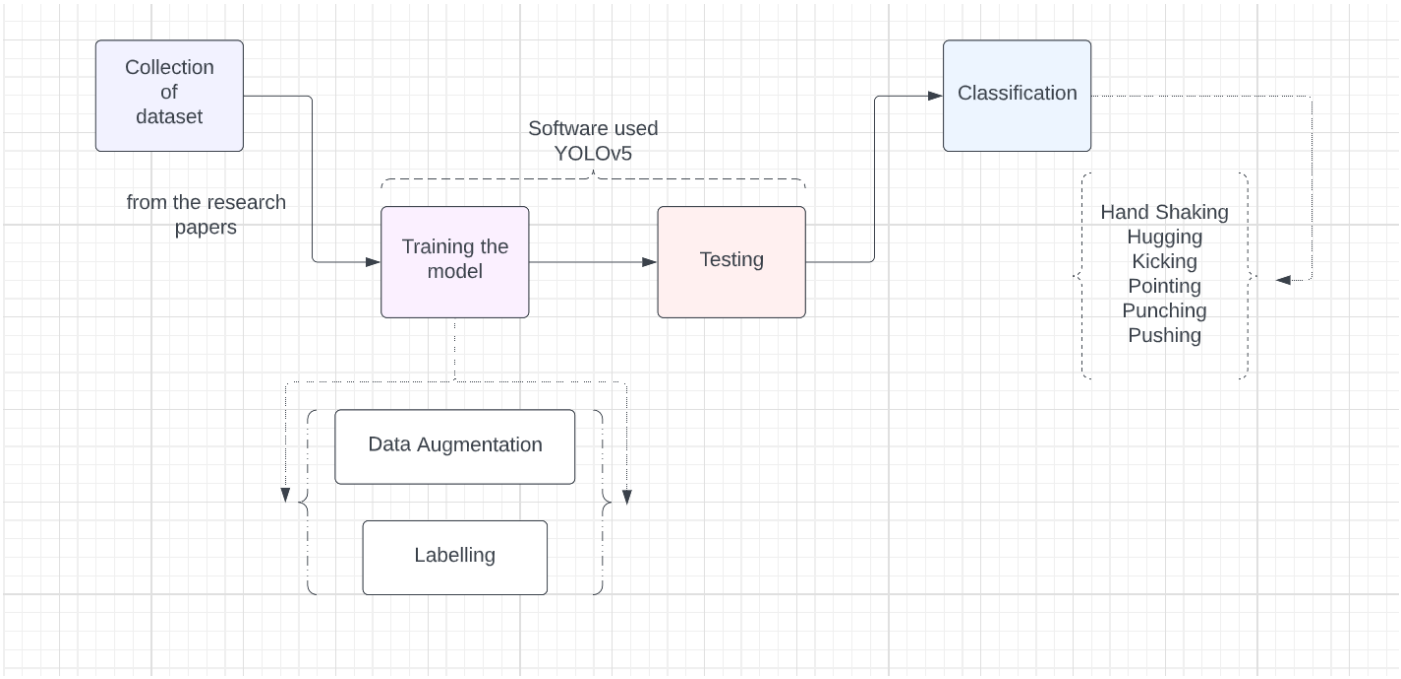
Dataset Configuration:

From each class:

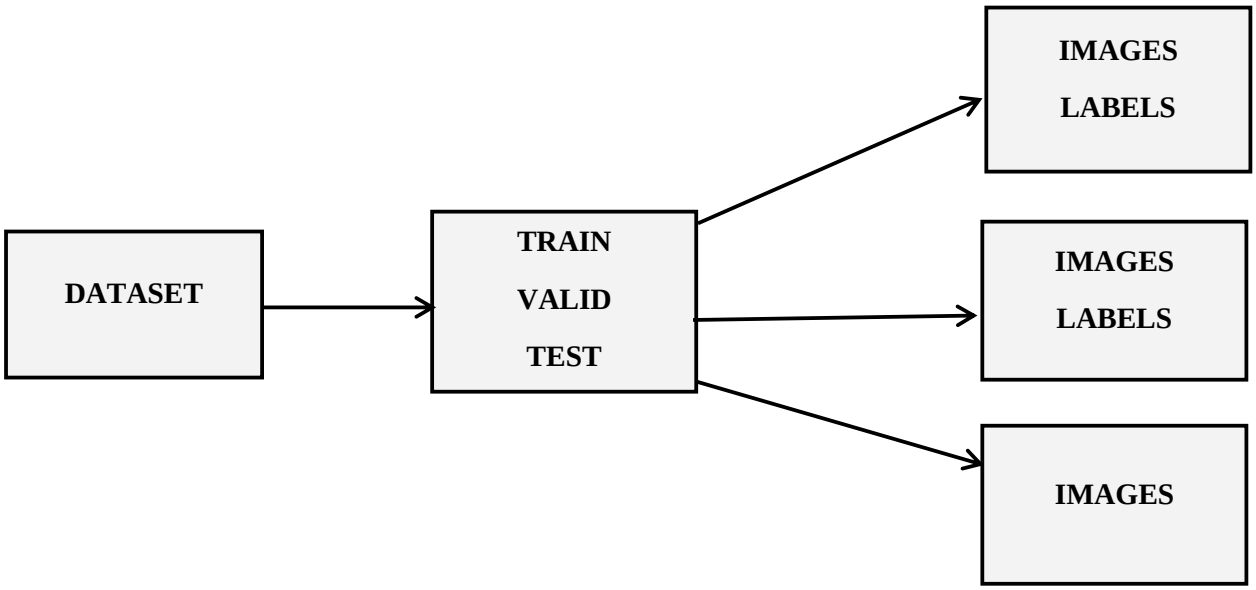
Total Images: 230 (on average)

Training Data	**184 images (80%)**
Validation Data	**44 images (19%)**
Testing Data	**2 images (1%)**

Methodology:



We will first begin by collecting datasets from an open library and then sort them for training and testing in the format given below, depending on the size of our input. Simultaneously, we will also be labelling them, followed by



The model is trained in the Google Colab by mounting the google drive. The dataset and the YOLOV5 repository are uploaded to the google drive. Inside the YoloV5 directory we place the yolov5s.pt which is the light weights file. The YAML file is edited by including the path of the images for training and validation. Also, we include the number of classes and list of all the class names. In order to train our model with the custom dataset under yoloV5 the provided python script train.py must be executed by providing the required arguments – weights file, YAML file, Image resolution, no. of epochs, Steps per epoch, Batch size in every step of an epoch, workers.

Weights file – yolov5s.pt, YAML file – coco128.yaml, Resolution – 640 x 640, number of epochs – 100, Batch size – 8, workers – 4.

```
!python /content/yolov5/train.py --data /content/drive/MyDrive/Capstone/coco128.yaml
--weights /content/drive/MyDrive/Capstone/yolov5s.pt --workers 4 --epochs 100 --batch 8 --img 640
```

Results:

- The results after training the model illustrates how efficiently the model has learned the features.

```
100 epochs completed in 0.837 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 16/16 [00:03<00:00,  5.04it/s]
                   all        247        247      0.897       0.86      0.921       0.57
             handshake        247         66      0.893          1      0.971      0.703
                   hug        247         56          1      0.764      0.995      0.588
                  kick        247         25      0.958      0.911      0.917      0.569
                 point        247         50          1      0.931      0.959      0.514
                 punch        247         14      0.586      0.607      0.718      0.501
                  push        247         36      0.948      0.944      0.964      0.546
Results saved to runs/train/exp
```

From the above figure

1. Precision signifies how accurately the model predicts the positives.
2. R (Recall) signifies how accurate or the bounded box create by the model compared with the ground Truth.
3. mAP50 signifies the mean average precision for the IoU (Intersection over Union) threshold 0.0 to 0.5 for the 100 epochs.
4. mAP50-95 signifies the mean average precision for the IoU (Intersection over Union) threshold ranging from 0.5 to 0.95.

The total results and graphs are provided in the GitHub repository:

[Human-gesture-detection-Results]

Further, our model is used in detecting some test images

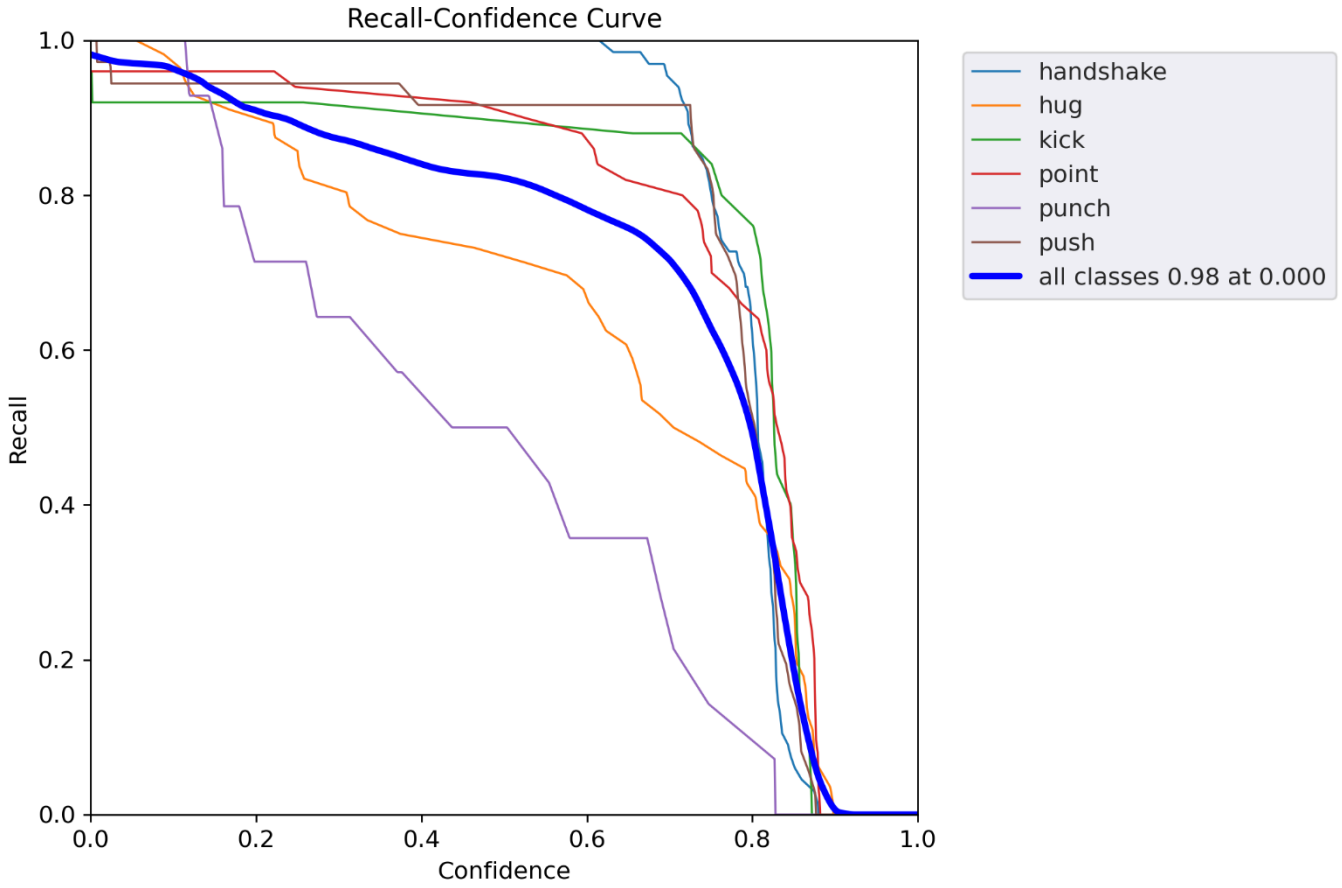
The results are:

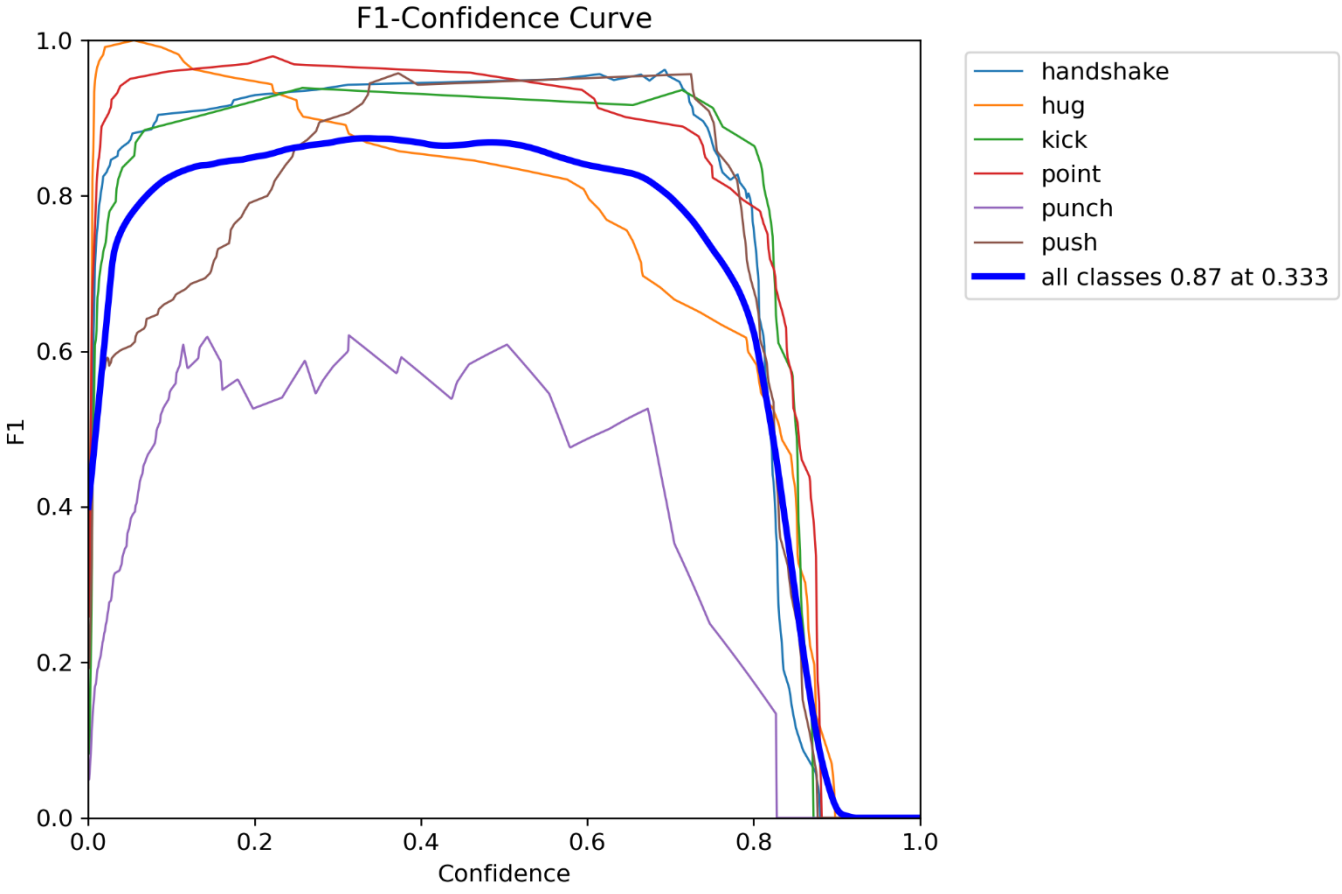

hug 0.88

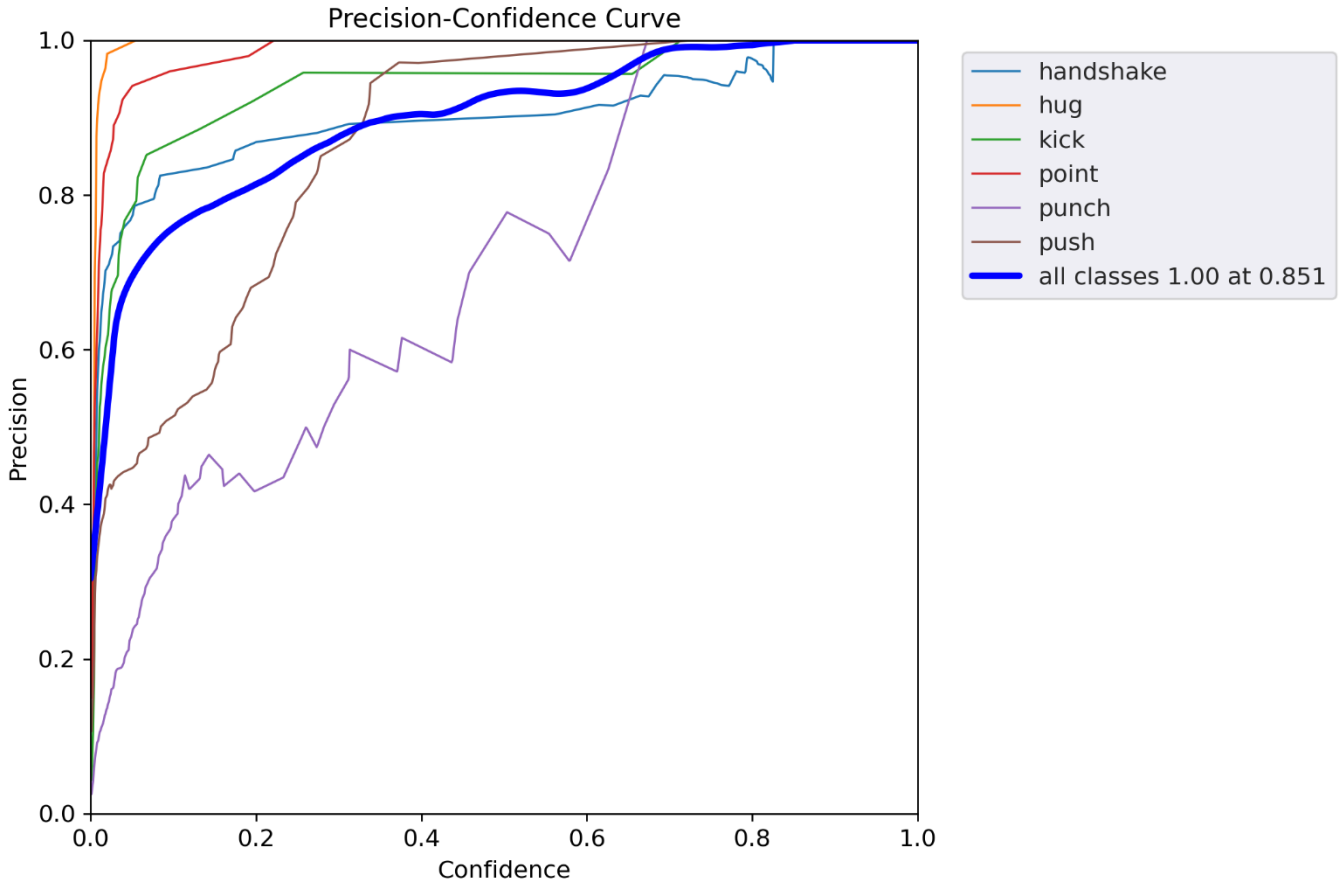

point 0.74

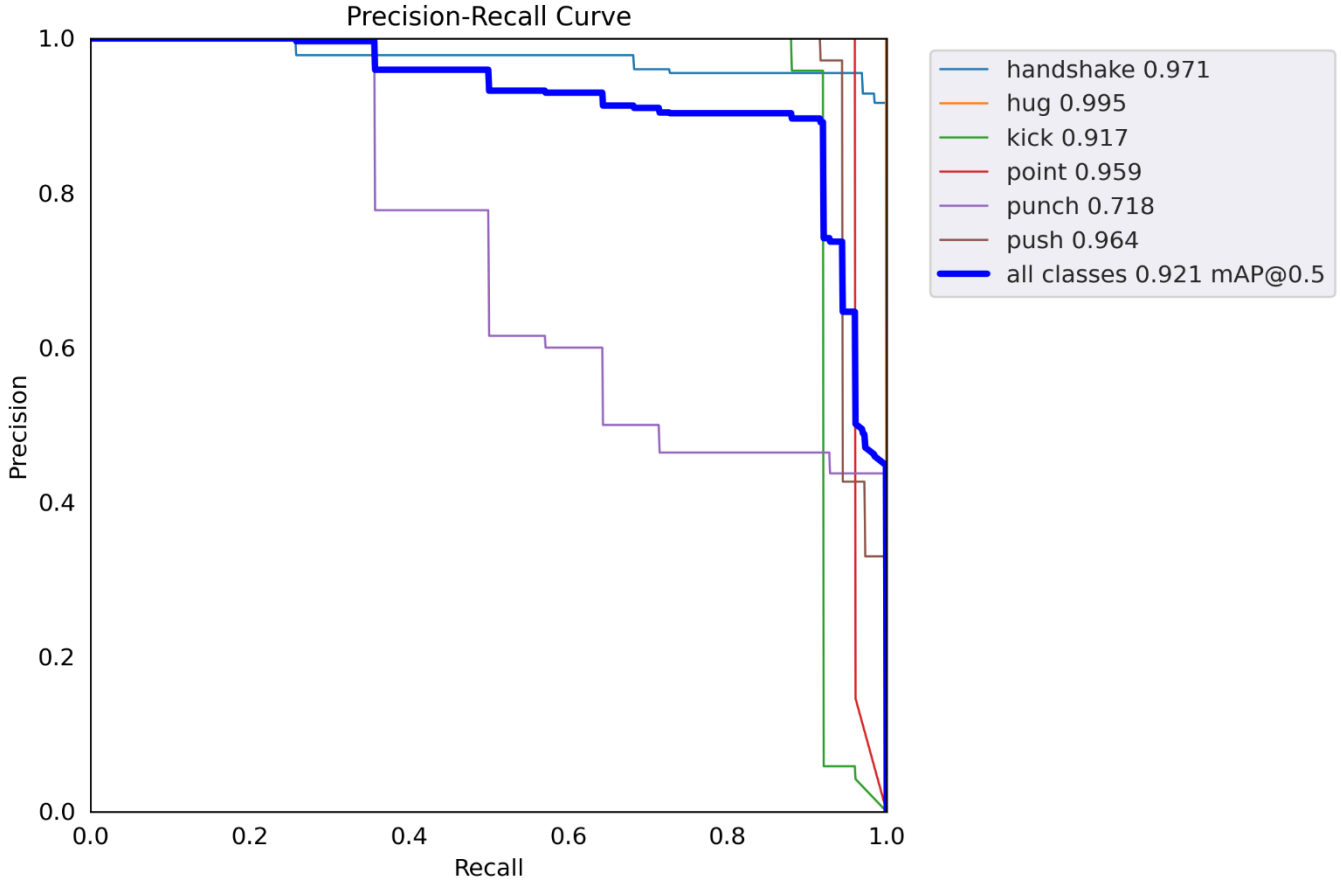

handshake 0.27

And the Graphs after training are given below:






Precision-Confidence Curve
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
handshake
hug
kick
point
punch
push
all classes 1.00 at 0.851


Precision-Recall Curve
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
handshake 0.971
hug 0.995
kick 0.917
point 0.959
punch 0.718
push 0.964
all classes 0.921 mAP@0.5

Conclusion:

In conclusion, our capstone project focused on the development and implementation of a gesture detection system using YOLOv5, a state-of-the-art deep learning model. We aimed to recognize and classify human interactions based on six predefined gestures: Hugging, Pointing, Handshaking, Kicking, Punching, and Pushing. The project utilized the capabilities of YOLOv5, a single-stage object detection model, with a specific focus on interacting body part attention. The model architecture included a CSPNet as the backbone for feature extraction, PANet for generating feature pyramids, and a final detection head for accurate prediction. We collected and labeled a dataset with 230 images for each gesture, distributed for training, validation, and testing. The LabelImg software was employed for annotation in YOLO format. Training the model took place in Google Colab by mounting Google Drive and uploading the dataset along with the YOLOv5 repository. The model underwent training for 100 epochs with a batch size of 8 and a resolution of 640 x 640. The results indicated the model's efficiency in learning features, with metrics such as precision, recall, mAP50, and mAP50-95 demonstrating the model's accuracy and performance. Additionally, the trained model was applied to detect gestures in test images, showcasing its real-world applicability. The project's GitHub repository includes detailed results, graphs, and further insights into the performance metrics. Despite the successes achieved, certain challenges were encountered during the project, including the unavailability of a high-quality dataset for gesture detection. Future work could focus on expanding the dataset, refining the model architecture, and exploring additional techniques to improve accuracy and robustness.