Problem Statement:

Build A Personal Safety Equipment Detection System

Dataset Details:

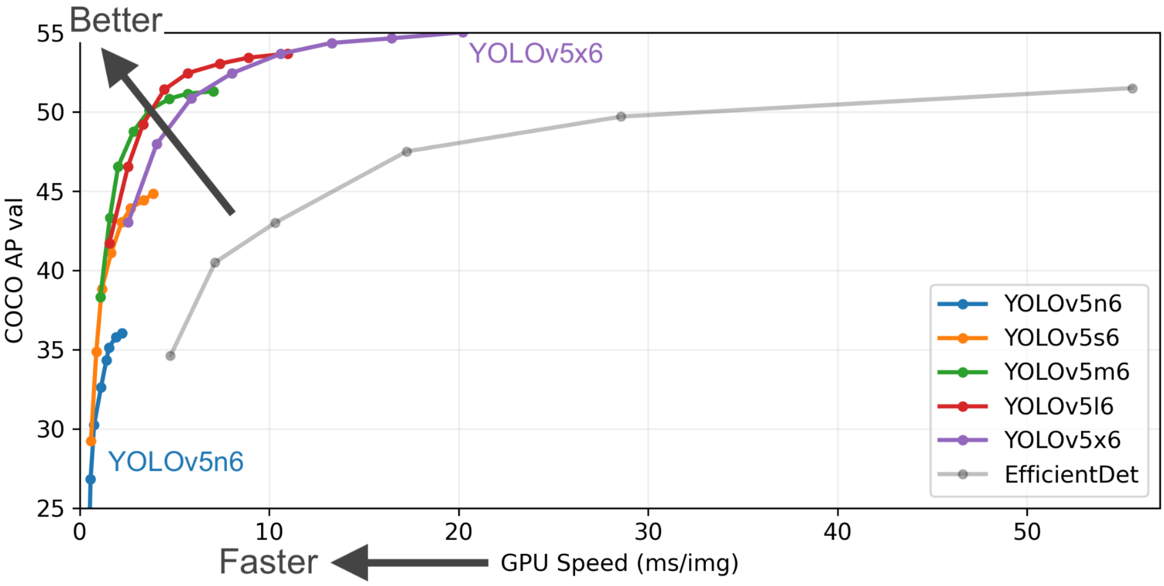
- Dataset annotation format: Pascal VOC (XMLs)
- Dataset Size:
  - → Data with Annotations: 4,750
  - → Data without Annotations (Test): 250
- Dataset Classes of interest:
  - → Head
  - → Hardhat

Model used:

- Transfer learning is used for the above task. The model used for the above task is **YOLOv5** (where YOLO is an acronym for *You Only Look Once*). It is a group of models which is trained on **COCO dataset.**

➔ Why YOLOv5?

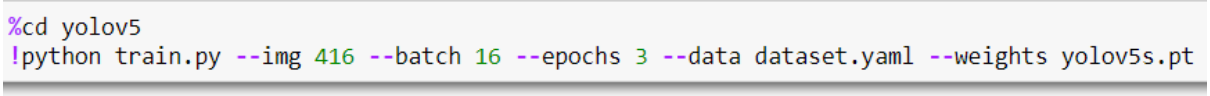
1. Easy to use and is faster than other algorithms.
2. Based on PyTorch framework.
3. Installation is easy, simple and straightforward(just clone the ultralytics YOLOv5 repository from GitHub)
4. Requires much less computational power than other architectures, while keeping comparable results.



5. Trained over 80 different real-time labels.
6. Training on custom data is hassle free and deploying the code for real-time purposes takes minimal effort.

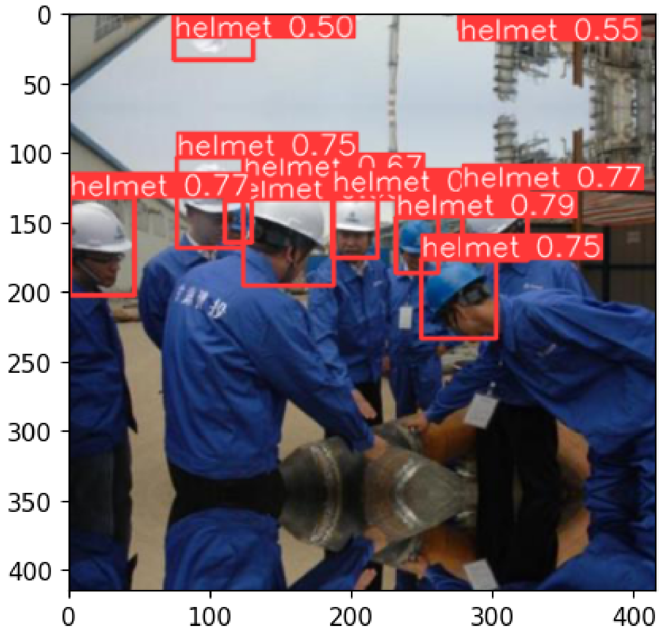
Stage 1 Evaluation – Deep Learning:

- **Stage 1 Video Link :** https://drive.google.com/file/d/1tn_VTRKUBxnd0MRFeUv5qpuy3hwduwcs/view?usp=share_link
- **Approach :**
  - Install the PyTorch dependencies
  - Clone the ultralytics YOLOv5 repository
  - Install the requirements for running the pre-trained model and import the dependencies
  - Convert the pascal annotations to YOLO annotations using this python script
  - Create a dataset.yaml file
  - Train the model using the below command:
    - --img 416 refers to the size of image
    - --batch 16 refers to the batch size
    - --epochs 3 refers to the number of epochs
    - --data dataset.yaml refers to the location of data and number of labels and name of labels
    - --weights yolov5s.pt refers to the pre-trained weights of the model to be tweaked

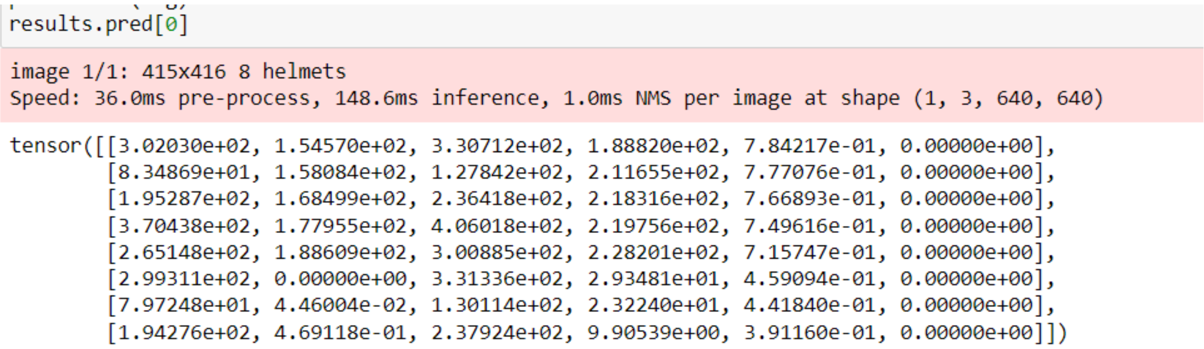


```
%cd yolov5
!python train.py --img 416 --batch 16 --epochs 3 --data dataset.yaml --weights yolov5s.pt
```

  - Load the pre-trained and plot the results on test image:



- Fetch the coordinates of the bounding box, confidence score and label using the ***.pred*** attribute.



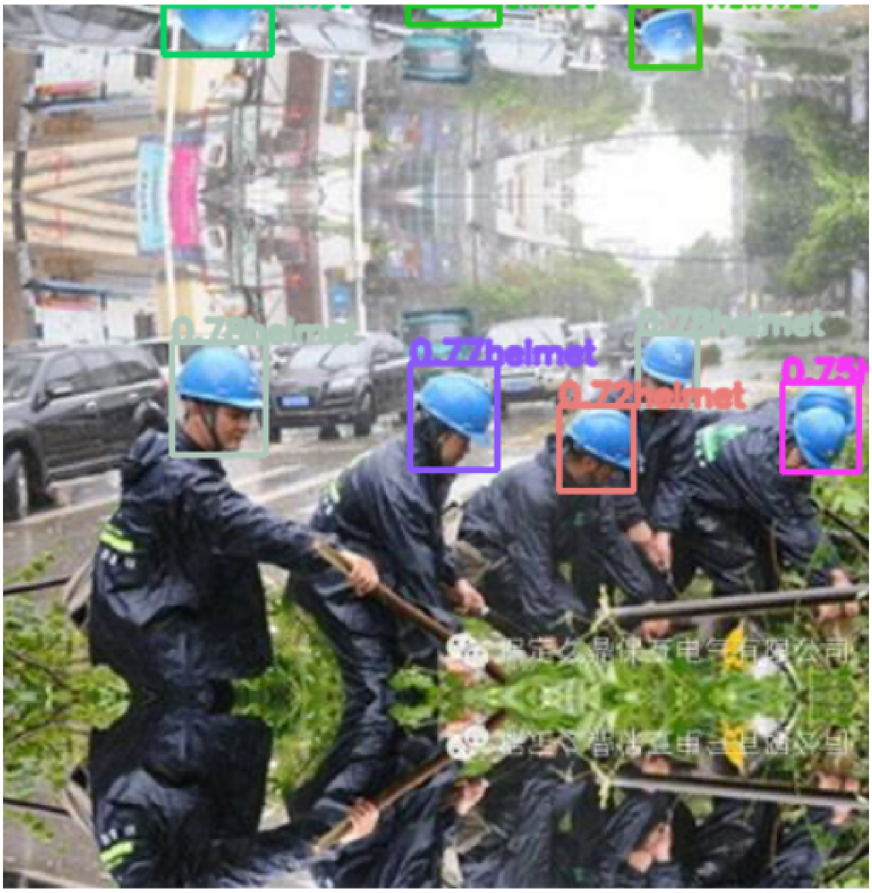
```
results.pred[0]
```

```
image 1/1: 415x416 8 helmets
Speed: 36.0ms pre-process, 148.6ms inference, 1.0ms NMS per image at shape (1, 3, 640, 640)
```

```
tensor([[3.02030e+02, 1.54570e+02, 3.30712e+02, 1.88820e+02, 7.84217e-01, 0.00000e+00],
        [8.34869e+01, 1.58084e+02, 1.27842e+02, 2.11655e+02, 7.77076e-01, 0.00000e+00],
        [1.95287e+02, 1.68499e+02, 2.36418e+02, 2.18316e+02, 7.66893e-01, 0.00000e+00],
        [3.70438e+02, 1.77955e+02, 4.06018e+02, 2.19756e+02, 7.49616e-01, 0.00000e+00],
        [2.65148e+02, 1.88609e+02, 3.00885e+02, 2.28201e+02, 7.15747e-01, 0.00000e+00],
        [2.99311e+02, 0.00000e+00, 3.31336e+02, 2.93481e+01, 4.59094e-01, 0.00000e+00],
        [7.97248e+01, 4.46004e-02, 1.30114e+02, 2.32240e+01, 4.41840e-01, 0.00000e+00],
        [1.94276e+02, 4.69118e-01, 2.37924e+02, 9.90539e+00, 3.91160e-01, 0.00000e+00]])
```

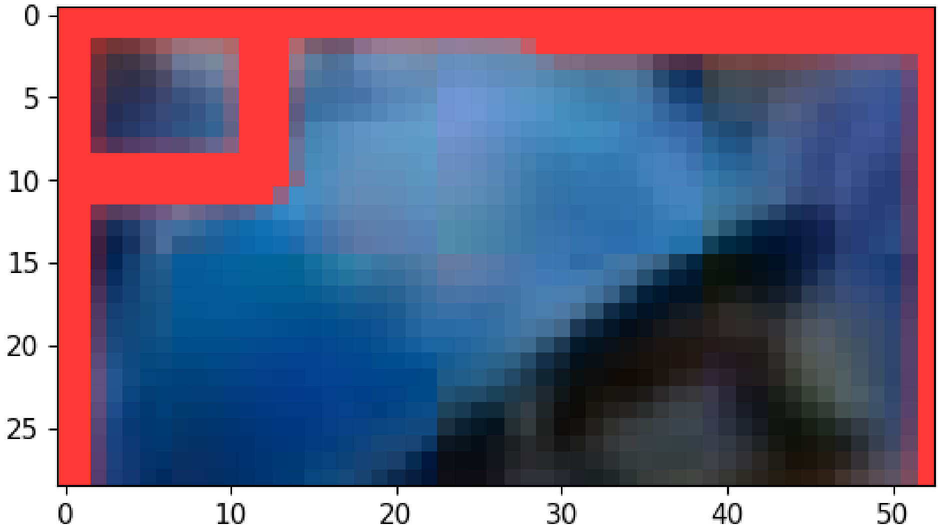
.pred output

- Each frame will have a set of confidence scores. These scores are stored in python set to remove the duplicates and hence converted to a dictionary where the key of dictionary is confidence score and value is a numpy array generated randomly to give a RGB value
- Youtube video is read and stored as frames. Each frame is processed and hence varying colour of the bounding box according to confidence score is accomplished.



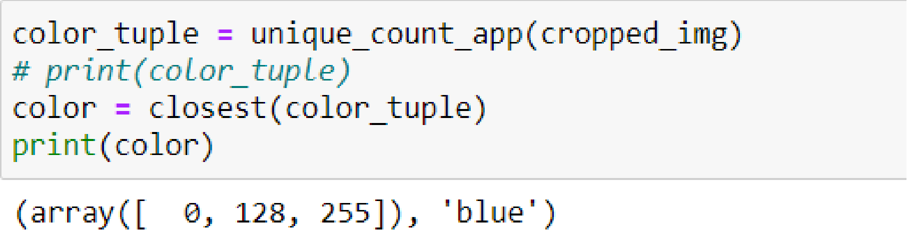
Stage 2 Evaluation – Find the Colour:

- **Stage 2 Video Link :** https://drive.google.com/file/d/1wROlLYVQE79cn68n7XtIRMmhi55Ttlfs/view?usp=share_link
- **Approach :**
  - Load the trained model using the weights created during training
  - Pass the image or frame to the model for prediction
  - Once the bounding box is created, fetch the coordinates using results.xyxy
  - Crop the region of interest to the upper half because the helmet is worn in the upper part of the face.



Cropped the Region of interest

  - Pass the cropped region of interest as an input to find the the colour of helmet
  - Occurrence of every distinct value of each colour channel,i.e., Red, Green and Blue are calculated.
  - The value with maximum count is selected as the particular value of that colour channel.



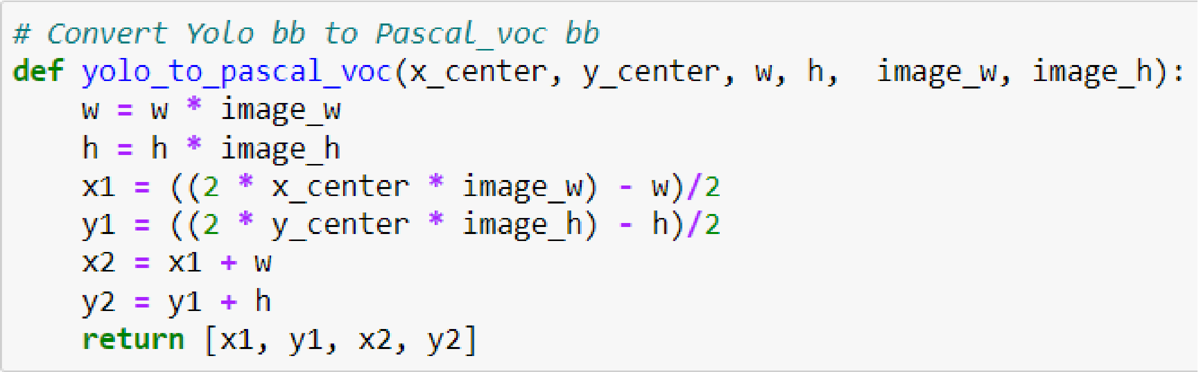
```
color_tuple = unique_count_app(cropped_img)
# print(color_tuple)
color = closest(color_tuple)
print(color)
```

```
(array([  0, 128, 255]), 'blue')
```

Getting the colour of helmet

Stage 3 Evaluation - mAP :

- **Stage 3 Zip Folder Link :** https://drive.google.com/file/d/1WDMklvTtsNOWpMfjwrPKcTFaTNCgsVyI/view?usp=share_link
- **Approach :**
  - Load the trained model using the weights created during training
  - Pass the image to model for predictions
  - Get the bounding box coordinates and detected class using .xywhn
  - Pass the values to this function



```python
# Convert Yolo bb to Pascal_voc bb
def yolo_to_pascal_voc(x_center, y_center, w, h,  image_w, image_h):
    w = w * image_w
    h = h * image_h
    x1 = ((2 * x_center * image_w) - w)/2
    y1 = ((2 * y_center * image_h) - h)/2
    x2 = x1 + w
    y2 = y1 + h
    return [x1, y1, x2, y2]
```

  - Store the values returned by this function and write to the .xml files to get the desired pascal voc annotations