Hard Hat Detection

Stage- 1

A project for the detection of hard hats in workplace areas with a given data

First, start with object detection.
For the part of object detection, I used two different models

1. Tensorflow Object detection API

This is an object detection API released by TensorFlow, which accepts the data in the format of tf.record

So let's start with code documentation here -

1. First, we import our necessary libraries
2. Cloning the required repository
3. Installing the object detection API and testing whether it is working or not
4. Unzipping the data
5. Then making two directories of train and test data, respectively
6. Now we need to make the CSV file of annotations and a pbtxt file for showing the different classes and their IDs
7. Then we analyze the different numbers of detection classes
8. We make the train and test records using the generatetfrecord.py from our model directory.
9. Now we need to select a pre-trained model to start with, for this purpose, we will be using ssd_mobilenet
10. Then we need to move this into the folder where our data is stored
11. Training the model using the pipeline config path and the model directory with batch_size=4 and 50000 steps
12. After training, we can get the model in the form of an inference graph
13. Now move to use the saved model file of our TensorFlow
14. Cloning the same repository again and installing object detection API
15. First, let's run it on a single image
16. Defining the image
17. Loading the saved model
18. Now loading the pbtxt file, which contains the classes
19. Converting the given image into numpy format
20. Making the tensor format and then detecting through it
21. Visualizing the detection images on the image,

We can see that the bounding box is much larger than the helmet.
23. I also tried training it on the given youtube it was taking more time, and in between, my google colab crashed, so let's change our model.

2. YOLOv5

Yolov5 is an advanced object detection model which is much faster to train and gives good results than other models.

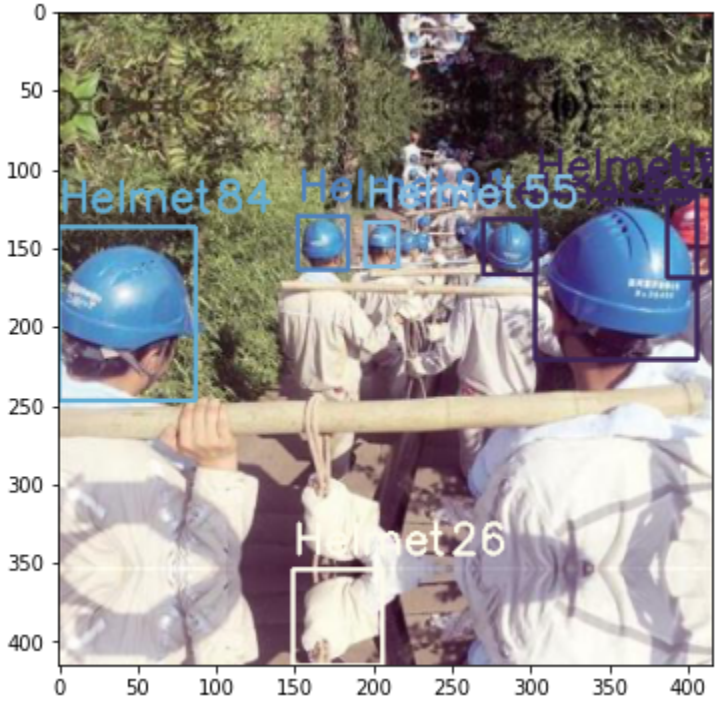
It takes data format in the annotations of the text file and a YAML file to navigate through the train and validation files.

1. Importing the necessary libraries
2. Cloning the repository of YOLOv5
3. Installing the requirements of YOLOv5
4. Now making two directories of labels and images for storing the image and labels.
5. Defining the detection classes
6. Now defining the function of annotations for the box into x,y,h, and w
7. Now the XML file needs to be converted into the label file and written there
8. Now the XML data is needed to be read to get the coordinates and then converted to the text format using the above function.
9. Now, write the annotations in the labels file.
10. Splitting the data into train and validation data in the ratio of 80 and 20
11. Copy_data for writing the storing the images and annotations in train and Val dataset. We need to check for the image to see whether or not a corresponding annotation exists. If there is a label, we can store them in the folder.
12. Creating the YAML file.
13. Now train the YOLOv5 on data using the yolov5s model.
14. Cloning the repository of YOLOv5
15. I trained the model for ten epochs using the weights of yolov5
16. The best model is now saved for further use
17. I then used the model on that youtube video for the results.
18. The FPS on the video is 30, as we can check through the cv2.
19. Detected video link- https://drive.google.com/file/d/1Hp9kkKZ9iDByP1a3nhjNi4McjcMShxgq/view?usp=share_link
20.

Stage 2

In this we need to change the detection color box according to the color of the helmet. So we will need to first find the region of interest and then find dominant color inside it.

1. First we need to export our yolov5 weights as a pytorch model
2. For confirmation that model is working fine we just predict it on our image
3. After getting the output we can export it as pandas dataframe

4. Now we need to make a function for finding the dominant color in that region of interest
5. Making a function named run_on_single_image
6. This function will make the first copy of the image and then predict it using our model
7. Now iterating through all pandas dataframe and checking for the detected class
8. If the detected class is a helmet, then extracting all the coordinates makes a cropped image of it and then finds the dominant color using our function
9. Extracting all the dominant colors as the numbers
10. Then we can put the bounding box around the helmet according to our dominant color and also put it across the percentage score
11. Doing the same for head and person object detection classes
12. Now returning to the final image
13. I ran this function for a single image, and the output came like this
14. 

15. As we can see that the bounding box is of same color as teh helmet in the above image
16. Now for the video processing part we opened the image as a cv2 object
17. Getting different frames as a single image and then processing our model on it and then writing it back to the video output
18. But the google colab is just keeping crashing in between so I am unable to share my video

Stage 3

I first used the whole model on all the folders and get the annotations and then get the annotations in the format of CSV, but later I tried to convert into XML annotations but the format got changed so I am just attaching the CSV file here.

Thanks