Assignment Report

Problem Statement:

- **Task**: Build a personal safety equipment detection system

Dataset Details:

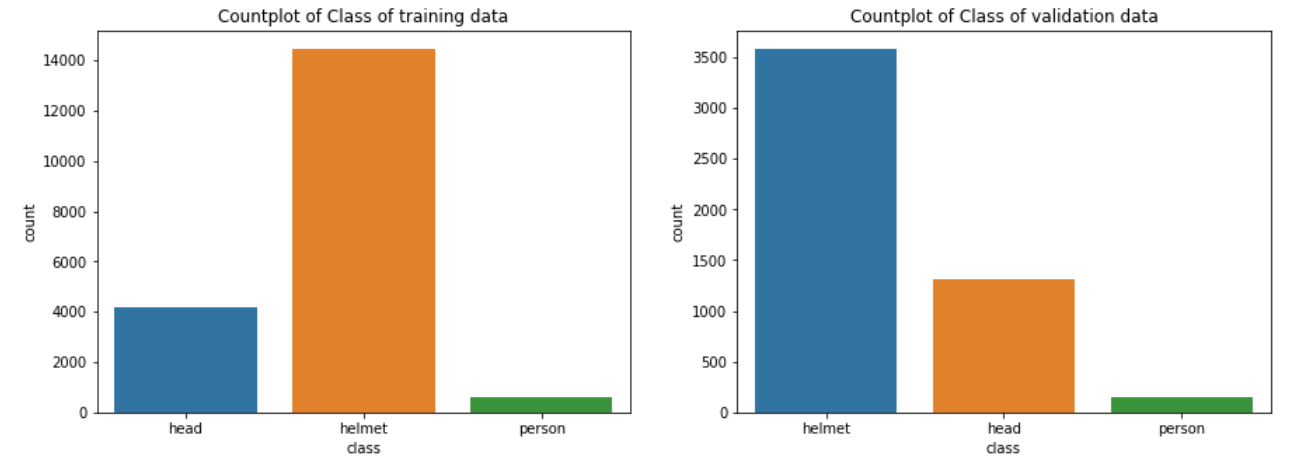
In this given problem, there are basically 2 categories present as hardhat and head. Around 4750 images are having this categories present along with annotations.

There is separate 250 test images are given to test the model. Whether is it detecting the object contain of the above categories or not? But along with inference on it we need to create respective annotation file for the test image.

Dataset Cleaning:

While wrangling with dataset, came across ***one new class as person*** present in some annotations files. Images with those class are few in counts. And also as we are detecting head and hardhat (helmet) in our problem so we don't require this class on which model to be train. We can remove those images from our dataset.

```
fig, ax = plt.subplots(1, 2, figsize=(15,5))
sns.countplot(x = 'class', data = df, ax=ax[0])
sns.countplot(x = 'class', data = df_val, ax=ax[1])
ax[0].set_title("Countplot of Class of training data")
ax[1].set_title("Countplot of Class of validation data")
plt.show()
```



There **are total 153 images contains the person annotated class** which needs to remove. After removing the images and respective annotation files with person class. Now the dataset is only having two class objects in the images as per our requirement of the problem statement.

After removing those images, we have total 4597 images to train the model on. I split those images into train and validation set with its annotation file as 80:20 rule.

Model Building:

I am using the transfer learning approach for this problem, using pytorch library **Detectron2** for this task. Since this problem having detection object of small scale with foreground and background ratio imbalancing. So decide to use **Retinanet** with Resnet50 as backbone on top of it a FPN (feature pyramid network).

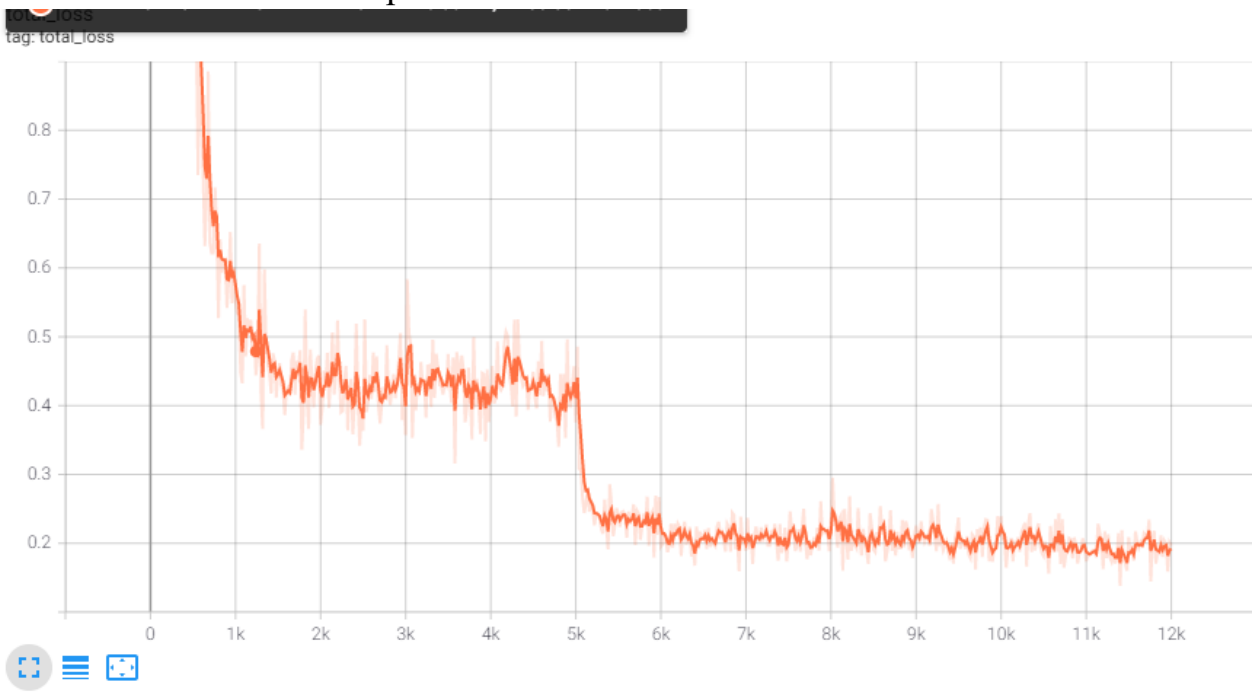
In the respective paper they mentioned about foreground-background imbalance ratio encountering while training the dense object detectors is the one of the prime reason of low accuracy compared to two stage detectors. To counter this problem they introduced **Focal loss** which is nothing but dynamically scaled cross entropy loss.

I trained this model for 12000 thousand only. We can train this model further to improve the mAP. Since I'm using colab gpu to train the model needs to use the colab gpu. Because after some usage of gpu colab restricts us to use gpu session for approximately 1 and ½ day. So I can train this model with such low epochs.

During training we are validating the model after 500 epochs for 8000 steps then change validation period for 1000. After completion we get **6.8 AP (average precision) for helmet and 5.08 for head**.

```
[08/01 14:30:54 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.20 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.059
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.102
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.064
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.044
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.087
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.312
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.014
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.056
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.068
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.051
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.098
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.342
[08/01 14:30:54 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
|:-----:|:------:|:------:|:-----:|:-----:|:-----:|
| 5.946 | 10.160 | 6.396  | 4.388 | 8.696 | 31.219 |
[08/01 14:30:54 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category   | AP     |
|:-----------|:-------|:-----------|:-------|
| helmet     | 6.809 | head       | 5.082 |
OrderedDict([('bbox',
              {'AP': 5.945926277033074,
               'AP-head': 5.082414707228105,
               'AP-helmet': 6.8094378468380405,
               'AP50': 10.160260439489425,
               'AP75': 6.3955769883763285,
               'APl': 31.218669632159003,
               'APm': 8.695993095404743,
```

Total Loss after 12000 steps



Inference on some images:

XML generation for test image:

As per instructed, XML's for each test image is generated. To create this need to write small loop using **xml** library**.**