# Facemask detection model using MMDETECTION

Coursework

Novosibirsk State University

28 May 2020

Mukul Kumar Vishwas

Advisor : Aleksey Okunev

## Content

- Problem Statement.
- What is MMdetection.
- General Model
- Resnet with FPN
- Comparison
- Model
- Training and Testing
- Annotation
- Result
- Future task.

## Problem Statement

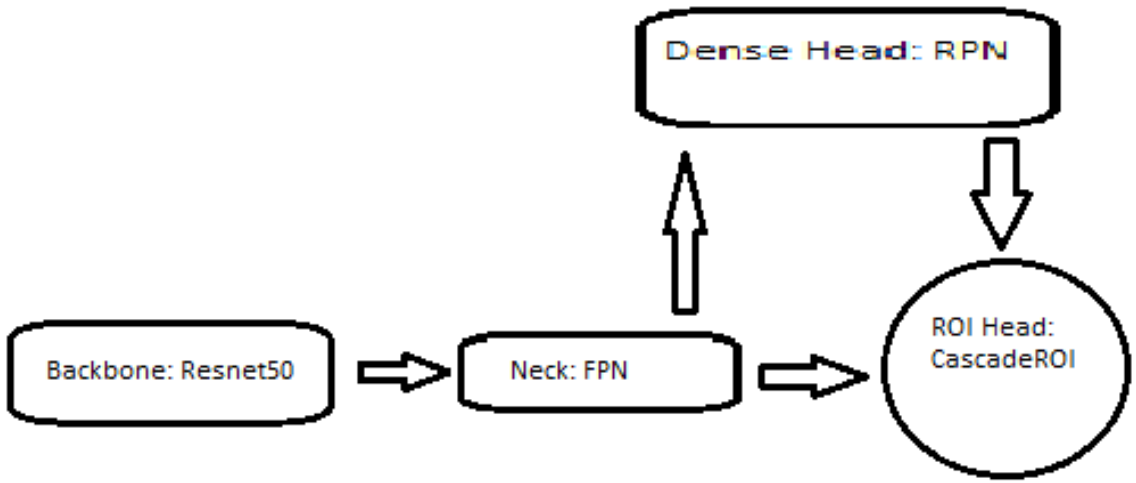
- COVID-19 pandemic changed our life. It is deadly and difficult to find decease cost more than 300 thousands of lives to date. On the other hand, following some simple rools can help to control the infection. The goal of this work is to create a model and train neural network to discriminate peoples who follow the sanitary rules from those who are violating them...

## What is MMdetection

- 'MMdetection' is an open source object detection toolbox based on PyTorch.
- It is faster.
- different detection framework's can be used to customize our model.
- It supports multiple Datasets like XMLstyle, COCO, PASCAL etc.

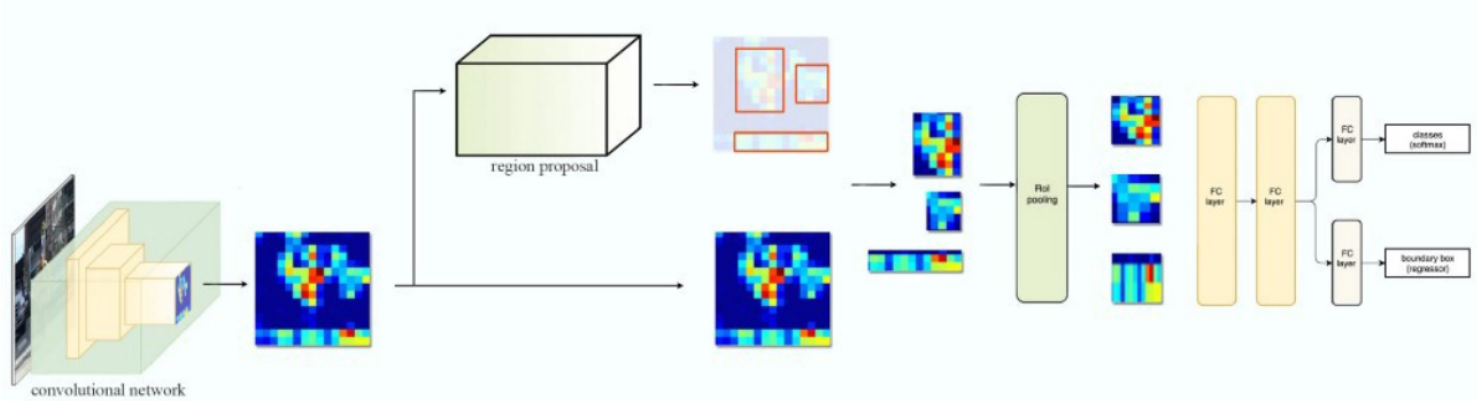
## Model

- This model used resnet50 for feature extraction without the last fully connected layer.
- Feature pyramid N/w do refinements of the raw feature extracted by backbone.
- Dense head (RPN) operates on dense location of feature map.
- RoIHead is the part that takes RoI features as input and make RoI-wise task specific predictions, such as bounding box classification or regression, mask prediction.

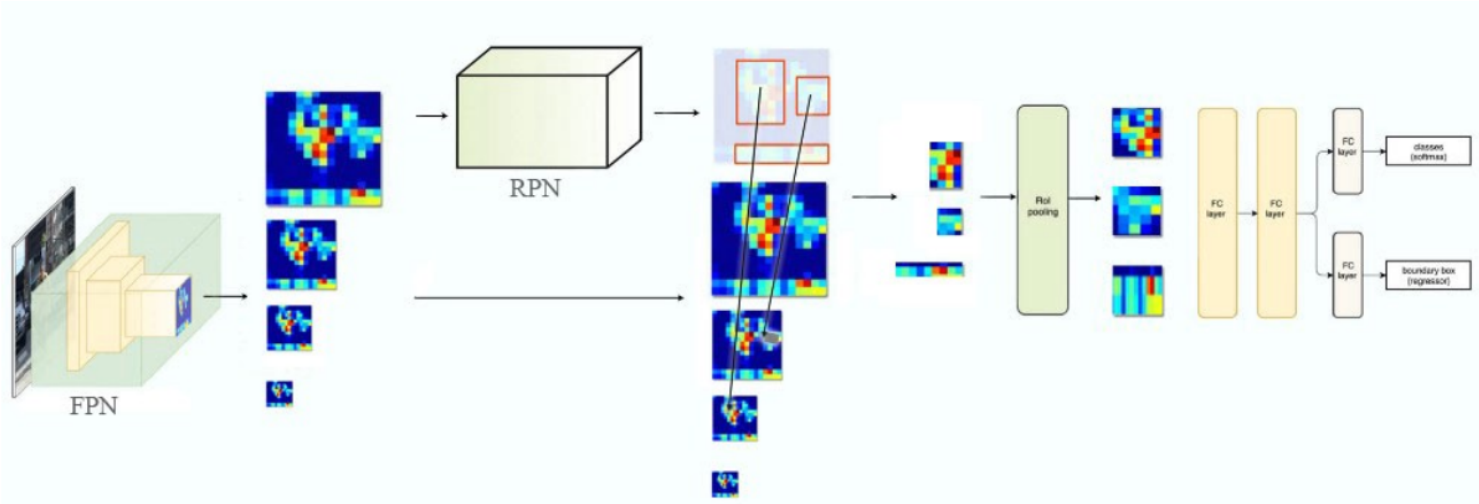


# Resnet with FPN

- Without FPN




- With FPN



# Comparison

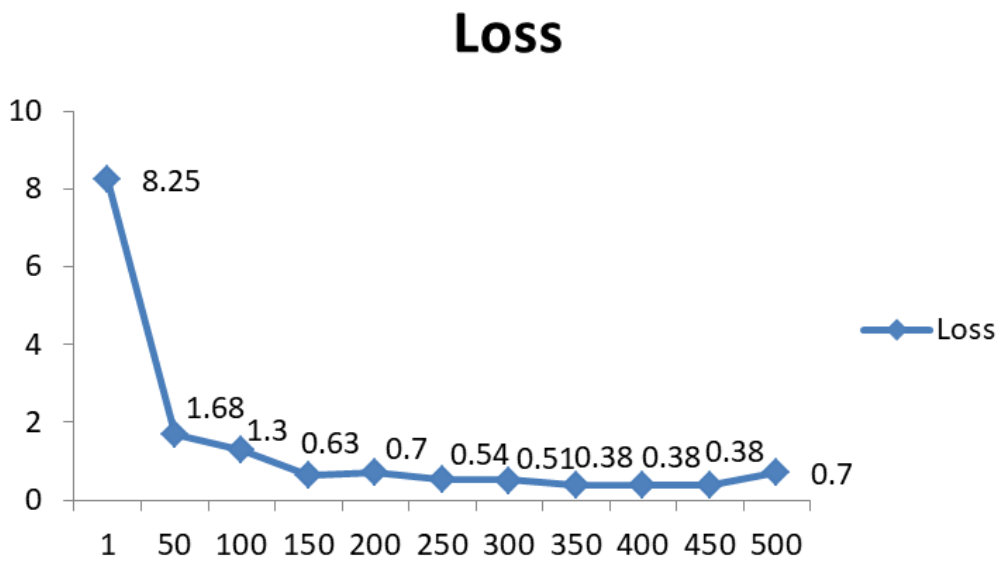
| Feature | Without FPN | With FPN |
|---|---|---|
| Training time | Slow | Increased |
| Dataset | Big | Small |
| Test/Val time | high | Decreased |
| Accuracy | Good | Increased |
| AR | 44.9 | 56.3 |
| Inference time | 0.32 | 0.148 |

AR(Average recall): the ability to capture objects.

IR: Time taken for prediction.

## Model

- Backbone: Resnet50, Neck: FPN, RPN Head: RPN Head
- ROI head: CascadeRoIHead
- bbox roi extractor: Shared2FCBBoxHead
- mask roi extractor: FCNMaskHead
- Loss during Traing



## Training and testing

- Optimizer= SGD, lr= 0.02
- Process of Training.

```
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(1333, 800), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),
]
```

- Process of Testing.

```
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(1333, 800),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ]
```

## Annotation

- In this model VGG Annotation tool was used for the Image annotation.

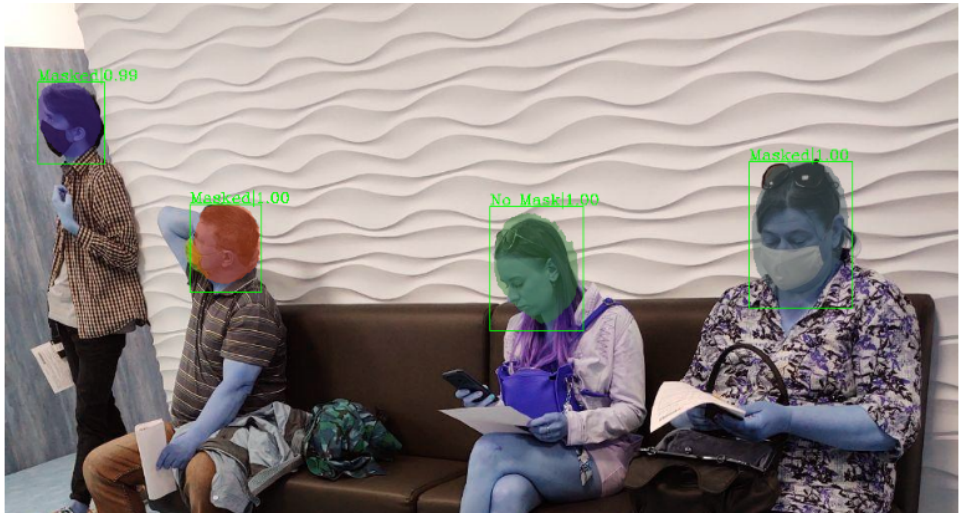
```
In [283]: num_imgs = len(coco['images'])
          ind = np.random.randint(num_imgs) + 1
          img = visualize_coco_img(coco, 133, img_dir)
          pil = Image.fromarray(img)
          pil
```

```
/home/data/Img/stream_img.jpg
2 ['No Mask']
2 ['No Mask']
1 ['Masked']
1 ['Masked']
2 ['No Mask']
2 ['No Mask']
```

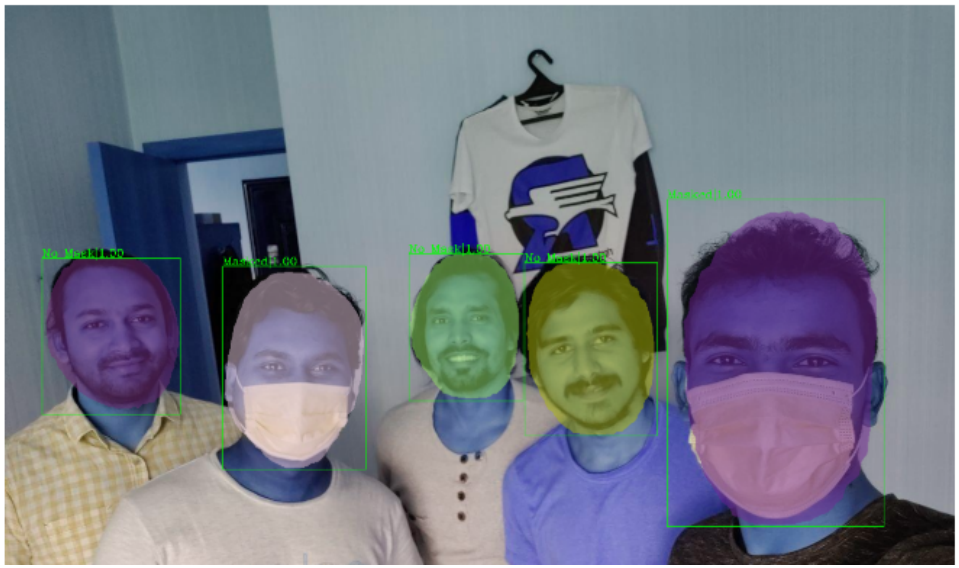
Out[283]:

# Result

- Threshold set to 90% Accuracy.




```
pil = Image.fromarray(img_masked)
pil
```



## Future Task

- Train the model with more data.
- Add more class so it can identify different face covers.
- Correctness of mask wearing
- Stream on live video

# Working hours decomposition

| The tasks | Description | Hours |
|---|---|---|
| Labelling images | Faces are to be annotated using Polygon shape from http://www.robots.ox.ac.uk/~vgg/software/via/via.html Faces should be classified into two types of attributes^ clear and masked. At least 5000 of faces are to be annotated. All faces on the image. Images are to be taken from here https://yadi.sk/d/Av-g9G3yVltqRA | 5000 * 0.01 = 50 hours |
| Converting to COCO mmdetection format | Annotations are to be exported as COCO format. It should further be converted to mmdetection COCO format in order import to go smooth | 8 |
| Splitting dataset into train and test | Dataset is to be split into test and train parts on the image basis. Nearly 4000 annotations for train, 1000 for the test | 4 |
| Understanding mmdetection framework | Getting familiar with https://github.com/open-mmlab/mmdetection | 16 |
| Tests | Setting test tools and test metrics | 16 |
| Optimization | Search for optimal hyperparametres for train and inference stage | 32 |
| Making cloud service | Publication of weights on cloud service http://hub.ci.nsu.ru:7700/ | 16 |
| Literature search | | 32 |
| Comparison with current solutions | | 16 |
| Writing coursework notes | Literature review Comparison with other solution (numerical metrics if possible) Technical notes on dataset created and other tools developed during coursework | 24 |
| Total | | 214 |

Thank you for your attention.