Facemask Detection using MMdetection Toolbox

Mukul Kumar Vishwas
Department of Mathematics and MechanicsNovosibirsk State University, Novosibirsk, Russia

Petr Menshanov
Novosibirsk State University
Novosibirsk, Russia

Aleksey Okunev
Vice director
NSU Higher College of Computer ScienceNovosibirsk, Russia

Abstract:- This paper described and compare between two object detection model for face mask detection. Using object detection we can predicts if the person on the picture wearing the mask correctly/incorrectly or not, in the current situation this model is extremely useful as this simple precaution will help to stop the spreading of deadly Coronavirus. In this paper, a comprehensive description of two operational and functional model was discussed how the data flows in the model and the type of operation performed on that. Additionally how the input data annotated and the result. The result was described usingthe mAP metric.

Keywords:- *MMdetection, Detectron2, COVID-19, Object Detection, Coronavirus, mAP.*

I. INTRODUCTION

The COVID-19 pandemic also widely known as coronavirus pandemic caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-Co-2). The World Health Organization instantly declared the outbreak on 30 January, and a pandemic on 11 March[1], It completely affected social and economi- cally and cost more than 1 million of lives[9]. On the other hand, by following some simple rules (mentioned below) the spreading of this virus can be stopped:

- Face masks and respiratory hygiene.
- Social distancing.
- Self-isolation.

The goal of this work is to compare two object detection model and train the best neural network to discriminate between peoples who follow sanitary rules like wearing the face mask properly from those people who are violating them by following this simple rule this virus can be stopped from spreading, until the vaccine was created these are our only option to be safe. As till October 321 vaccine candidates are developing vaccine but none of them are able to complete clinical trials to prove it's safety and efficiency[10]. Described model should able to perform below task's:

- Identify the position of the person.
- Identify person with correct mask, no mask or incorrectmask.
- A boundary region with the probability/confidence of themodel's prediction (varies from 0-1).
- A segmentation mask on the confident region.

II. MMDETECTION

A. *What is MMdetection*

As per the paper published by K.Chen in 2019 *"MMdetec- tion is an object detection toolbox that contains a rich set of object detection and instance segmentation methods as wellas related components and modules[2]"*. Major features of MMdetection are:

- **Modular design:** Because of it's design the detection framework can be easily changed and a flexible/customize version can be created as per the requirement, it canbe done by combining different kind of modules likebackbone, neck and RIO extractor.
- **Support of multiple frameworks:** The toolbox and it's simple architecture made it very easy to use additionally it provides a large variety of detection frameworks like Fast R-CNN, Faster R-CNN, Mask R-CNN, RetinaNet, DCN etc.
- **High efficiency:** All operations (masking, boundary box Creation, prediction) run on GPUs, hence the training speed is faster than or comparable to other code–bases in- cluding Detectron, mask rcnn-benchmark and SimpleDet. It also provides vides weight of more than 200 network model.

B. *Architecture*

Although the model architectures of different detectors are different, they have common components, which can be roughly summarized into the following classes:

- **Backbone:** Backbone is the part that transforms an image to feature maps, such as a ResNet-50[8] without the last fully connected layer.
- **Neck:** Neck is the part that connects the backbone and heads. It performs some refinements or re configurations on the raw feature maps produced by the backbone. An example is Feature Pyramid Network (FPN).
- **Dense Head (Anchor Head/ Anchor Free Head):** DenseHead is the part that operates on dense locations of feature maps, including Anchor Head and Anchor Free Head, e.g., RPNHead, RetinaHead, FCOSHead.
- **RoIExtractor:**RoIExtractor is the part that extracts fea- tures as per the region of interest from a single or multiple feature maps. An example that extracts RoI features from

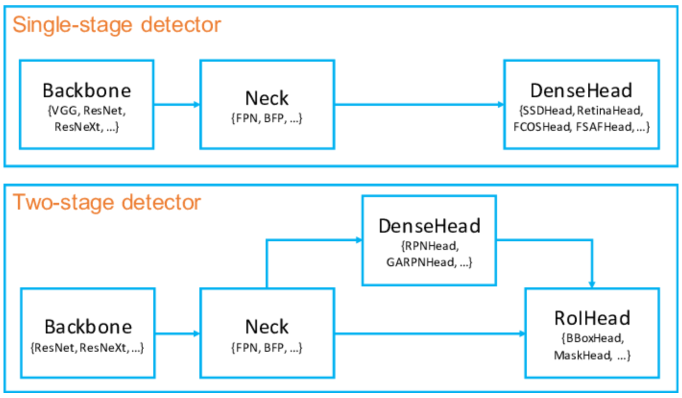

Fig. 1. Framework of single-stage and two stage detector.

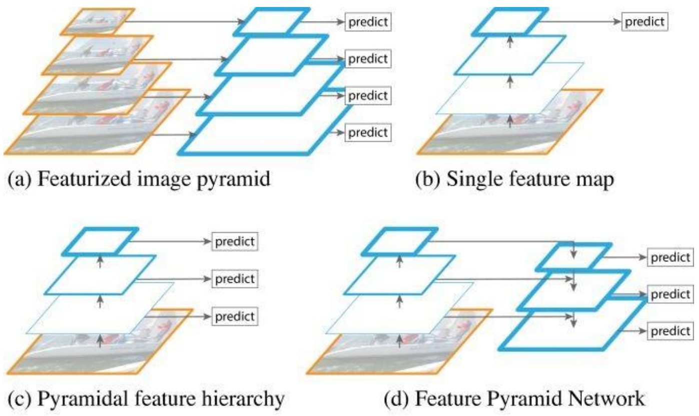
predict
(a) Featurized image pyramid
(b) Single feature map
(c) Pyramidal feature hierarchy
(d) Feature Pyramid Network

Fig. 2. Feature Pyramid Network.

The corresponding level of feature pyramids is single RoIExtractor.

- **RoIHead (BBoxHead/ MaskHead):** RoIHead is the part that takes RoI features as input and makes RoI-wise task specific predictions, such as bounding box classification/ regression, mask prediction.

C. *Resnet50 with FPN*

In this section, the architecture of the object detector ex- plained With Feature Pyramid Network (FPN). In the Fig. 2 the working of FPN is shown.

Identifying components of different scale and complexity is a difficult task which we overcome by using the same picture of the different size /scale, however this approach has some disadvantages, including high memory demand and high time consumption. [3].

For the above issue, FPN has a fantastic approach, it con- structs a feature pyramid and uses this for object recognition. The Feature Pyramid Network (FPN) is a feature extractor programmed with accuracy and speed in mind.

It substitutes the feature extractor of detectors such as Faster R-CNN and generates various feature map layers (multi-scale feature maps) with higher quality information as compare to the standard feature pyramid for object detection. A bottom- up and a top-down pathway are constructed of FPN as shown in Fig. 2. The image quality reduces as we go up. The semantic meaning for every layer increases with more high- level structures detected.

FPN extracts feature maps and later feeds into a detector, says RPN, for object detection. RPN applies a sliding windowover the feature maps to make predictions on the object (has anobject or not) and the object boundary box at each location[3]. In the FPN framework, for each scale level a 3 × 3 convolution filter is applied over the feature maps followed by separate 1 × 1 convolution for object predictions and boundarybox regression. These 3 × 3 and 1 × 1 convolutional layers arecalled the RPN head. The same head is applied to all different scale levels of feature maps.

D. *Formula to pick feature map*

The equation for determining the characteristic maps is dependent on the ROI’s width w and height h.

$$k = ko + log2 \sqrt{wh}/224) \qquad (1)$$

Where, ko = 4
k is the Pk layer in the FPN used to generate the feature patch. if the model has assigned k = 2, it made P2 as model’s feature maps and ROI pooling will be done and it will feed the result to the framework used like Fast R-CNN head (Fast R-CNN and Faster R-CNN have the same head) to finish the prediction.

E. *Comparison*

The comparison of different feature with and without FPN is mentioned in the table 1.
AR (Average recall): The ability to capture Object.
Inference time: Time taken for prediction.

Table 1:- Comparison of Feature with and Without FPN

Feature	***Without FPN***	***With FPN***
Training Time	Normal	Increased
Dataset requirement	Big	Small
Test/validation time	High	Low
Accuracy	Normal	Increased
AR	44.9	56.3
Inference time	0.32 sec.	0.148 sec.

III. MODEL

A. *Model composition*

In this section, a detail description of the model used for mask detection is explained. The model described part by part in every section it’s corresponding architecture was explained in dictionary format for ease of understanding.

- **Backbone:** The backbone of this model was created usingResnet50[8] with batch normalization it is used for the feature extraction from the image, this can easily be replaced by some other feature extracting network.
- **Neck:** The neck used in this model is FPN and the full description is below:

Neck = dict (type = ‘FPN’,
In channels = [256, 512, 1024, 2048],
Out channels = 256,Num outs = 5)

- **Dense head:** For the dense head architecture had used RPNHead it extract feature from the dense part of the image:

Rpn head = dict (type = 'RPNHead', In channels = 256, Feat channels = 256)

- **ROI head:**ROI head is a very interesting part of the model in this model CascadeROIHead was used, the main work of RIO head is to propose the region of interest from where the model should detect the object. The description is below:

Roi head = dict (type = CascadeRoIHead', num$_s$*tages* = 3)

Table 2 Model Comparison Detectron2 And Mmdetection

Module	***MMdetection***	***Detectron2***
Base Model	Resnet50	Resnet50
Neck(FPN)	Yes	Yes
Training time	more	less
RIO	Yes	Yes
Learning rate	0.02	0.02
mAP	0.139	0.158

- **Optimizer:** The Stochastic gradient descent (SGD) optimizer was used in this model. SGD is an iterative method for max- imizing an objective function with adequate and appropriate (e.g. differentiable or subdifferentiable) smoothness properties. It can be described as a stochastic gradient descent optimiza- tion approximation, since it replaces the actual gradient (cal- culated out of the whole data set) with a gradient optimization approximation (calculated from a randomly selected subset of data). In high-dimensional optimization problems, this approach is very usefull as it decrease the computational stress and achive faster iteration in exchange of low convergence rate.
- **Process of Training:** In this section we will see the flow of data during training cycle. We can see all the main operation and transformation performed on the data.

train pipeline = [
dict(type = 'LoadImageFromFile'),
dict(type = 'LoadAnnotations', with$_b$*box* = *True*),
dict(*type* = '*Resize*′, *img*$_s$*cale* = (1333, 800), *keep*$_r$*atio* = *True*,
dict(*type* = '*RandomFlip*′, *flip*$_r$*atio* = 0.5), *dict*(*type* = '*Normalize*′, ∗∗ *img*$_n$*orm*$_c$*fg*),
dict(*type* = '*Pad*′, *size*$_d$*ivisor* = 32), *dict*(*type* = '*DefaultFormatBundle*′), *dict*(*type* = '*Collect*′, *keys* = ['*img*′,′*gt*$_b$*boxes*′,′*gt*$_l$*abels*′])
]

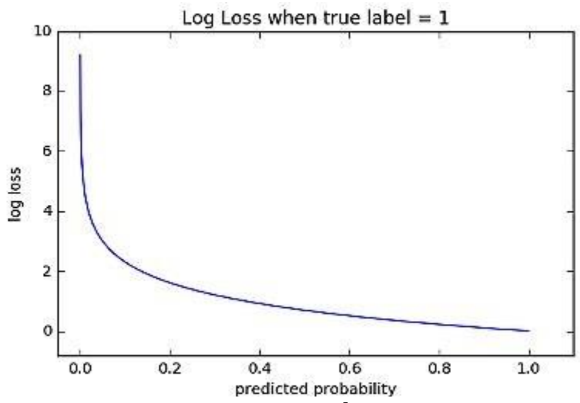

Fig. 3. Log loss

- **Process of Testing:** Below is all the operation performed onthe data during testing.

test pipeline = [
dict(type = 'LoadImageFromFile'), dict(type = 'MultiScaleFlipAug, img$_s$*cale* = (1333, 800), *flip* = *False*, *transform* = [
dict(*type* = '*Resize*′, *keep*$_r$*atio* = *True*), *dict*(*type* = '*RandomFlip*′),
dict(*type* = '*Normalize*′∗∗ *img*$_n$*orm*$_c$*fg*), *dict*(*type* = '*Pad*′, *size*$_d$*ivisor* = 32),
dict(*type* = '*ImageToTensor*′, *keys* = ['*img*′]), *dict*(*type* = '*Collect*′, *keys* = ['*img*′])]
]

B. Model loss during training

To fine tuning our model, we used Cross Entropy Loss. Cross-entropy loss, or log loss, measures the efficiency ofa model of classification whose output is a zero to one probability value. As the expected probability diverges from the real mark, cross-entropy loss increases. But it will bewrong to estimate a chance of .015 where the real observation label is 1 and result in a high loss value. A great model will have a 0[5] log loss Fig.3.

The graph in Fig.3 shows the range of possible loss values given a true observation (Masked = 1). As the expected likelihood reaches 1, log loss decreases steadily. However,the log loss increases significantly as the expected likelihood decreases. Log loss penalizes both types of errors, however especially those predictions that are confident and wrong!

Cross-entropy and log loss are slightly different depending on context, however in prediction when calculating error rates/probability between 0 and 1 they resolve to the same thing.

Fig. 4 showing epochs versus loss value graph for our model,

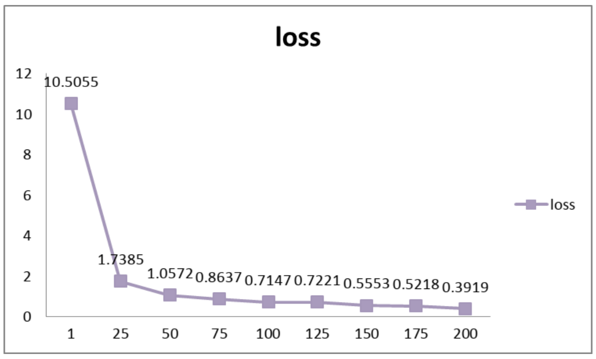

Fig. 4. Loss during training

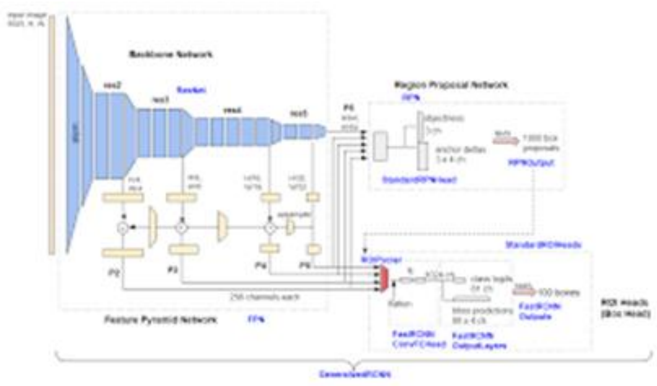

Fig. 5. Detectron2 Archetecture

We trained our model for 200 epochs. The loss value was takenon every 25 epochs to draw.

IV. DETECTRON2

A. *What is Detectron2*

Detectron2 is Facebook AI research's next generation soft- ware system, it is a ground-up reconstructed on the previ- ous version of Detectron and it originated from maskrcnn- benchmark [11].

Major feature of Detectron2 are:

- It is based on PyTorch.
- It has more feature like panoptic segmentation, Densepose, Cascade R-CNN, rotated bounding boxes, PointRend, DeepLab, etc.
- It can be easily integrated with different project's becauseit's usability as a library.
- Less training time.
- Models can be exported to TorchScript format or Caffe2 format for deployment.

The main component of Detectron2 is shown in figure 5.

Both model Detectron2 and MMdetection share some common module like FPN, RPN, ROI pool and backend network (Neck), all these are explained in MMdetection section.

We used ResNet50 for feature extraction with a learning rate of 0.02

V. DATASET

To train the model it need's annotated images so the model can extract features from the images and distinguish between masked and unmasked faces. We used total 8982 annotated images to train and validate the model, we used LabelImg software to annotate all images (figure 6). It creates an individual json file for each image file according to the annotation, these json files contain the information like file/image name, category(one or more than one), bounded region information in image, height of the bounded region, width of the bounded region and total area enclosed within labelled region.

All the individual json files converted into one single COCO file using the labelme2coco converter package in python. We need the images to be in Common Objectsin Context (COCO) formats, it stores the annotation details for the bounding box in JSON format. The main component of the coco file are:

– **Info:**Contains high-level information about the data set.
– **Licenses:** Contains a list of image licenses that apply to images in the data set.
– **Categories:** Contains a list of categories. Categories can belong to a super category.
– **Images:** Contains all the image information in the data set without bounding box or segmentation information. image id's need to be unique.
– **Annotations:** List of every individual object annotation from every image in the data set.

Example of a coco format(file and image) is below."info": info,
"licenses": [licenses], "categories": [categories], "images": [images], "annotations": [annotations]

VI. RESULT

The final output is displayed in figure 7, the model is successfully able to predict and discriminate between Masked and unmasked faces. Mask face is further classified as Correct(Mask) and Incorrect.

As per the mAP vale in table III, we used MMdetectionfor our facemask detection model.

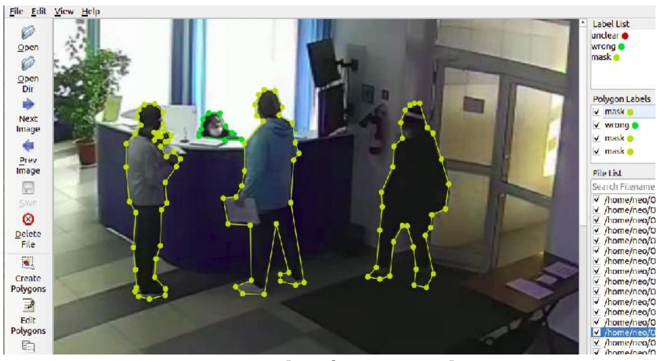

Fig. 6. Example of Annotated Image

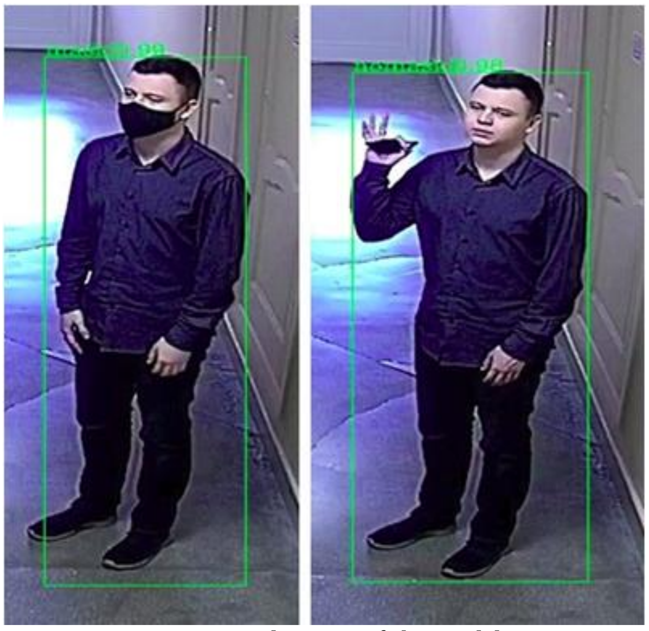

Fig. 7. Final output of the model.

To measure the accuracy of our model we used mAP (mean Average Precision), it is a popular metric to measure object detector like faster R-CNN, SSD, etc. It gives a calculated value between 0 and 1.

Calculating accuracy for an object detector is a little complicated as we have to detect the object or class and also the area where the object was detected.

– **Precision:** It calculates the Accuracy of the predictton.
– **Recall:** It measures how good model find all the positives.

$$Precision = TP/(TP + FP) \quad (2)$$
$$Recall = TP/(TP + FN) \quad (3)$$

Where, TP = True Positive TN = True Negative
FP = False Positive FN = False Negative

Another important term we have to understand is IoU (Intersection over Union) [6], To check the correctness of our model's we first have to judge the correctness of each of these detection's. The metric that tells us the correctness of a given bounding box is the IoU.

A visual representation of IoU is shown in Fig.8. subcap-tion

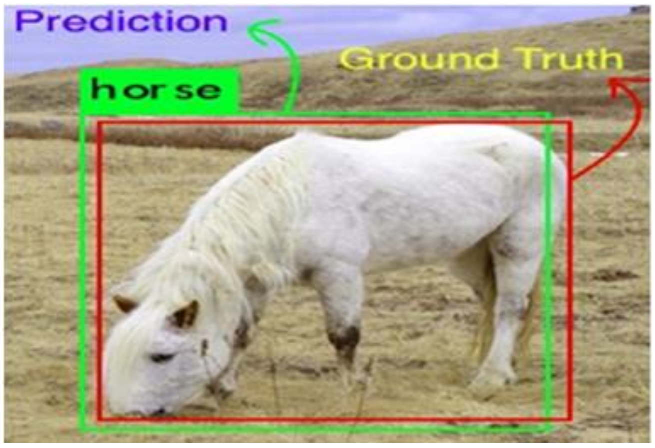

Fig. 8. Visual representation of IoU.

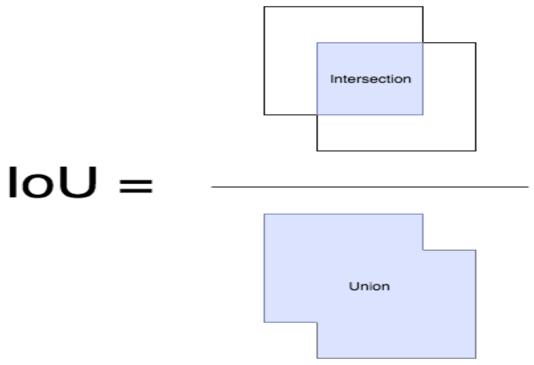

Fig. 9. Calculation of IOU

To get TP and FP we use IoU, we now have to identify if the detection (a Positive) is correct (True) or not (False). The most commonly used threshold is 0.5 - i.e. if the IoU is ¿ 0.5, it is considered a True Positive, else it is considered a false positive.

Table 3:- Comparison Of Different Map Between Mmdetection And Detectron2

Metric	**IoU**	**MMdetection**	**Detectron2**
mAP	@[IoU= 0.50:0.95]	**0.158**	0.139
mAP	@[IoU= 0.50]	0.238	0.277
mAP	@[IoU= 0.75]	0.181	0.109

VII. FUTURE WORK

We are trying to create a pipeline of 2 model which should be able to do facemask detection and person re-identification. For that purpose we are using the facemask

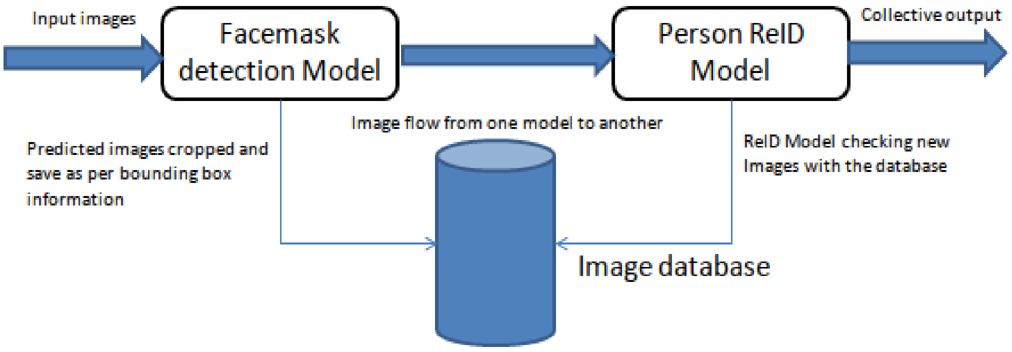
Input images
Facemask detection Model
Person ReID Model
Collective output
Image flow from one model to another
Predicted images cropped and save as per bounding box information
ReID Model checking new Images with the database
Image database

Fig. 10. Final pipeline of the model

Detection models output and using the boundary box information to create a database of all person who enteredthe premises. Then we used torchreid a python library and pre-trained person re-identification model available on our collected data.

Initially our gallery size is 21127 images and we used top- 10 search result to calculate accuracy. if we choose 10 images randomly we get 2% accuracy but with torchreid we are able to get 67% accuracy. Currently we are working to improve our person re-identification model and complete pipeline.

REFERENCES

[1]. "Naming the Coronavirus Disease (COVID-19) and the Virus That Causes It." World Health Organization, World Health Organization, www.who.int/emergencies/diseases/novel-coronavirus-2019/technical -guidance/naming-the-coronavirus- disease-(covid-2019)-and-the-virus -that-causes-it (accessed October 19, 2020).
[2]. K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, Z. Zhang, (2019). MMDetection: Open MMLab Detection Toolbox and Benchmark. arXiv preprint arXiv:1906.07155.
[3]. Lin, T.-Yi, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117-2125. 2017.
[4]. Wikipedia contributors, "Stochastic gradient descent," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org /w/index.Php ?ti- tle =Stochastic-gradient-descentoldid=983180780 (accessed Octo-ber 19, 2020).
[5]. Wikipedia contributors, "Cross entropy," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org /w/index.php?title= Cross- entropy oldid=983515385 (accessed October 19, 2020).
[6]. T. Shah, "Measuring Object Detection Models-MAP-What Is Mean Average Precision?" Tarang Shah - Blog, 26 Jan. 2018 Tarangshah .com /blog /2018-01-27/ what-is -map-understanding- the-statistic-of-choice-for-comparing-object-detection-models
[7]. J. Hui., "MAP (Mean Average Precision) for Object Detection." Medium, Medium, 3 Apr. 2019, medium.com/@jonathan- hui/map-mean-average-precision-for-object-detection- 45c121a31173 (accessed October 19, 2020).
[8]. K. He, X. Zhang, S. Ren. J. Sun (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[9]. Wikipedia contributors, "COVID-19 pandemic by country and territory," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=COVID-19-pandemic- by-country-and-territoryoldid=984313691 (accessed October 20, 2020).
[10]. Wikipedia contributors, "COVID-19vac- cine," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=COVID-19- vaccineoldid=984344201, (accessed October 20, 2020).
[11]. Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detec- tron2," https://github.com/facebookresearch/detectron2, 2019.
[12]. Zhou, K. and Tao Xiang. "Torchreid: A Library for Deep Learn- ing Person Re-Identification in Pytorch." ArXiv abs/1910.10093 (2019): n. pag.