Neural Networks and Neurocontrollers course

Project report

Face mask detection and classification

Emilia Szymańska

Wrocław
University
of Science
and Technology

15th May 2021

Contents

1 Project objective

The objective of this project was to implement a neural network that detects human faces and classifies them into three categories - with mask, without a mask and with a mask worn incorrectly. I used YOLOv5, but as it turned out to be simple to use, I decided to extend the project with a convolutional neural network that classifies images into two classes (with mask and without a mask). In both cases, I have used PyTorch framework in Jupyter Notebooks in Google Colaboratory - the notebooks are available on repository `https://github.com/emilia-szymanska/mask_detection_and_classification`.

2 Convolutional Neural Network for face mask classification

The dataset (`https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset`) contains around 11 800 RGB human face images (distributed into test set, vaildation set and train set) that had to be scaled (and cropped if needed) to 64x64 dimension. Samples from the dataset are presented in fig. 1.

Figure 1: Face mask classification dasaset samples.

The convolutional neural network I implemented takes RGB images of the dimension 64x64 as inputs and the output of the network are two values (summing up to 1) which reflect the probability of belonging to the corresponding classes - with mask and without mask. The network has the following architecture:

- convolution layer with kernel size 3x3 (3x64x64 -> 32x62x62),
- ReLU activation function,
- max pooling with kernel 2x2 (32x62x62 -> 32x31x31),
- dropout with probability of an element to be zeroed of p = 0.3,
- convolution layer with kernel size 4x4 (32x31x31 -> 64x28x28),
- ReLU activation function,
- max pooling with kernel 2x2 (64x28x28 -> 64x14x14),
- dropout with probability of an element to be zeroed of p = 0.4,
- flattening the network,
- fully connected layer with 256 neurons,
- ReLU activation function,
- dropout with probability of an element to be zeroed of p = 0.5,
- fully connected layer with 2 neurons,
- application of log(Softmax(x)) function.

With a batch size of 128, 30 epochs and a learning rate equal to 0.01, the network achieved an accuracy of 98% on the training phase. There were only 33 incorrectly classified images in the test set containing nearly 1 000 images, which gives 97% of accuracy.

Figure 3 presents the losses during the training phase.

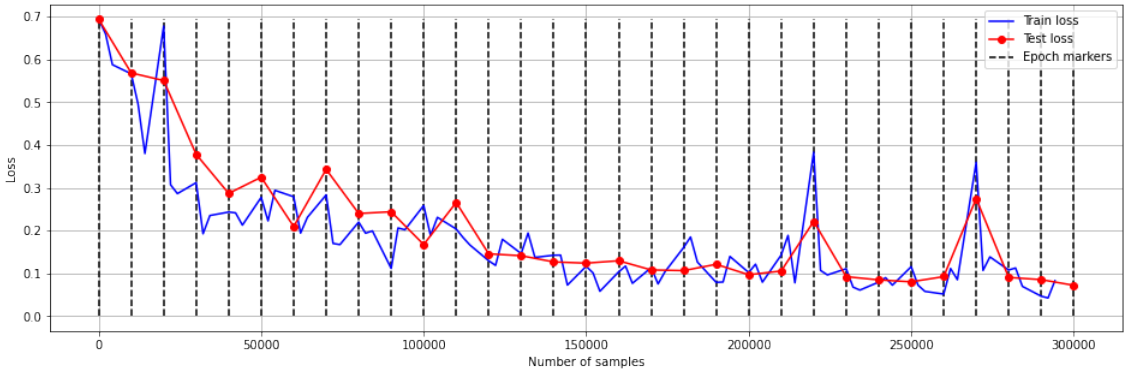

Figure 2: CNN training results.

3 YOLOv5 for face detection and mask classification

YOLO [1] [2]- which stands for You Only Look Once - is a network which sees the entire image during training once, it requires only one forward propagation pass through the neural network to make predictions [3]. It is fast and achieves better accuracy for object detection, therefore it was used in this project. My goal was to detect and classify human faces into those having masks, not wearing them or wearing them incorrectly.

My first choice was the dataset `https://public.roboflow.com/object-detection/mask-wearing/`, which can return data in darknet (YOLO) format. Unfortunately, the dataset contains about 150 samples, which is not well distributed regarding mask types and (with mask)/(without mask) ratio. The network learnt that only surgical mask should be taken into consideration. What is more, it tended to classify skin-coloured areas next to bright areas as detected masks, although it was not even a human face.

Figure 3: Overtrained network result.

Therefore, I chose a different dataset with better distributed classes and with over 850 samples - `https://www.kaggle.com/andrewmvd/face-mask-detection`. It was necessary to perform labels conversion as they are originally in COCO format - they needed to be 'translated' into darknet type. With 50 epochs and a batch size of 16, the network training process looked as presented in fig. 4. There are plots presenting loss function for bounding box detection, general object detection, classification (for both training and validation), precision, recall (how good it finds all the positives) and mean average precision (for recall value equal to 0.5 and 0.5:0.95).

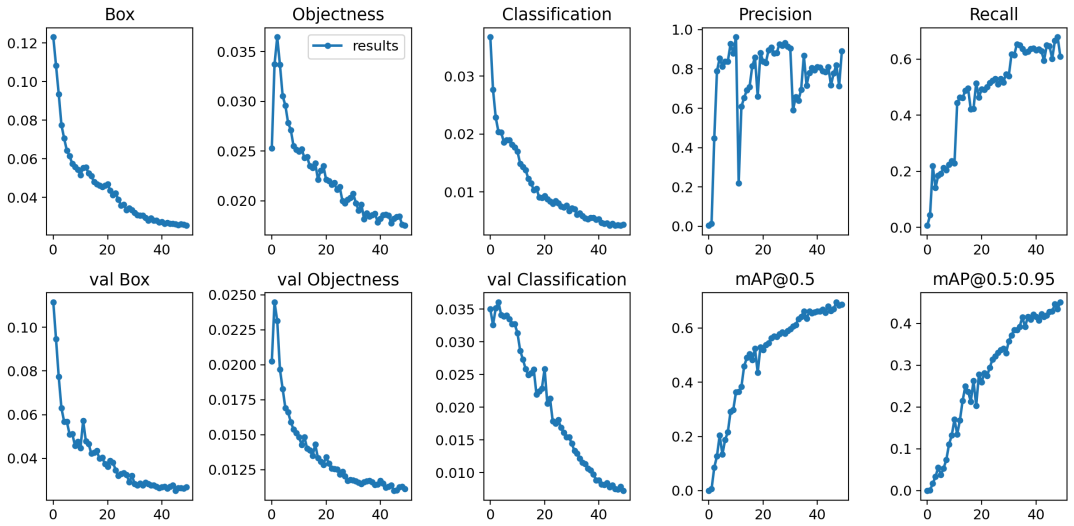

Figure 4: YOLO plotted training results.

Figure 5 shows the results of putting test samples as input for the trained network. It can be seen that this NN detects different kinds of masks (not only surgical as in previous case) and can state if a mask is worn incorrectly. It is also possible to run a script with image stream from web camera - the result of such a use is shown in fig. 6.

Figure 5: YOLOv5 results on test samples.

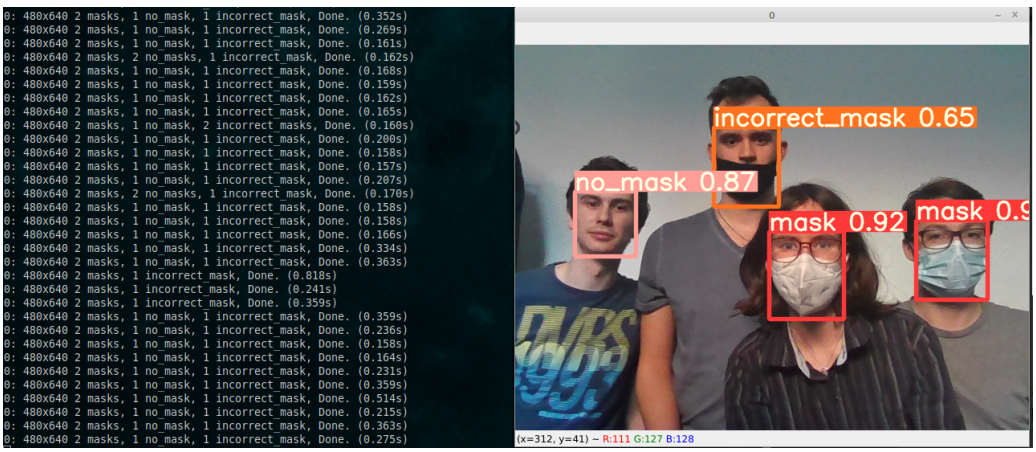

Figure 6: YOLOv5 object detection on web camera stream.

4 Conclusions

- YOLOv5 is very easy to use - you need to take care of labels format if needed and to make sure the data is well distributed among classes.
- Classification with a CNN is as well not hard to achieve, but it demands more effort from a programmer than YOLO - implementing your test, train or validation functions takes more time than running a few lines of code as in YOLO.
- My CNN cannot detect bounding boxes with human faces - to have such an effect, additional neurons for bounding box parameters would need to be added in the output layer. It is also be more complicated if there is not a fixed number of people in the image to detect.

References

[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[2] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[3] "Overview of the YOLO object detection algorithm," `https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0`.