Aide à la décision

M2 VMI - TP AD: Object detection using CNNs - Facemasks

Université de Paris

Amine MARZOUKI

17/10/2021

M2 VMI, FA

Clarifications

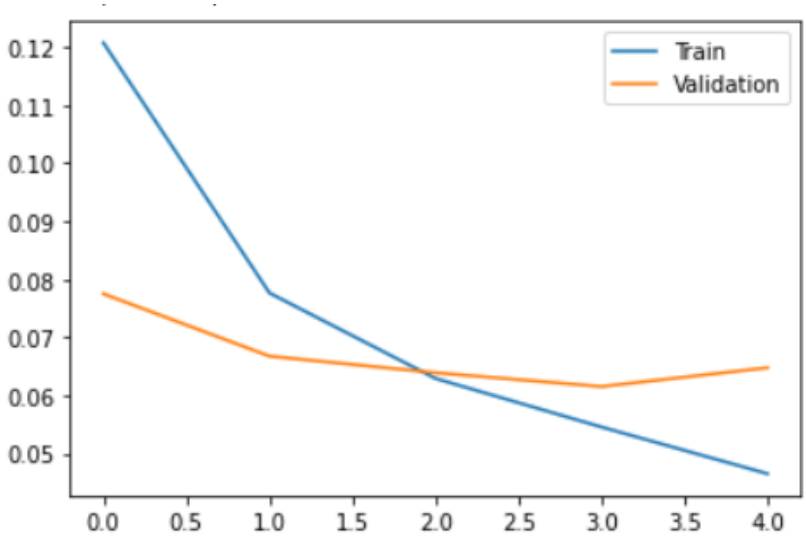
- The questions were answered directly in the notebook
- There are two notebooks, one that was downloaded from colab in which we did the training (provided only for the trace of the training and as a proof of work), the other one contains the answers and the evaluation of the model after downloading the weights and loading it locally.

Training

The training, validation and testing set percentages were fixed at 0.7, 0.15, and 0.15 respectively in all of our xps.

We tried to optimize a little bit the code to exploit a local GPU (RTX 3060 6GBs) since it was faster than K80 *(3 mins vs 10 mins in the cloud per epoch)* provided on google colab and also since we run out of limit that was provided by the cloud service. The problem is that given the small memory the RTX had, we couldn't do even a batch size of 2 since the accumulation of the losses cause a run out of memory exception in later epochs, hence we tried many hyperparams in the google colab platform, and we adjusted the batch size to 4 instead of 2 to exploit the full memory of the GPU and to have a better estimate of the gradient.

We noticed that even with very small LR, we started overfitting so we did only 5 epochs in training and that was also because it took ~2h, and so we had this plot for the losses.



So given that the validation loss started to increase, we did an early stop. We provide two notebooks, the first one of the cloud as a proof that we did the training, the second one where we provide the answers to the questions and where we loaded the weights of the trained model after we downloaded them in addition to the evaluation as we couldn't do it in colab since the session crashed so we lost all the objects in memory and we reached the limit of using the GPU.

Evaluation

First measure we used to evaluate our model is the Intersection over Union. Nevertheless, we run into a problem which is when the model outputs more bounding boxes than there is in the ground truth. For example, an image may have only 1 person, so we must have only one box, but our model may output more than 1 box, so we somehow need to deal with how we are going to take this issue into account.

In order to make the predictions of the model a little bit better and lower the amount of boxes in the output, we added a post processing step that consists of Non Maximum Suppression that made our problem a little bit easier in the sense that we have less boxes to deal with, but it didn't eliminate it. The threshold of the NMS was cherry picked.

Now, essentially we measured the IoU in the following way:

- Given an image and its corresponding ground of truth N bounding boxes
    - Calculate the IoU between all the combinations of the N ground of truth boxes and M predicted boxes, which will result in a matrix of shape (N,M)
    - Take the maximum value for each line, that is equivalent to focus on only the predicted boxes that have some overlap with the ground truth boxes.
    - Calculate the mean of these IoUs
- Do the first step for all the images and then calculate the mean.

We got a mean IoU value over all the test set of 86,68%.

While this gives us an idea about the performance of our model, it isn't enough, because it doesn't capture whether the model gave the right label for the boxes or not, which we will cover in the next section.

Precision

Now we are going to calculate the precision taking into account the predicted labels and calculate our confusion matrix. But again we have a problem, as you may notice below.

Here is an example of prediction done on the last image in the test set.

```
In [299]: pred

Out[299]: [{'boxes': tensor([[244.6213, 131.6936, 364.6190, 246.4414],
                             [ 35.7443, 101.2780, 163.5118, 237.9713]], device='cuda:0'),
           'labels': tensor([1, 1], device='cuda:0'),
           'scores': tensor([0.9973, 0.9955], device='cuda:0')}]
```

```
In [300]: labels

Out[300]: [{'boxes': tensor([[ 35., 106., 162., 245.],
                             [248., 135., 361., 254.]], device='cuda:0'),
           'labels': tensor([1, 1], device='cuda:0'),
           'image_id': tensor([287], device='cuda:0')}]
```

We can see that actually our model predicted correctly both labels, and we have very good bounding boxes, but the issue is they didn't come out in the same order as we have in the ground truth.

So our prediction is right, the bounding boxes that were predicted are very good, but we just can't rely on the order that our model gives us, so for example if we had instead [1,2] our model would have said [2,1] but the bounding boxes are also swapped, so at the end our model was not wrong, it is simply an output of a different order.

To fix this problem, we are going to fix the order of the output predicted labels based on the overlap between the boxes, so for example in this case we have a very big IoU between box1 and predicted box0, and box0 with the predicted box1, so we will rearrange the labels based on these indices.

This is done by simply calling the max method in torch which also returns the indices of the maximum values, then we use these indices to rearrange the list.

After doing these adjustments, we calculated the confusion matrix using sklearn.

```
target_names = ['with_mask', 'without_mask', 'mask_weared_incorrect']
print(classification_report(all_labels, all_predicted_labels, target_names=target_names))#
```

```
                      precision    recall  f1-score   support

           with_mask       0.95      1.00      0.97       475
        without_mask       0.97      0.93      0.95       179
mask_weared_incorrect       0.00      0.00      0.00        16

            accuracy                           0.96       670
           macro avg       0.64      0.64      0.64       670
        weighted avg       0.93      0.96      0.94       670
```

CONCLUSION

We actually now have some good results, our model is performing well on all but the 3rd class (mask_weared_incorrect).

Our hypothesis is that it is due to the very few examples of these cases. Otherwise we have a really good precision in both classes, and we got a very good IoU value over all the test set using only 70% of data for training with no data augmentation.