# Ear Detector Development

Assignment #2

Image Based Biometrics 2020/21, Faculty of Computer and Information Science, University of Ljubljana

Žiga Trojer

## I. Introduction

In this seminar paper, I tried to implement an ear detector. I did this with two different methods, which I will present below. The first method is the SSD method - single shot multibox detector. A simple architecture was used, which allowed me to train on a laptop graphics card (GTX 1650 Ti). I put together several models, which I then compared with each other. Out of interest in better results, I also tried to implement ear detection on more demanding architecture. I use the RetinaNet object detection. I also trained this model on my graphic and got pretty good results. For training, I used 5 epochs with 250 steps per epoch. Each training took me around 12 minutes to train. Code is published on my Github repository: https://github.com/trojerz/keras-eardetection and https://github.com/trojerz/keras-retinanet

## II. Methodology

Authors of the Single Shot MultiBox Detector states: "Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network" [1]. SSD only needs an input image and ground truth boxes for each object for training.

## III. Results

I first used the original version of the detector. I adjusted the detector to my data (of course I had to process the images and prepare annotations before). The detector uses the Adam optimizer and uses the SSD7 network architecture. It is a simple 7-layer version architecture, adapted for easier training. Let's look at how the training and validation loss evolved to check whether our training was going to the right direction - Figure 1.

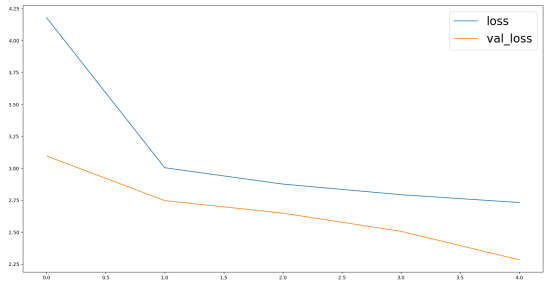


Figure 1: Loss function for classification - original model

The validation loss (val_loss) has been decreasing at similar pace as the training loss (loss). This indicates that the model has been learning effectively over the 5 epochs. The graph also indicates that the model is not over-fitted, because both losses are still decreasing. We could train a bit longer to get better results, but I think results are still pretty good for analysis.

In my second attempt, I used 3 different optimizers: Adadelta, SGD and NAdam. All the other parameters stayed the same. Also, parameters for optimizers were not changed. Adadelta optimizer yielded slightly better validation loss at the end, but model may be a little over-fitted, because validation loss did not decrease with the loss from the second to the fourth epoch - Figure 2 - first graph. SGD optimizer was a bit worse than Adam and Adadelta. The final validation loss and loss was 3 at the end, there was even an increase in loss at 4th epoch - Figure 2 - second graph. NAdam optimizer yielded similar results to Adam, the decreasing of losses was even more constant than Adam's - Figure 2 - third graph.

In my third attempt, I changed the network architecture - for each layer, I added another Convolution on top - I added a bit of complexity to a model. I tried two different optimizers - the ones that promised the best results: Adam and NAdam. The model with Adam optimizer seemed a bit better from the minumum loss perspective, but NAdam was better in 'not-overfitted' aspect - Figure 2 - fourth and fifth graph.

Because I still could not decide, which model is the best, I implemented some functions that returns the IoU metric - Intersection over Union metric. It tells us how good the predictions were. My threshold for prediction was 30 % - if a prediction was over that threshold, we used it, if not, we counted it as there was no prediction. The table below shows next metrics: Average IoU, Maximum IoU and number of cases when there was no prediction (out of 250 cases).

| Model name | Avg. IoU | Max. IoU | # no predictions |
|---|---|---|---|
| Adam original | 32 % | 87.92 % | 58 |
| Adadelta | 34 % | 92.47 % | 50 |
| SGD | 25 % | 84.42 % | 76 |
| Nadam | 36 % | 90.87 % | 80 |
| Nadam "complex" | 37 % | 91.44 % | 76 |
| Adam "complex" | 36 % | 83.33 % | 148 |
| RetinaNet | 73 % | 98.63 % | - |

We need to take into account that those metrics are lower than they should be. The problem was with pictures with more than 1 ground truth box - the IoU for those was near zero, because we let only one prediction per picture due to complexity reasons.

From all given metrics, we could conclude that Nadam "complex" (model with Nadam optimizer and with added complexity to a model) is the one that has the best over-all performance (from the SSD type models).

Now it is time to introduce another model, which really impressed me. This model is called RetinaNet object detection. Authors of this models states: "To evaluate the effectiveness of our loss, we design and train a simple dense detector we call RetinaNet. Our results show that when trained with the focal loss, RetinaNet is able to match the speed of previous one-stage detectors while surpassing the accuracy of all existing state-of-the-art two-stage detectors" [2]. I trained and try the model on AWEForSegmentation dataset myself. I already included the results in the table, where model name is RetinaNet. You can see that those values are a lot better than for other metrics. Metrics for this model should be higher as well, but I adapted them so the results are comparable to each other. For this model, I calculated additional metrics - table below. We can observe, how good RetinaNet object detection is. Unfortunately, I did not calculate these metrics for other models.

| Model name | Avg. precision | Inference time | mAP |
|---|---|---|---|
| RetinaNet | 92.85 % | 0.30 | 92.85 % |

## IV. Conclusion

I have tried quite a few different approaches on how to detect an ear in an image. Some approaches performed better than others. Among SSD models, the NAdam 'complex' model, which has the highest average IoU, proved to be the best in terms of IoU metrics and loss graphs. In my opinion, the best prediction of my ear was from NAdam model. RetinaNet model is on a different level. It has much better results in terms of metrics and in detection of a ear in a picture (probability of 95.8 % that there is my ear is really impressive). RetinaNet is also impressive in terms of speed - I needed the same amount of training for RetinaNet than for SSD models. There is still a lot of place for improvement in those models. With the experimenting with optimizers and network architecture, I gained a lot of insights how those models work and what is important.

## References

[1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, "SSD: Single Shot MultiBox detector," *ECCV*, 2016.

[2] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detector," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

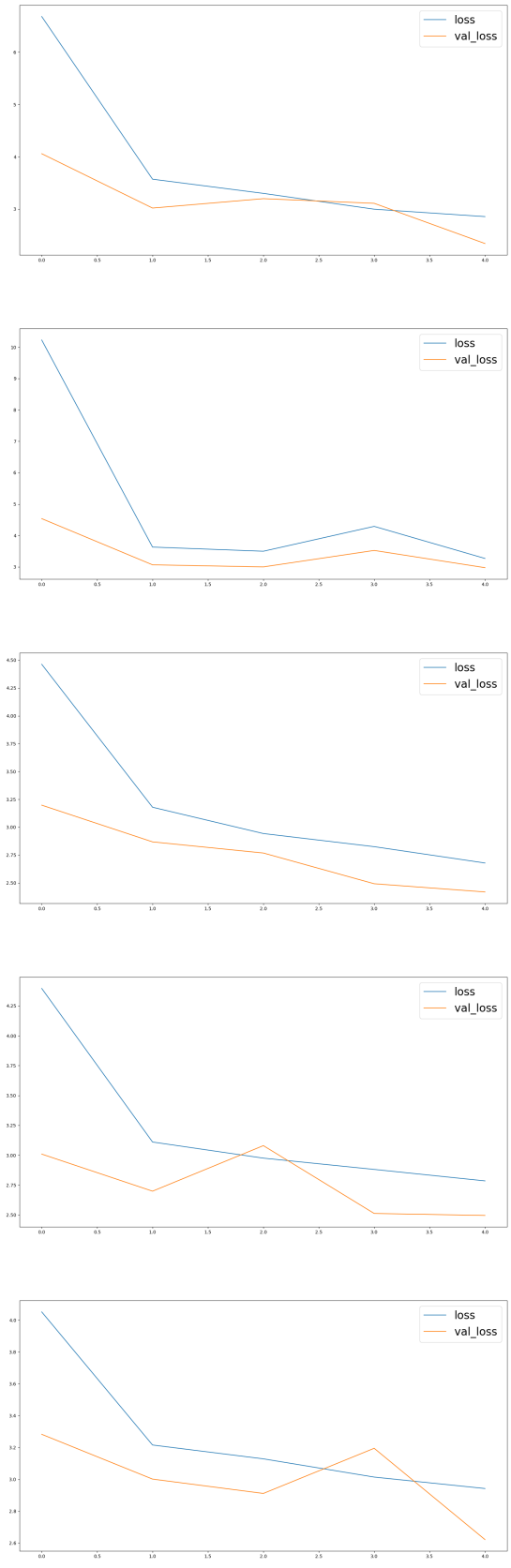


Figure 2: Loss function for classification - Adadelta, SGD, Nadam optimizers and loss function for classification - NAdam and Adam optimizer with changed network architecture.

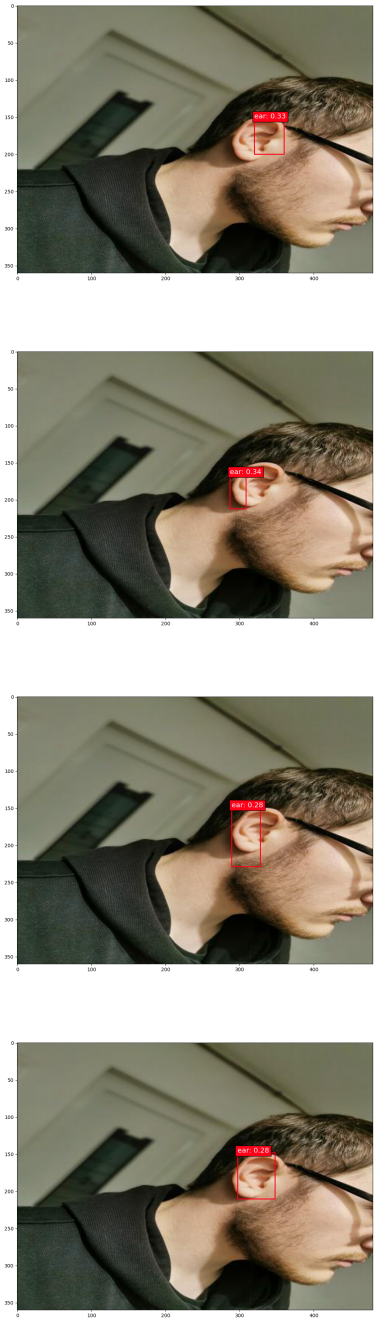


Figure 3: Detection with models with Adam, Adadelta, SGD and NAdam optimizer.

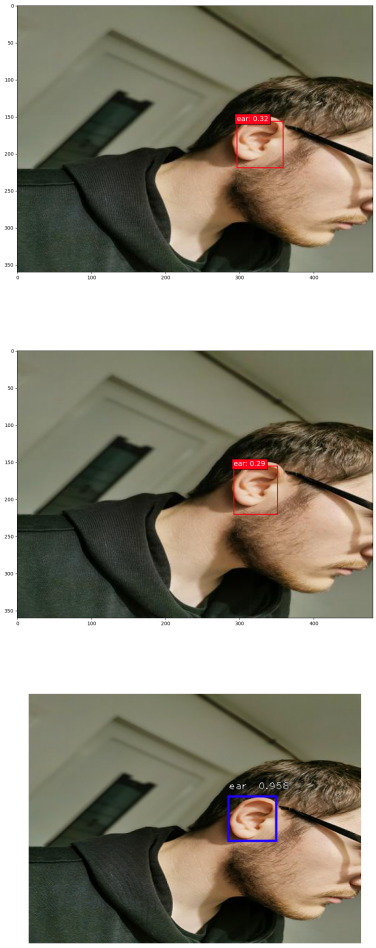


Figure 4: Detection with models with NAdam and Adam optimizer and changed network architecture and with RetinaNet model.