Machine Learning and Recognition 2022
Project – Single Person Detection

Martin Osvald, Jan Láncoš

xosval03, xlanco00

1 Summary

As part of this project we have developed three separate classifiers, one for audio recordings, another for photo evaluation with the last one being the combination of both. The all results for evaluations are in the folder *results_eval* where should be 4.txt files: "audio_gmm.txt","mixed_less_peak.txt", "mixed_more_certain.txt","nn_classification.txt".

The audio classifier uses two Gaussian Mixture Models to model the targeted and all the non-targeted classes respectively, and uses cepstral coefficients and their variance over time to characterize the individual voices.

Photo classifier is a basic convolutional network with four convolutional layers and an output of two probability values indicating if the person is the target or not. The dataset intended for training was fairly small, so we used a special library called Albumentations[1] for data augmentation [2].

2 Scripts usage

2.1 Libraries

Before running all the scripts, please download and install the Anaconda library[1]. All libraries should be in the environment.yml file. You can create the environment by simply typing:

```
make create_environment
```

You can then activate the environment called SUR_2022.

2.2 Audio scripts' usage

The audio script consists of two parts and can be run using the following syntax:

```
python3 audio_GMM_eval.py [eval_directory]

python3 audio_GMM_train.py [target_train_directory] [non_target_train_directory]
```

Scripts use the `wav16khz2mfcc()`, `train_gmm()` and `logpdf_gmm()` functions from the `ikrlib.py` library, they therefore need it to run properly.

The evaluation script's output is written as ASCII to file `audio_GMM.out`. It uses the pretrained GMM model. Its parameters are saved in the `audio_GMM_model` directory. The training script creates new parameters. To use these parameters for evaluation, simply move the generated files into the `audio_GMM_model` directory and replace the previous ones.

[1] https://www.anaconda.com/

2.3 Photo scripts' usage

For the image classifier training it is necessary to download the dataset with:

```
make download_dataset_CNN
```

The dataset is the same as the one given in this project, but the structure of directories has been changes a bit. The usage of scripts is then very simple:

```
make train_nn
```

This will train the neural network with the transformations applied.

```
make make_eval
```

This will create the results of the final evaluation as long as the downloaded evaluation data are in the root of the structure

```
make make_graphs
```

This will create graphs during the training using pickle files.

3 Audio classifier

The idea from the start was to use two Gaussian Mixture Models over the cepstral coefficients or both the target and non-target recordings. All the coeficients analyzed together without continuity don't say much about speech patterns or what was said, but they should tell enough about voice colour, and therefore enough to identify a person. The first issue was however determining the ideal number of clusters for both models to work with.

3.0.1 GMM parameters

By employing the method trial and error we have reached the ideal number of 3 clusters for the target class. Lower count led to misses, higher to false positives.

A slightly more complicated process yielded the final number 50 for the other classes combined; the experiments have generally shown though that pretty much any number over 30 yields a success rate over 93%. We settled on 50, mainly to keep the runtime of the classifier reasonable.

As for the number of learning iterations, the rule generally seemed to be "the more, the better". At a certain point though it was just dragging us down with only a little gain, so 100 iterations was decided to be an optimal middle path.

The model was already performing very well at this point, often with a 100% success rate. When it failed, though, it was a false positive. The next step was therefore trying to minimize the maximum log-likelihood yielded from the non-target evaluation to get the classifier to decide confidently by preparing the data.

3.0.2 Data preparation

Since all the recordings contained the initial "pop", we first got rid of that by removing the first 190 ms from all the recordings.

All the recordings then got "normalized" – functions of the respective coeficients over time had the mean of their values substracted from them. All the functions therefore settled around the zero mark – with the mean of exactly zero.

The recordings still contained a lot of silent parts, whether between sentences, or at the end of the record. Recordings were for that reason segmented, and the mean energy of these segments,

counted as the mean or their first (zeroth) coefficient, was then compared to the mean energy of the whole recording (now effectively zero). If it was lower, it was classified as silence and cut out.

These changes together pretty much guaranteed the model would perform with a 100% success rate over 30 iterations only. However, while the gap between the classes has grown as expected, it has only done so a little bit.

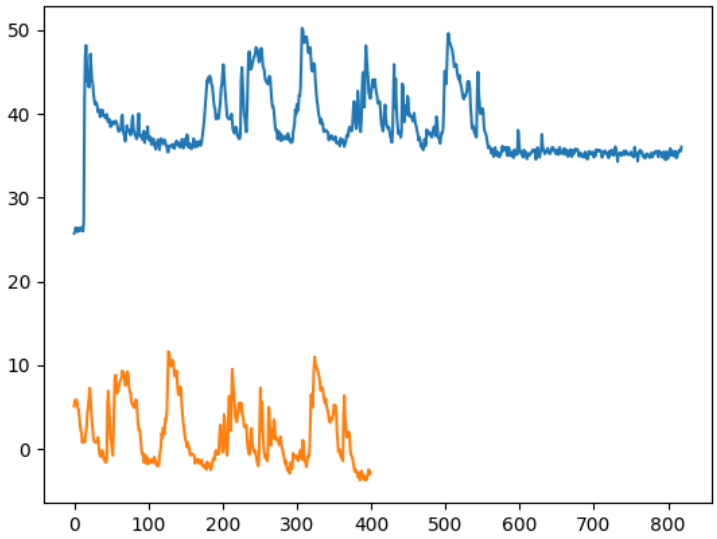

Figure 1: The preparation of the first coefficient's function

3.1 Model review

When trained on the `target_train` and `non_target_train` sets of examples to be then tested on the `target_test` and `non_target_test`, the model performs very well. When the training data is scrambled a bit, the model doesn't dip below 95%, as long as the ratios of the training/testing examples is the same.

It's biggest weakness is probably the fact, that it doesn't consider continuity. Each sound frame is independent and if they were interchanged randomly, the model would not notice. There is however undoubtedly a lot of information hidden in between the coefficients' order, so one way to make the model better would be to extract derivative of the coefficients, to see whether there's a rising or falling trend.

Another weak point of the system is the not ideal way of eliminating silence and cutting off the pop. The people recorded had to measure two seconds themselves, therefore – every time we cut, we either cut away a part containing voice or leave a bit of silence in. The method for cutting of silence generally is also flawed. Depending on the segment size, we can either cut tiny silent parts that are however still a part of speech, or big segments overall containing a lot of silence but also a piece of speech as well.

Perfecting the data preparation along with actually considering the fact that the order of the samples within a recording matters would undoubtedly lead to even better results and a more confident classifier.

4 Photo classifier

4.1 Image dataset analysis

The dataset is divided into two sections, Train and Dev. In the train section the ratio of targets and non-targets is 19:119, which is certainly not a balanced dataset. The amount of target needs to be increased at least seven fold. Another problem is that even after increasing the number of targets, the whole dataset would have less than 250 samples total.

It was therefore decided to use *Albumentations*[1]. *Albumentations*[1] is a widely popular library for fast and easy data augmentation. It's based on applying transformations on probability values. All of the transformations were created as a list of commands and were then applied during the training. Practically, each time the image is used during the training phase, a new image is created which has never been seen before, which is suitable in our situation.

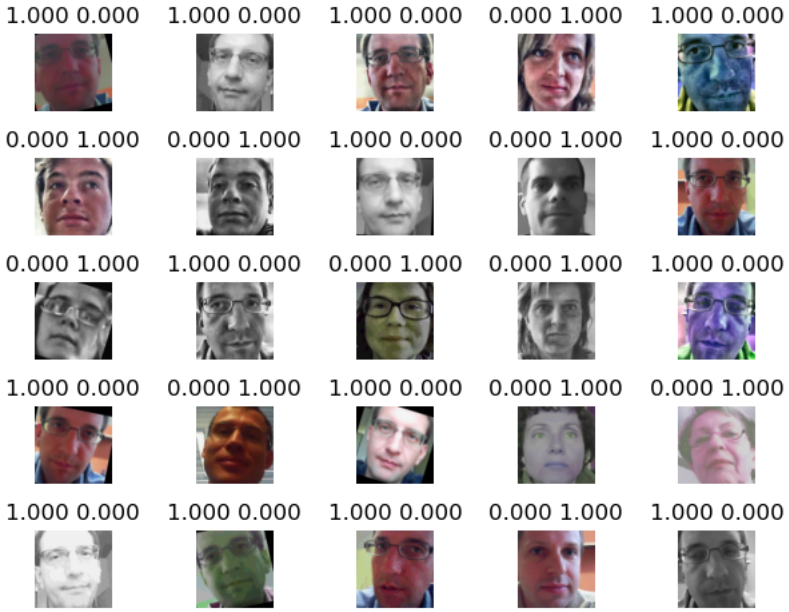

Figure 2: The example of data for training.

In Figure 2, we can see how the dataset for training can actually look like. Each picture had the effects of ColorJitter, ShiftScaleRotate, HorizontalFlip, CLAHE, Posterize, ToGray and ChannelShuffle applied randomly. Transformations can be edited and removed in config.py file.

4.2 Model

The model input's size is 80 * 80 * 3, which are the dimensions of the images from the dataset. The output of this model is one array the size of two numbers which are the resulting probabilites. The activation function is a Sigmoid and the loss function is the Cross Entropy Loss. The model is very simple and it can be train on every gpu. This model was trained on Nvidia 1050ti 4GB. The architecture of this model is described in Figure 3.

4.3 Training

First we tried to train the Convolution neural network on a dataset without transformations, to see if the model can predict at least something correctly. In each training phase we trained for 1000 epochs and saved checkpoints every 20 epochs, so we could have the best solutions and also

Type	size	output Channels
Conv2d Relu MaxPool2d	3×3 2×2	8
Conv2d Relu MaxPool2d	3×3 2×2	16
Conv2d Relu MaxPool2d	3×3 2×2	32
Conv2d Relu MaxPool2d	3×3 2×2	64
Flatten Linear Sigmoid	1600	num_classes

Figure 3: The architecture of the model.

avoid over-fitting. The model was over-fitted very quickly, only after 200 epochs or so. Also the count of false-alarms oscillated. The overall correctness of the model, however, was 85% on the validation dataset, which is not great, but not terrible either[2]. The training process can be seen on figure 6.

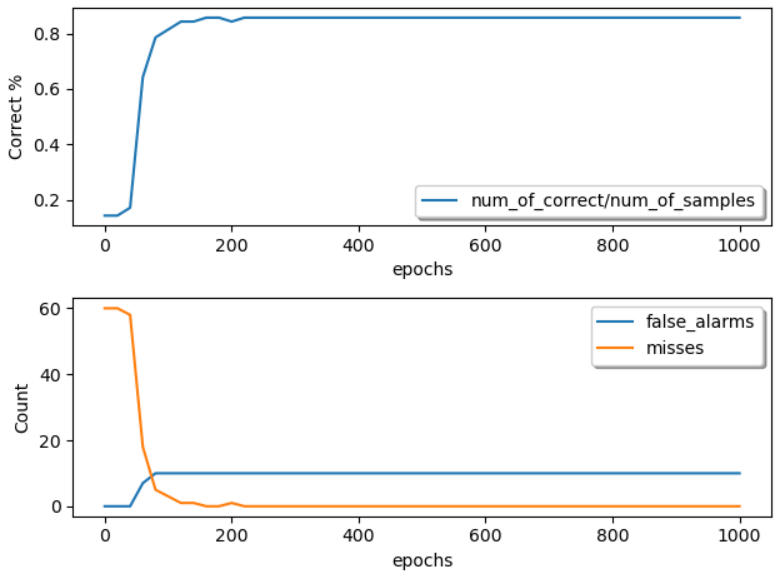

Figure 4: Training the CNN without any transformations on the dataset.

During the second stage of the model training we applied some of the recommend transformations for the YOLOv3 architecture for object detection. The main issue with these transformations is that we don't have to correlate to the best possible solution. The network also starts to yield good predictions only after twice as many epochs than without the usage of transformations. During this specific training we hit the overall correctness of 100%. So for the final evaluation, the

[2]https://www.imdb.com/title/tt7366338/

model from the 900th epoch was selected to create an evaluation. The results from the second stage of training can be found in Figure 5.

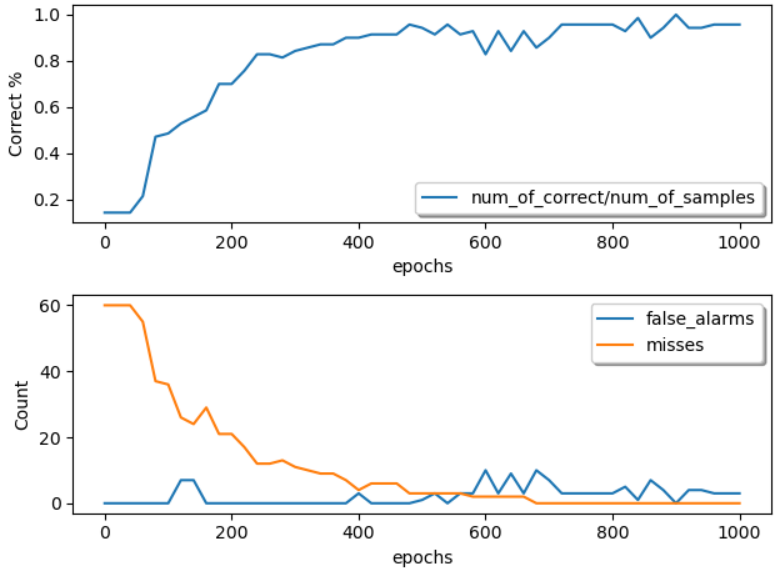

Figure 5: Training the CNN with transformations on the dataset.

The last graph is a comparison between the training with transformations and without transformations applied. The transformations clearly helps to better the classification.

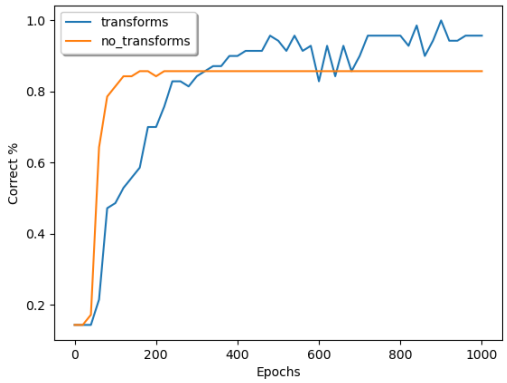

Figure 6: Comparison between the two models in one graph.

4.4 Results

The model was trained to yield a 100% score, but after a closer examination, some problems can be found. People most similar to the target person also possess glasses. It is also possible that the transformations and CNN layers lost some critical information about the target during the training, that would otherwise help better decision making. We can see the results in Figure 7, where the first row of numbers is the prediction and the second row are the target values.

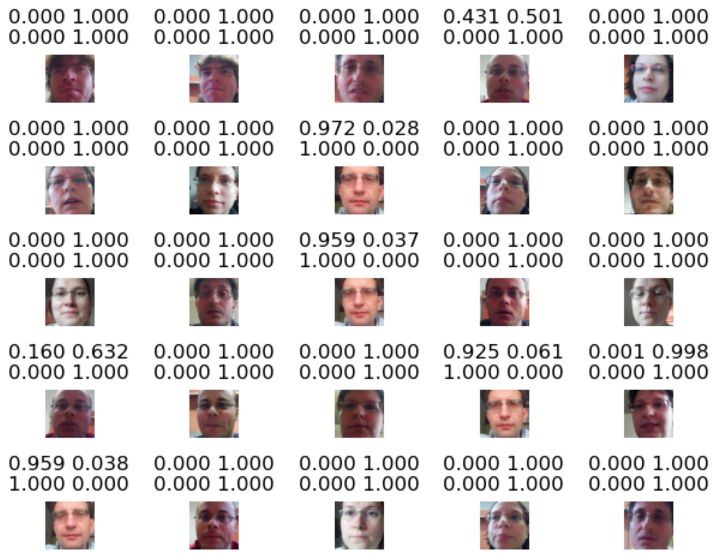

Figure 7: Validation examples.

5 Experiments

After training of convolution neural network and training Gaussian mixture model , we tried to combine these two trained model. The first option was that, we choose the model which was more certain in the prediction. The accuracy on given dataset was 97%. The second option, we let predict a model which has a lower peak value. The accuracy on given dataset was 85%. These evaluations are in results_eval like mixed_more_certain.txt and mixed_less_peak.txt.

6 What to do next

We should definitely try to use neural network for sound samples. Also instead of implementing own architecture from scratch, we should try AlexNet or other popular CNN architectures. At last, maybe download a bigger dataset of faces but from the perspective of this work, we couldn't do it .

References

[1] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[2] *Data augmentation*. Apr. 2022. URL: https://en.wikipedia.org/wiki/Data_augmentation.