VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Machine Learning and Recognition – Project

Model for classification of persons by face images and voice recordings

Marián Tarageľ (xtarag01)
Martin Horvát (xhorva17)

May 4, 2025

1 Introduction

The assignment of this project was to create models for classifying people according to images of their face and recordings of their voice. We were tasked with choosing an appropriate classifier model and implementing it in a programming language of our choice. Training was carried out on facial images and voice recordings of 31 people provided in 2 datasets `dev` and `train`.

- `dev/`: 2 images and 2 voice recordings per person
- `train/`: 6 images and 6 voice recordings per person

We can see that the datasets are very small, so the amount of data that could have been used for training was rather limited.

2 Model Design

We have developed three models, two for classification of face images and one for classification of voice recordings.

2.1 Images - Convolutional Neural Network (CNN)

The first image classification model we decided to implement was a Convolutional Neural Network (CNN).

Data augmentation: Due to the small amount of training data, we increased the amount available by augmenting selected image sets. The images were augmented by randomly modifying the images in the following metrics: rotation, vertical shift, horizontal shift, shear, zoom, horizontal flip, and brightness.

Network architecture: The CNN model is built using Keras Sequential API. It has three blocks of convolutional layers, where each block uses more filters (32, 64, then 128), followed by batch normalization, max pooling, and dropout to reduce overfitting. After these blocks, the features are flattened and passed through a dense layer with 256 units, followed by more batch normalization and dropout. The final output layer uses softmax to produce the probabilities for each class. The maximum probability class is chosen.

Training data: We have experimented with different combinations of `dev` and `train` mixed together, training only on `train` etc. We have found that the best results are achieved when we train the neural network on both the `dev` and `train` datasets, adding 20x augmented `train` data. Given the hyperparameters in Table 1 we have found that CNN performs subpar, given the limited data.

Parameter	**Value**	**Meaning**
`epochs`	30	number training epochs
`lr`	$1e^{-4}$	learning rate
`batch_size`	32	training batch size
`n_augments`	20	number of data augmentations
`loss`	CCE[1]	loss function
`opt`	Adam	optimizer

Table 1: Parameters used in Convolutional Neural Network

[1]Categorical Cross Entropy

2.2 Images - Support Vector Machine (SVM)

The second image classification model that we implemented was the Support Vector Machine (SVM).

Kernel function: We had evaluated multiple kernel functions. According to our evaluation, the best is linear. The evaluation strategy used for multiclass classification is one-vs-rest.

Training: The loss used for SVM was the Hinge loss. The maximum number of training epochs is 1500 or until the solution converges to tolerance $1e^{-2}$. The penalty for slack variables is $1e^{-2}$, which means that the solution must fit almost all training data. The parameters were found by `GridSearchCV`.

Parameter	Value	Meaning
`max_iter`	1500	maximum number of iterations to be run
`tol`	$1e^{-2}$	tolerance for stopping criteria
`C`	$1e^{-2}$	strength of the regularization
`multi_class`	one-vs-rest	determines the multi-class strategy

Table 2: Parameters used in Support Vector Machine

2.3 Voice - Gaussian Mixture Model (GMM)

For voice classification, we decided to implement a Gaussian Mixture Model (GMM) that classifies the speakers according to their recordings transformed to a Mel-frequency cepstral coefficient representation.

Preprocessing: Voice recordings have the first and last two seconds removed, as they usually do not feature the speaker talking. After clipping, the recording is split into a number of windows, each window that has signal energy under a given threshold, is dropped from the recording. This removes periods of silence during recording.

Transformation: Preprocessed voice recordings are transformed to their Mel-frequency cepstral coefficient representation. We have used the functions provided in `ikrlib`. The transformation has the following parameters, which are summarized in Table 3.

Parameter	Value	Meaning
`num_gauss`	6	number of gaussian distributions
`epochs`	30	number of training cycles
`energy_level`	15	signal energy cutoff threshold
`window`	400	window length for frame (in samples)
`noverlap`	240	overlapping between frames (in samples)
`nfft`	512	number of frequency points used to calculate the FFT
`nbanks`	23	numer of mel filter bank bands
`nceps`	13	number of MEL cepstral coefficients

Table 3: Parameters used in Gaussian Mixture Model

Training: The EM algorithm is used to train a GMM model for each speaker.

3 Running

The required Python packages are `numpy`, `matplotlib`, `keras`, `tensorflow`, `python-opencv`, `scipy`, `scikit-learn`.

The project assumes the following folder structure. Where SRC/ containts the implementations of individual models, and data/ contains the dev, test and eval datasets.

```
./
 |-- SRC/
 |      |-- image_cnn.py
 |      |-- image_svm.py
 |      |-- voice_gmm.py
 |-- data/
 |      |-- dev/
 |      |-- train/
 |      |-- eval/
 |-- dokumentace.pdf
```

Figure 1: Expected directory structure.

Before running, we suggest creating a virtual environment with all the necessary packages.

```
$ virtualenv venv
$ source venv/bin/activate
(venv) $ python3 -m pip install numpy matplotlib keras tensorflow
    python-opencv scipy scikit-learn
(venv) $ python3 SRC/image_cnn.py
```

Figure 2: Run image_cnn.py classifier.

4 Model Evaluation

We evaluated the performance of the models by testing their prediction accuracy on the dev dataset. The best performance, the average performance for 10 runs is summarized Table 4.

Model	Best (%)	Average (%)[2]
Image - CNN	70.4	43.3
Image - SVM	66.4	62.8
Voice - GMM	89.4	84.2

Table 4: Model Performance

4.1 Images - Convolutional Neural Network (CNN)

The CNN model achieves variable performance, based on data inputted. It cannot be said that it performs well and will likely perform poorly on unobserved test data. It is known that CNN's require large amounts of data to function well. The provided dataset is very small to train a CNN from scratch and usually results in an overfit network.

[2]Average performance for 10 runs

4.2 Images - Support Vector Machine (SVM)

The SVM model performs relatively well after using grid search to find optimal parameters. It consistently achieves performance of $\sim$62%. The parameters found after training can be seen in Table 2. We expect to achieve similar performance on the evaluation dataset. Better performance could be achieved on with different data representation . We note, that a simpler machine learning model achieves better performance than a large CNN.

4.3 Voice - Gaussian Mixture Model (GMM)

We evaluated the performance of the model by its accuracy in classifying speakers in the `dev` dataset. We train the model using the `train` dataset. The model has an average classification accuracy of $\sim$84% (mean of 10 model runs). We have found that the number of gaussian distributions (`num_gauss`), number of epochs (`epochs`) and window signal energy cutoff threshold (`energy_level`) parameters have the largest impact on classifier performance (in decreasing order). The parameters, which we have found to work best, can be seen in Table 3.