Speaker and Face Recognition (SUR project)

Author: Jakub Ryšánek (xrysan05)

May 5, 2025

Project structure and setup

The project contains documentation in the Dokumentace.pdf and files

```
result_audio.txt
result_picture.txt
```

which are results from the speaker and face recognition on the test data. Then in the SRC/ folder there are python scripts and Makefile with requirements for the project. To install the required dependencies and setup virtual environment run:

```
make venv
```

To remove the environment run:

```
make clean
```

The **audio.py** and **picture.py** file are training models from the train folder which was provided. The **picture_aug.py** script was used to augmented pictures from the training data. The **evaluation.py** script was used for creating the result text files for final evaluation.

Image Recognition System

Feature Extraction

Each image is processed using the Histogram of Oriented Gradients (HOG) method applied on the luminance (Y) channel of the YCrCb color space. In addition, color information is partially retained via histograms on the Cr and Cb channels. This approach resulted in higher accuracy then with only grayscale parameters.

For the HOG parameters the best result I got with orientations=12, pixels_per_cell=(6,6), cells_per_block=(2,2). I inspected multiple combinations of these parameters and this gives the best results.

Augmentation

Due to the lack of training data I used augmentation to create 5 more pictures out of one original. The `albumentations` library was used for augmentation. The five augmented variants were generated using a combination of:

- Horizontal flipping (p=0.5)
- Affine transformations (scale, translation, rotation)
- Brightness adjustments
- Saturation shifts

The original and augmented images were saved to a new `train_aug/` folder and used for training.

Classification

Features were standardized and reduced using PCA (target variance retained: 85%). In many instances tried rbf kernel for SVM but a linear SVM classifier always resulted in higher accuracy. Evaluation on `dev` showed improved performance after augmentation and PCA.

Audio Recognition System

Feature Extraction

We used MFCC features with delta and delta-delta coefficients, along with RMS energy. Each recording was transformed into a feature matrix of shape (frames, features). Feature dimensionality: 61 (20 MFCCs + 20 delta + 20 delta2 + 1 RMS).

GMM Supervectors

A global Gaussian Mixture Model (GMM) with 24 diagonal components was trained on all MFCC frames from the training set. At first I tried setting covariance_type to ”full” which resulted in the creation of a more complex model and it was taking longer to train. But in later staged the covariance_type ”diag” gave better performance and the training was faster. Each file was then represented by a supervector: the weighted average of MFCCs per component based on posterior probabilities.

Classification

The resulting supervectors were standardized and used to train an SVM classifier with RBF kernel. Also here tried to use linear kernel but was rbf better. For the n_components parameter the 32 gave better accuracy but I though that with this little training data the higher number of components the higher risk of overfitting so I ended up with 24.

Evaluation and Output Format

Each system generates a result file with the following structure:

- Filename (without extension)
- Predicted class (1 to 31)
- 31 log-probability scores

Both models use classifiers that support probabilistic output, so real-valued log-probabilities are computed using `predict_log_proba()`.

Observations and Improvements

- Data augmentation greatly helped image classification but for audio the augmentation did not bring better results. Maybe the augmentation were too aggressive but I tried to lower the parameters but it did not helped. That is why I was training only on the original audio training data.
- I think it could help for face recognition to add some other HOG features.
- In the results I saw that the speaker recognition model was inclined to choose speaker 20. I could have inspect whish speaker the models are choosing too often and balance the data.
- To increace overall performace I could essemble these two models together.