SUR 2024/2025 Project

Login: XSYNAK03

---

1. Project Summary

The goal of this project is to develop a system capable of identifying 31 individuals based on face images and voice recordings. Two independent pipelines were implemented and improved: one for image-based recognition, and one for speaker recognition using audio signals.

2 Data Split

- **Training Data**: `train/1` to `train/31`
- **Testing Data**: `dev/1` to `dev/31`

---

2. Image Recognition System

At first, a basic SVM classifier was implemented using raw pixel values from grayscale images, with dimensionality reduction performed via PCA. This base had around 19 % accuracy.

---

3. Improvements tested

Feature Extraction

We experimented with several types of image feature extractions to find the one performing the best:

Feature Type	Description	Accuracy
Raw Pixels	Flattened grayscale 128×128 images.	19%
HOG (Histogram of Oriented Gradients)	Gradient orientation histograms, tuned via grid search.	63–75%
HOG + LBP + Raw Pixels	Concatenation of multiple descriptors, overfitted.	45%
LBP	Local Binary Pattern histograms.	48%

We akso tested several parameters for HOG with the best configuration being:

- `orientations=9`
- `pixels_per_cell=(8, 8)`
- `cells_per_block=(2, 2)`
- `block_norm='L2-Hys'`

Data Augmentation

To help the model generalize better and become better at dealing with slight variations, data augmentation techniques were applied to the training images.

- **Without augmentation**: ~63.2% accuracy
- **With augmentation**: Horizontal flip and 10 degree rotations increased accuracy to ~75.8%. More adcanced augmentations were also tested(scaling, noise injection, contrast jitter) but did not yield better results.

Classifiers Compared

Several classifiers were tested:

Classifier	Accuracy
Linear SVM	~75.8%
RBF SVM	~75.8%
KNN (k=35)	~61.6%
Random Forest	~54.2%
MLP	~45.2%

Linear SVM with C=0.01 was chosen for performance and speed.

Dimensionality Reduction

- Features were standardized using StandardScaler
- PCA tested for 25–300 components
- Best result at 100 components giving slight improvement

---

4 Final Image Pipeline

The final image recognition system is based on a Histogram of Oriented Gradients (HOG) feature extraction followed by PCA dimensionality reduction and a Linear Support Vector Machine (SVM) classifier.

Pipeline Overview

1. Load images from train/ and dev folders.
2. Preprocess: resize each image to 128×128 and convert to grayscale.
3. Augment training images with horizontal flips and 10 degree rotations.
4. Extract HOG features, capturing the structure and edge orientations in the face.
5. Normalize features using StandardScaler and reduce dimensionality using PCA.
6. Train a Linear SVM.
7. Evaluate model on the dev set.

Final Accuracy: ~77%

---

5. Voice Recognition System

The first base system used MFCC features and trained 1-component GMMs per speaker using diagonal covariance matrices. No normalization or parameter tuning was applied initially. This simple setup achieved ~35% accuracy.

Iprovements tested

GMM Classifier parameters

Different number of GMM components and full/diagonal covariences were tested notable being:

Configuration	Accuracy
Diagonal, M=1	~35%
Full, M=1	~54%
Full, M=10	~60%

Full matrices significantly improved performance. Best result was with M = 10 (tested 1-30) components.

MFCC Parameter Tuning

To improve the model’s representational power, we conducted extensive testing with over 70 different MFCC configurations. The following parameters were varied during tuning:

- Window size
  - Overlap
  - FFT size
  - Number of Mel filterbanks
  - Number of cepstral coefficients

The best-performing configuration was found to be: win = 512, noverlap = 256, nfft = 512, nbanks = 23, and nceps = 16.

This configuration achieved an accuracy of approximately 83%.

CMVN Normalization

To stabilize feature distributions across speakers and sessions, we applied CMVN (Cepstral Mean and Variance Normalization) after MFCC extraction. The final accuracy after this was 86%

Additional Improvements Tested

After establishing the strong CMVN + MFCC + GMM baseline, we explored a range of advanced strategies. However, most offered little or no improvement.

Improvement	Description	Impact
Delta/Delta-Delta	Added time derivatives	Decrease
Score Normalization	z-score normalization of log-likelihoods	No gain
Ensemble GMMs	Multiple GMMs per speaker, score averaging	No gain
Jackknife	Leave-one-session-out cross-validation	No gain

UBM + MAP Adaptation	Universal Background Model + adaptation	Decrease

---

6. Final Configuration Summary

Component	Configuration
GMM	M = 10, full covariance
MFCC	(512, 256, 512, 23, 16)
CMVN	Enabled
Accuracy	86%

The final voice recognition pipeline is implemented using a straightforward GMM-based classification strategy:

1. Load .wav audio data from train/ and dev/ folders.
2. Extract MFCC features using the mfcc function from IKR_demos_py:
    - Window size = 512
    - Overlap = 256
    - NFFT = 512
    - 23 Mel filterbanks
    - 16 cepstral coefficients
3. Apply Cepstral Mean and Variance Normalization (CMVN) to normalize speaker variability.
4. For each speaker, train a GMM with 10 components and full covariance matrices.
5. During classification, for each test audio segment:
    - Compute log-likelihood scores from all 31 GMMs
    - Choose the speaker with the highest score.

The GMMs are initialized with random means from the data and trained using EM until convergence or a fixed number of iterations.

Final testing yields a classification accuracy of approximately 86%.

---

7. How to Run

Requirements:

```
pip3 install -r requirements.txt
```

Voice Recognition Model

Usage:

```
python3 sur_voice_GMM.py
```

It will train on data in test dir measure accuracy on dev dir and evaluate eval. Train, dev and eval need to be in the same directory as source files and the results are saved in results_voice_gmm.txt in the same directory. IKR_demos_py folder also needs to be at same directory as some functions are sourced from the demo(adjusted for python3).

Image Recognition Model

Usage:

```
python3 sur_image_SVM.py
```

It will train on data in test dir measure accuracy on dev dir and evaluate eval. Train, dev and eval need to be in the same directory as source files and the results are saved in results_image_linear.txt in the same directory. Directory structure :

```
xsynak03/
│
├── src/
│   ├── IKR_demos_py/
│   │   └── ikrlib.py
│   │
│   ├── sur_voice_GMM.py
│   ├── sur_image_SVM.py
│   └── requirements.txt
│
└── documentation.pdf
```

So eval dev and train need to be placed in src and output will also be in src.