Documentation for SUR Project

Author: Leoš Nevoral
contact: xnevor03@stud.fit.vutbr.cz

Models Used

1. Audio Model

The audio model is based on a **Gaussian Mixture Model (GMM)**. It uses MFCC (Mel-Frequency Cepstral Coefficients) features extracted from audio files. The GMM is trained with different covariance types (`spherical`, `tied`, `diag`, `full`) and the number of components is tuned to find the best configuration.

2. Image Model

The image model is a **Convolutional Neural Network (CNN)** implemented in PyTorch Lightning. It has the following configurable parameters:

- **Number of Layers**: The number of convolutional layers can be adjusted.
- **Kernel Sizes**: The kernel size for each layer can be customized.
- **Pooling and Activation Decisions**: Each layer can optionally include pooling and ReLU activation.
- **Fully Connected Layers**: The fully connected layers dynamically reduce the feature size until the number of classes is reached.
- **Rest**: Parameters (outgoin channel size for convolution and dropout odds) can be adjusted only directly inside the code.

It has so many configurable parameters mostly because it did'nt produce satisfying results (35% accuracy on dev), thus a lot of testing was done. I have also tried to import few famous CNN models for experimenting but also with little to no success. All the CNN models were overfitting on training data without producing solid results on test data.

Data Preparation and Augmentation

1. Audio Data

- **MFCC Features**: Audio files are converted to MFCC features using the `wav16khz2mfcc` function from the `ikrlib` library.
- **Audio Cutter**: Audio files are trimmed by 1.5-second from the beginning to get rid of the camera imperfection.

2. Image Data

- **Augmentation**: Images are augmented using the following techniques:
  - **Rotation**: Random rotation within a range of ±45 degrees.

    - **Coarse Dropout**: Randomly masking out sections of the image.
    - **Gaussian Noise**: Adding random noise to the image.
- **Normalization**: Images are normalized to the range [0, 1].

Installation Instructions

Prerequisites

Ensure you have the following installed:

- Python 3.12 or higher
- `pip` (Python package manager)
- `make` (build automation tool)
- The given datasets

Installation Steps

1. Install the required Python packages:

   ```
   make install
   ```

   This will install all dependencies listed in the `requirements.txt` file.

Running Training Scripts

1. Audio Model Training

To train and test the models (the testing functions are commented and arent probably of much interest to anyone, because the best found models paramateres are insterted for learning):

```
make run
```

This will train the GMM model on the audio dataset and save the best model as `best_gmm_audio.pickle`, and the CNN model on the image dataset and save the best model as `CNN_image.pickle`.

Running Evaluation Script

To evaluate the trained models on the evaluation dataset:

```
make eval
```

This will:

1. DISCLAIMER: it will cut all the audio 1,5 second shorter, if using multiple times, make sure to commment the function audio_cutter

2. Evaluate the audio model on the evaluation dataset and save the results in `audio_GMM`.
3. Evaluate the image model on the evaluation dataset and save the results in `image_CNN`.

Makefile

The `Makefile` includes the following targets:

- **`install`**: Installs all required Python packages.
- **`evaluate`**: Runs the evaluation script.

Current Limitations

CNN Model

The CNN model currently does not perform well on the image dataset. The architecture and hyperparameters need further tuning to improve its performance. Even tho a lot of tuning was done. I was kind of defeated, that I didnt manage to produce a solid model... But I think the scripts can still showcase I put in the time.