Course Name: CS360 Machine Learning
Instructor: Yik-Cheung Tam
Student Name: Yanxuan Dong
NetID: yd2179
Date: May 12, 2024

CS360 Machine Learning Final Competition Report

Introduction

This report concludes the methods and processes employed in the machine learning competition. The task involved classifying audio snippets into four categories based on the type of voice present: no voice, male voice, female voice, and multiple voice.

Data Preprocessing & Data Augmentation

The audio files were loaded with the librosa library and converted to mono (one channel). This process included averaging channels if the audio had more than one. A resampling to a consistent rate of 16000 Hz was applied in the beginning. However, since no significant effects were found, this resampling was removed later.

Several data augmentation techniques were employed to enhance the performance of the model. While some augmentation methods were proved making a negative impact to the performance and hence removed later (e.x.: Pitch Adjustment - properly shifted the pitch of the audio). These augmentations were applied randomly with a certain probability. Below are effective augmentation methods that were preserved in the final version:

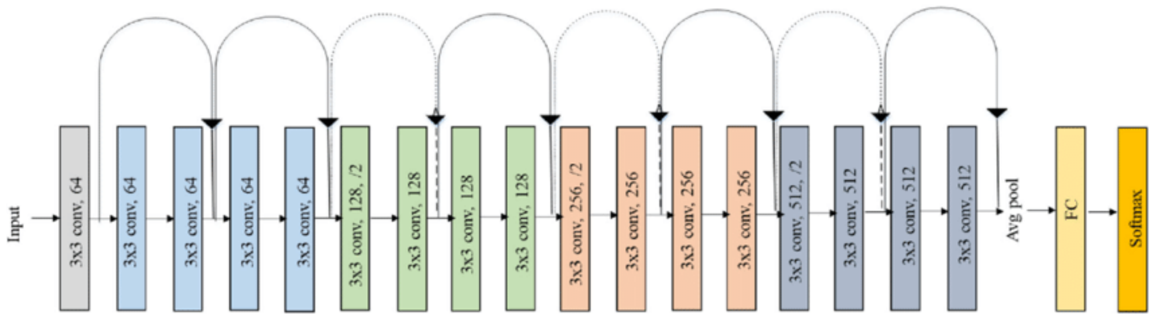
- Echo: Added a echo effect to the audio (effect duration is slight compared to the audio's length)
- White Noise: Added random white noise
- Speed Up: Randomly adjusted the speed of the audio playback to simulate faster or slower speaking rates (0.8 - 1.2x).
- Volume Increase: Randomly adjusted the volume to simulate loudness variations.

Afterwards, a spectrogram conversion was conducted to make neural networks like CNN and ResNet better fit in this task for their expertise in dealing with graphs. Audio signals were transformed into spectrograms using the Mel scale, which is more perceptually relevant than a linear scale. This transformation aids the model in capturing important frequency-based features. The overall data pre-processing was time-consuming, which took 4-5 hours to complete (varies due to resources allocated). Therefore, the processed tensors were saved to avoid time wasting in the following training process.

Model Architecture

The model used was a modified ResNet18, adapted for one-dimensional audio input (mono, 1 channel) and four channel output representing the four target classes. No pretrained parameters were used in the training process. Before deploying ResNet18, CNN model was also used and tuned several times but ultimately achieved a best performance of 69% accuracy on test set, which is less favorable than ResNet18. Below picture is the original settings for ResNet18, but some changes were made to fit this specific dataset:

- Custom Convolution Layer: The first convolutional layer was modified to accept a single-channel input.
- Fully Connected Layer: The output layer was modified to classify four categories of audio.



Training Procedure

The original training dataset was split into training and validation sets (1000 audios), data was loaded in batches using PyTorch's DataLoader. GPUs on Colab and RTX8000 were used for training this model. As mentioned above, the main time bottleneck for this training is the on-cpu preprocessing and augmentation part. When loading the saved processed data, on Colab it took between half an hour to finish training, and on RTX8000 it took within 10 minutes to finish training. But in the submitted version, I did not separate preprocessing and training.

Cross-entropy loss was used for its suitability for multi-class problems. Adam optimizer was used with various learning rate schedulers to adjust the learning rate during training based on validation loss. The model was trained for multiple epochs, with intermittent validation to monitor overfitting and performance on unseen data (epoch = 40 gives an impressive performance on validation set, but has a tendency of overfit. Therefore, several smaller epochs were tried and epoch = 20 was selected in the end).

Running Steps:

1. `pip install librosa` before executing the program
2. Unzip the `train.zip` file and enter the file
3. Put the train and test data folder `train_mp3s` and `test_mp3s` into the `data` folder located in the unzipped folder
4. In terminal, run `python -m train`
5. After training is completed, the prediction csv file can be found in the `result/Today/` folder