CSCI-SHU 360 Machine Learning Final Competition

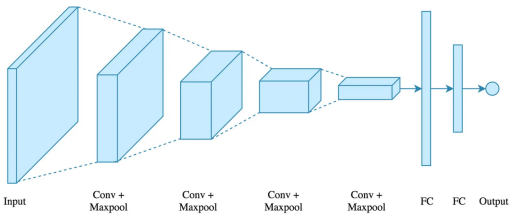
Cissy Xie

Task Overview

Develop a model to classify song snippets into one of four categories: no voices present, male voice present, female voice present, or more than one person's voices present.

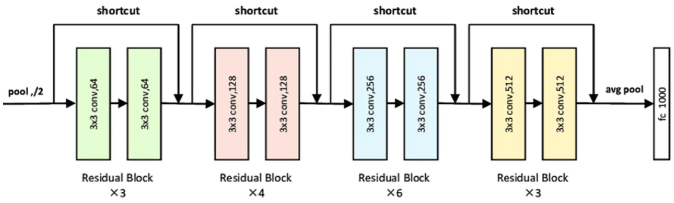
Models Overview

4 Layer CNN



CNN 4-layers Architecture

Resnet34



Resnet34 Architecture

Improvement

1. Data Pre-processing

   I transform the data into Mel spectrogram using `Librosa` , then turn the numpy array into a tensor and add a dimension on the tensor. After Transforming, the size of the unbatched input tensor is `torch.size[1,128,130]`

```
mel_spec = librosa.feature.melspectrogram(y=y, sr=sr)
mel_spec_db = librosa.power_to_db(mel_spec, ref=np.max)
```

2. Data Augmentation

   (1) Time Mask/ Frequency Mask/ Reverse:

   I generate time and frequency mask on the Mel spectrogram and reverse the audio waveform by horizontally flipping the audio samples.

   (2) Mix-up

   I blend pairs of input data samples and their labels, generating mixed inputs. They are used to compute the loss, which is a combination of losses calculated for both original and shuffled

labels, weighted by the mixing coefficient.

(3) Up-sampling based on labels' weight

To address class imbalance in the dataset, an up-sampling technique based on the weight of each class label is implemented. I calculate the class weights, converted to a `PyTorch` tensor, and passed to the `nn.CrossEntropyLoss` criterion during model training. This approach helped mitigate the effects of class imbalance by assigning higher weights to underrepresented classes.

(4) Human voice separation

I use tools to separate human voice from the song snippets to improve model performance

3. Model Improvement

   (1) Change the Input layer of Resnet:

   Change the kernel size, stride and padding on the input layer of the Resnet34 from 7,2,3 to 3,1,1

   (2) Ensemble the models:

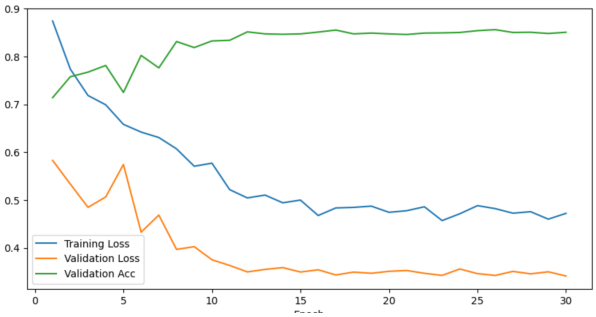
   I ensemble the CNN and ResNet34 with weights [0.5, 0.5] and sum their `softmax` **(**probabilities) to predict the class. The test accuracy for each model was 0.7691 and 0.7817 respectively, but increased to 0.7944 after ensembling.

```
outputs_cnn = torch.softmax(model1(inputs), dim=1) * weights[0]
outputs_resnet = torch.softmax(model2(inputs), dim=1) * weights[1]
outputs_ensemble = outputs_cnn + outputs_resnet
_, predicted = torch.max(outputs_ensemble.data, 1)
```

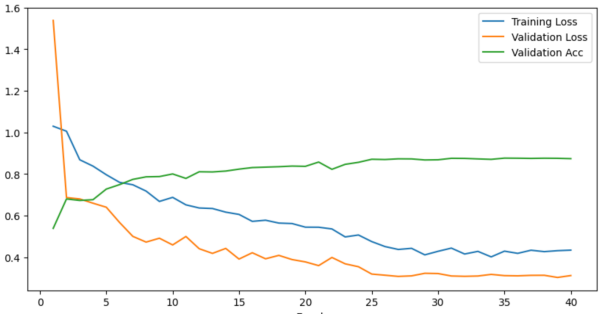
4. Choice of Hyper-Parameters/Scheduler

   After experiments on the hyper-parameters/schedulers, I finally decided on the hyper-parameters/schedulers as shown below:

Model	Scheduler	Batch Size	Initial lr	Mixup Alpha	Model Weights
CNN 4-layers	MultiStepLR	16	0.001	0.4	-
Resnet 34	ReduceLROnPlateau	16	0.005	0.4	-
Resnet34+CNN	-	-	-	-	0.5, 0.5



CNN 4-layers Train/Valid Loss and Valid Accuracy



Resnet34 Train/Valid Loss and Valid Accuracy

Model	Epoch	Train Loss	Validation Loss	Validation Accuracy	Test Accuracy
CNN 4-layers	30	0.3828	0.3531	0.8579	0.7691
Resnet 34	24	0.4228	0.2814	0.8852	0.7817
Resnet34+CNN	-	-	-	-	0.7944

Requisites

In order to run my code, make sure to install packages below

```
pip install torch
pip install librosa
```

Remember to modify the file directory in the Jupyter Notebook

```
song_dir = "train_aug"
label_dir = "train_label.txt"
test_dir = 'test_aug'
```