CSCI-SHU 360 Machine Learning
Report of Final Competition

Yufeng -AeoLian- Xu yx3038@nyu.edu

May 7, 2024

1 Guideline

This section aims to provide a navigation of this repository.

- Folder **src** contains the source code of the final competition.
- File **main.py** is the main file for training of the models. File **data_utils.py** contains the functions for data processing as well as the customized dataset class. File **models.py** contains the customized ResNet18. File **infer.py** utilizes the ensemble of several trained model to make predictions.
- Folder **write-up** contains the report on the final competition.
- Folder **checkpoints** records the parameters, prediction, as well as best validation accurcay so far for each of the models.
- Folder **data** contains the original as well as processed data.
- File **prediction.csv** is the prediction made by **infer.py**.

2 Data Preprocessing

We called a Python library called **spleeter** to separate vocal and instrumental parts in each music snippet, and chose and vocal parts as our training and evaluation data. Afterwards, we used **torchaudio** to load and transform(or augment) the vocal files.

3 Models

The models I have tried for the final competition include: ResNet18, ResNet50, ResNet101, ResNext, DenseNet201, Vision Transformer (ViT), Long Short Term Memory (LSTM). Among all these models, ResNet18 demonstrates excellent performance and the best performance-efficiency tradeoff. Therefore, the final results are based on resnet18. An unexpected observation is that the resnet18 network implemented by myself has better performance than the built-in one from torch, although according to **model.modules** the two have exactly the same components.

```
from spleeter.separator import Separator

separator = Separator('spleeter:2stems')

def separate(idx, mp3_dir, output_dir):

    # Define the input audio file path
    audio_file = os.path.join(mp3_dir, f'{idx}.mp3')

    # Perform separation
    separator.separate_to_file(audio_file, output_dir)
```

```
audio_transform = transforms.Compose([
      torchaudio.transforms.MelSpectrogram(sample_rate = 22050, n_fft = 2048),
      torchaudio.transforms.AmplitudeToDB(),
  ])

  audio_transform2 = transforms.Compose([
      torchaudio.transforms.MelSpectrogram(sample_rate = 22050, n_fft = 2048),
      torchaudio.transforms.AmplitudeToDB(),
      torchaudio.transforms.TimeMasking(time_mask_param = 50, iid_masks = True, p = 1.0)
  ])


  audio_transform3 = transforms.Compose([
      torchaudio.transforms.MelSpectrogram(sample_rate = 22050, n_fft = 2048),
      torchaudio.transforms.AmplitudeToDB(),
      torchaudio.transforms.FrequencyMasking(freq_mask_param = 100, iid_masks = True)
  ])
```

```
epochs = 24

optimizer = optim.SGD(model.parameters(), lr = 1e-3, momentum = 0.9, weight_decay = 1e-3)

scheduler = optim.lr_scheduler.MultiStepLR(optimizer, milestones = [epochs // 4, epochs // 2, epochs * 3 // 4],
gamma = 0.1, last_epoch = -1)
```

4 Training Setups

Vocal Separation

Data Transformatation

Optimization

5 Discussion