Music Genre Classification

Hyunsung Cho

April 30, 2019

1 Algorithm Description

The program consists of the dataset and six main python scripts (hparams.py, audio_augmentation.py, feature_extraction.py, data_manager.py, models.py, and train_test.py) for music genre classification and two helper scripts for the ease of testing (train_test_multi.py and results.py). The way that the program functions is explained in this section one by one.

1.1 Dataset

The program uses a subset of the GTZAN dataset made by Tzanetakis and Cook [1]. It is an audio dataset containing a range of musical genres. The used subset consists of 8 classes of musical genres (classical, country, disco, hiphop, jazz, metal, pop, and reggae) and is split into training, validation and test sets. There are 353 audio samples in the training set, 150 in the validation, and 227 in test sets resulting in a total of 730 audio samples.

1.2 Audio Augmentation

In order to increase the diversity of training data, data augmentation is applied on the training dataset. Running the audio_augmentation.py augments the original data by adding randomly generated numerical noise value to the data and saves the augmented data into a separate directory. Therefore after augmentation, there are 706 training samples, 150 validation samples, and 227 test samples, resulting in a total of 1083 audio samples.

1.3 Feature Extraction

As the first step towards instrument classification, the program performs feature extraction in `feature_extraction.py`. It loads each audio file of the dataset and calculates the melspectrogram for each.

1.4 Segmentation

In order to generate more data for training and validation data, the program segments the extracted melspectrogram into smaller sizes of 128 frames which correspond to 3 seconds of audio. This is done as a part of feature_extraction.py after the melspectrogram calculation and before saving the calculated value into a feature file. The length of original audio files is around 30 seconds which corresponds to approximately 1291 frames given the sampling rate of 22050 Hz and hop size of 512 ($audio\ length = feature\ length * hop\ size/sample\ rate$). The code cuts the whole file into smaller files with the length of 128 frames (the feature length) each and throws away the remaining part after dividing the original file. Since the melspectrogram has the dimension of 128 mels, each input audio sample has the size of 128x128. Test data are not segmented in this step. Later in the classifier testing step, it slices the 30-second audio file, makes prediction for each slice or segment, and performs a majority voting over the segments to determine the prediction result for the 30-second data in train_test.py.

1.5 Training the Deep Neural Network Model

The model for deep neural network is defined in models.py. In `train_test.py` file, the training and testing take place using the extracted and summarized features in the above sections.

Model

The program uses a 2-dimensional convolutional neural network (2D CNN) model. The model consists of two convolutional layers and one fully connected layer. The first convolutional layer takes 128 input channels and outputs 64 channels with the kernel size of 3, stride and padding sizes of 1. Then batch normalization and ReLU activation are applied in sequence. Batch normalization does the role of preventing overfitting, and ReLU activation determines whether to activate the neuron or not. The model then performs max pooling over the size of 4x4. The second convolutional layer takes 64 input channels and processes 128 output channels with the kernel size of 3, stride and padding sizes of 1. Batch normalization, ReLU activation, and max pooling are applied in the same way. Then it passes through a fully connected layer from 128 channels to 8 which is the number of genres to classify. Lastly, log softmax is applied to obtain the probabilities that sum to 1.

Train and Test

Given the preprocessed input data and the neural network model, the program trains the neural network and makes prediction for the test data. It uses SGD with Nesterov as the optimizer and cross entropy loss function. Hyperparameters used in the program are listed in Table 1.5. It also utilizes batches for

Hyperparametser	Value
Sampling rate	22050
FFT size	2048
Window size	2048
Hop size	512
mels	128
Feature length	128
Batch size	16
epochs	100
Learning rate	$1e^{-3}$
Stopping rate	$1e^{-5}$
Weight decay	$1e^{-6}$
Momentum	0.9
Factor	0.2
Patience	10

Table 1: Hyperparameter settings

Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8
81.94%	81.94%	81.06%	80.62%	79.30%	84.14%	80.18%	78.85%

Table 2: Musical genre classification test result

training. Sixteen audio samples are grouped into a batch and used as the input for the model.

2 Experiments and Results

2.1 Performance Evaluation Method

A helper Python script train_test_multi.py is used to test the model efficiently by running the train_test.py in eight GPUs in parallel at the same time. The other helper script results.py prints out the result from these eight times of testing in a pretty format.

2.2 Results

Running the described model 8 times in each GPU showed the highest test accuracy of 84.14% and an average accuracy of 81.00% over the 8 trials with the standard deviation of 1.58%. All values from the 8 trials are reported in Table 2.2, and the results are summarized in Table 2.2.

Average	SD	Best
81.00%	1.58	84.14%

Table 3: Test result summary

3 Discussion

3.1 Audio Segmentation

Segmenting the training and validation data played the most significant role in increasing the performance of the musical genre classification model. Training with audio data of short duration can achieve high accuracy on classifying audio samples of longer duration.

3.2 Feature Selection

The work presented in this report utilizes only the melspectrogram of audio signals. However, reflecting on the machine learning task from Homework 1, adding more diverse set of features such as MFCC, chroma, and temporal feature can have higher effects.

References

[1] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.