The Happy and Sad Music Recognition Using Simple 2D CNN.

Ziyi Liang *

1Department of Computer Science, The Ocean University of China, Qingdao 266000, China

* Corresponding Author Email: Lzy4805@stu.ouc.edu.cn

Abstract. At the present time is the rapid development of artificial intelligence, deep learning is an important part of it, the use of neural networks for deep learning in which has its own unique advantages. Music, as an important component of the human emotional world, has its related applications in various fields. This paper focuses on the classification of Happy and Sad in music emotion. The article constructs a simple VGG-like neural network model by using 2D convolutional neural network through Mel's spectrum processing. The data files in wav format are fed into the network for training and evaluated by observing the accuracy and loss function values. Eventually the model was able to perform some recognition but failed to achieve good recognition due to some of its limitations. By constructing such a neural network, the simple VGG-like convolutional neural network model is explored for simple music emotion recognition to try if a robust model can be built.

Keywords: Music Recognition, 2D convolutional neural network, artificial intelligence, the dataset.

1. Introduction

Music is an art of expression. It graves in human beings' genes and delivers through every generation. It is the most universally understood language in the world, transcending geographical boundaries. The music sounds contain plenty of feelings from the composers and their creator. It's considered as the central part of a music piece. The emotion among the music can influence people's current senses. Happy music brings enjoyments while sad music contains gloom. Thus, specific music involving these types of sensibilities has already been use in medical domain such as therapy treatment for depression. It shows great potential to improving emotion regulation [1]. In addition, the music emotion can be a tag in music classification or music retrieval [2-4]. The power of musical emotion is evident. With the development of computer technology, music emotion recognition (MER) becomes a potential method for application like music therapy, auto-tagging [2] etc. The music emotion computing becomes more important in the period of data. It has extensive applications in music retrieval and recommendation. As the speeding development in artificial intelligence, the combination of AI and music emotional computing are increasingly interconnected. Developers use chatbot to create an interaction between the system and the users, where the robot can understand contest and respond to the users [5]. From these perspectives, music emotion recognition has important research implications.

Mel spectrograms and log amplitude spectra are used for audio signal processing. By capturing the frequency distribution and energy variations of audio signals, the use of Mel spectrograms can improve feature representation. On the other hand, noise immunity can be improved by using logarithmic amplitude spectra. These are based on the representation of the time axis and the frequency axis. The time axis is divided into several windows, and frequency distribution and amplitude information are calculated within each window [6, 7]. Convolutional neural networks are widely used in the field of machine learning and to process raw data. One of the key advantages of this approach is its ability to learn high-dimensional feature representations directly from raw data and propagate them through multiple layers, enabling end-to-end learning. While initially popularized in the field of image recognition, this architecture has proven to be highly adaptable, finding applications in various domains as technology has advanced [8, 9]. In this paper, a VGG-like neural network structure is used to increase the depth of the network by constructing a series of small-sized convolutional kernels for feature extraction and stacking the same-sized convolutional layers multiple times. After the convolutional layers, the spatial size of the feature map is reduced by a pooling

operation. VGG uses maximum pooling, typically a 2x2 window and a step size of 2, to reduce the dimensionality of the feature map. After multiple convolution and pooling layers, the feature maps are spread out into one-dimensional vectors and classified or feature extracted by fully connected layers [10, 11].

This paper tries to build a simple convolutional neural network model for music recognition under two emotions and tries to test the accuracy of this model under not much amount of data. The article begins by discussing the acquisition and processing of data, followed by the basic construction of the model and a detailed exposition of the results obtained from model training. Subsequently, the limitations of the model and the data are critically examined, reflecting on their implications and engaging in analysis and discussion. Finally, some reflections on future developments are presented.

2. Data And Method

2.1. Data

The data is from Kaggle, which was scaped using python scripts from FesliyanStudio. The datasets contain 2 classes that are “Happy” and “Sad”. The audio files have been converting into smaller chunks of equal size, that each have a duration of 10 seconds, allowing for easier processing, analysis, or streaming of the audio data. The metadata.txt file contains 2 columns with the information that the files and their corresponding classes. The audio files are in wav format and the sample rate of these wav files is 44100, whose number of channels is 2. The audio files are separated into 3 sets as Test-Sets, Train-Set and Valid-Set.

The article uses the dataloader function in pytorch to load the data, and before loading, the wav files of the dataset are set to the same length of 10s, and the individual wav files that appear to be corrupted are deleted, so as to form a new dataset. Since the wav files used in the article were dual channel, the numpy module was used to merge (average) them into mono channels. The authors divided the data into a training set, a test set, and a valid set according to the previous txt file, yielding a tensor of [16, 352799] for the training set. The wav files were randomly packed in batches of 16 using the dataloader function, and each time a random batch of files from the dataset was input to the model.

2.2. Model

This article uses 2D CNN model with Mel spectrogram inputs. Using a convolution module that consists of 3×3, batch normalization, ReLU non-linearity, and 2×2 max pooling. The module is used in each layer of the 2D CNN. The parameters of the Mel Spectrum are as follows: the number of audio channels is the same as the number of batch input data is 16, the sampling rate is 44100, the short-time Fourier transform is 1024, the minimum frequency is 0.0 Hz, the maximum frequency is 11025.0 Hz, the number of Mel filters is 64, and the STFT window jump length is 256. The parameters of the convolutional layers are as follows: the first layer has a single channel input and 16 channels of output; the second layer is the same as the first; the third layer has 16 channels of input and 32 channels of output; and the fourth layer has 32 channels of input and 64 channels of output. After each convolutional layer, the size of the pooling layer is: (2,2), (3,4), (2,5), (3,3) respectively. The dense layer is composed in the order of linear layer, batch normalization layer, linear layer, discard layer, and activation function layer. The first linear layer has an input and output size of 64, the batch normalization layer has an input size of 64, the second linear layer has an input size of 84, the output size is the same as the classification category of 2, and the discard probability of the discard layer is 0.5.

3. Results and discussion

3.1. Feature Extraction

The model first passes the wav file that has been converted to tensor data in dataloader into the MelSpectrogram module in pytorch, uses the STFT function in pytorch to create a Mel spectrogram from the waveform, and then uses the AmplitudeToDB module to change the spectrogram from the power/amplitude scale to the decibel scale. The Meier spectrogram is shown in Fig. 1.

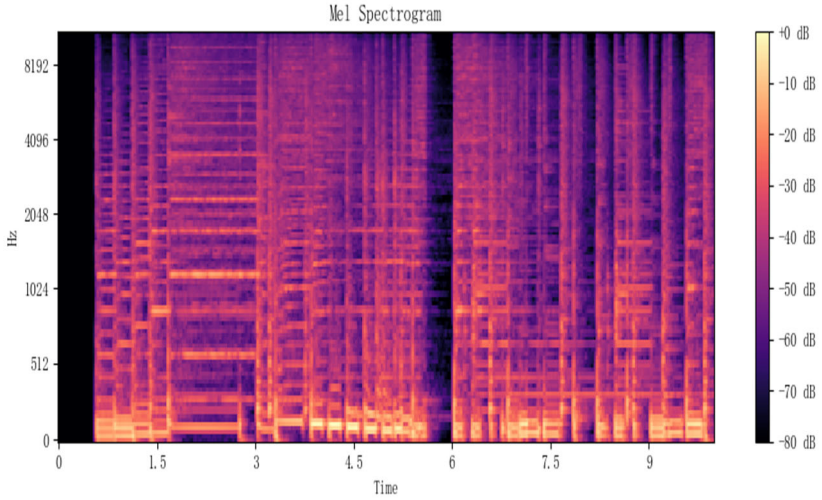

Figure 1. The Meier spectrogram (Photo/Picture credit: Original).

3.2. Feature Extraction

The wav files are divided into two categories, 928 "Happy" files and 1175 "Sad" files. These files were randomly divided into train, test and valid datasets by three txt files. The target folder names and target file names are stored in the three datasets. There are a total of 1600 files in the training set, of which 682 are "Happy" and 918 are "Sad"; the test set has a total of 365 files, of which 169 are "Happy There are 365 files in the test set, of which 169 are "Happy" and 196 are "Sad"; and there are 157 files in the validation set, of which 86 are "Happy" and 71 are "Sad".

3.3. Training

During the training phase, the code checks for CUDA device availability, sets the device to the first available GPU device ('cuda:0') or CPU, then creates the CNN model and moves to the selected device. Define the loss function as 'CrossEntropyLoss'. Use Adam optimiser with a learning rate of 0.001 and create an empty list called "valid_losses" to store the validation losses for each calendar element. Set the total number of training rounds to 30.

Then start training for each epoch. Create an empty list named "losses" to store the training losses for each sequence and put the model in training mode (cnn.train()) at each epoch. Iterate the training data loader and return the audio data and sentiment category indexes. Move audio data and sentiment category indexes to selected cells. Perform model forward propagation to process the audio data with the CNN model to obtain the predictive output. Compute the cross-entropy loss between the predicted output and the sentiment category index. Backward propagation by setting the optimizer gradient to zero (optimizer.zero_grad()), computing the gradient of the loss function value relative to the model parameters (loss.backward()), and calling the optimizer's step() method to update the model parameters Execution. The training loss for each calendar element is added to the loss list and the average training loss is computed and output.

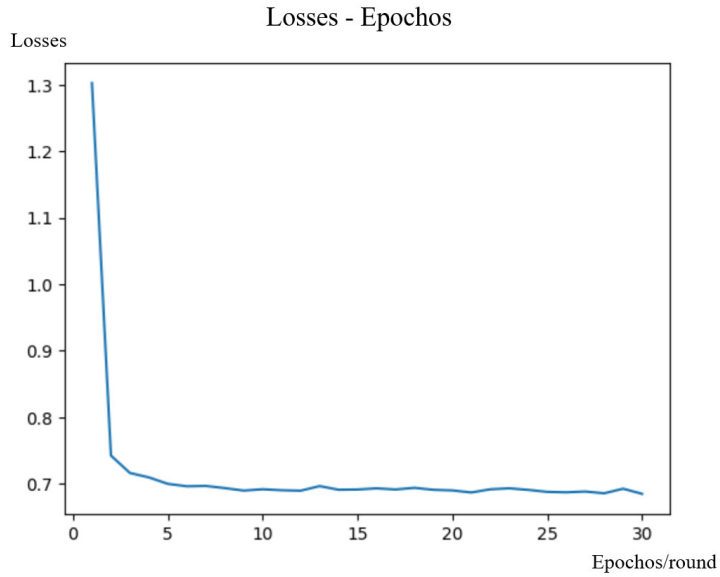

Figure 2. Losses (Photo/Picture credit: Original).

A validation step is performed at the end of each epoch. The model is put into evaluation mode (cnn.eval()). Iterate through the validation data loader and return the audio data and sentiment category indexes. The audio data and sentiment category indexes are transferred to the selected device. Unlike the training phase, the verification phase does not require gradient computation. This can be disabled in the verification loop using the context manager torch.no_grad() to improve efficiency. In the verification loop, the input speech data is first processed through shape transformation and aggregation, and the predicted output of different speech chunks is averaged to obtain the final predicted output. Next, the cross-entropy loss between the predicted output and the adjusted category index is computed and the loss is added to the loss list. The maximum value of the output and the corresponding index are determined using the torch.max() function, and the predicted values are stored in the predictor variables. The indices and predictions for the sentiment categories are converted to a list and added to y_true and y_pred, respectively. The accuracy_score() function is used to compute the forecast accuracy and average the validation losses. Finally, display the validation loss and validation accuracy for each epoch. Finally, save the model parameters to a ckpt file.

After several rounds of training and evaluation, the model performance results show significant variability. The accuracy sometimes reaches 0.78, but the lowest value is 0.28, and the final accuracy is stable around 0.5. During the learning process, the loss function is kept around 0.7. On the other hand, during the learning process, the loss function decreases from an initial value of 1.23 to around 0.7. These results indicate that the model struggles to perform the task and is not very robust. There is still much room for improving the model's performance. To further improve the performance and stability of the model, the strategy can be optimized in terms of model complexity, data expansion, optimizer and learning rate tuning, regularization, increasing the amount of training data, and model selection, in addition to adjusting the hyperparameters. The results of losses and accuracy are shown in Fig. 2 and Fig. 3.

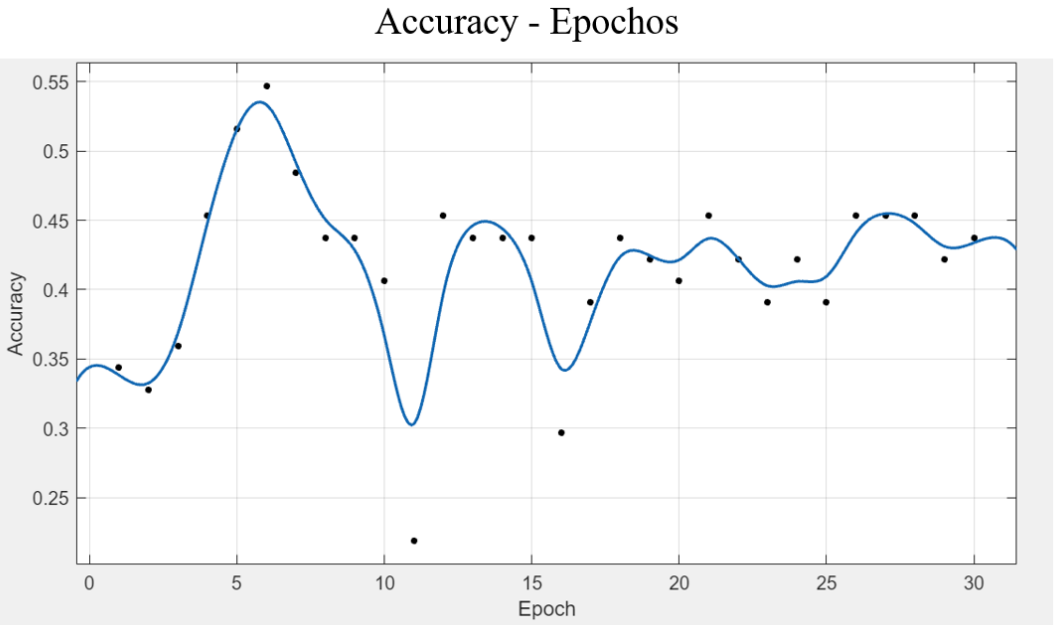

Figure 3. Accuracy (Photo/Picture credit: Original).

4. Limitations and prospects

Nevertheless, this study has some shortcomings and also proposes some of the future prospects. The following limitations exist:

- Dataset dependency: since the experience of emotions is personalized and there is a considerable degree of subjective component in classifying musical emotions, the generalization ability of the model may be somewhat compromised.
- Recalcitrance of tuning hyper-parameters: the model uses fixed hyper-parameter values, such as learning rate, discard rate, etc. Fixed parameters are not conducive to improving the accuracy of the model.
- Singularity of model evaluation: the model used cross-entropy loss and accuracy rate as evaluation indexes, with fewer evaluation parameters, which may not reflect the performance of the model well.

The future outlook is as follows:

- Use of more accurate models: in future work, more advanced and accurate model architectures can be explored and tried to improve the accuracy of speech emotion recognition.
- More rational data labeling: using a multi-label classification approach that allows each audio file to have multiple emotion labels and modifying the model structure and loss function accordingly.
- Interactive tuning and annotation: by providing model-classified audio segments to a human annotator for manual recognition, and then adding the labels of these complete audio segments to the training set to further improve the model's recognition accuracy and generalization ability.
- Diversified model evaluation metrics: F1 scores, confusion matrices, ROC curves, AUC values, and other metrics can be used to examine the model's performance on different sentiment categories to gain a more comprehensive understanding of the model's classification and generalization capabilities.

5. Conclusion

In this work, the authors used a simple two-dimensional convolutional neural network to recognize two emotions in music, happy and sad. They first edited the audio files in the dataset, including length

correction and signal normalization, and converted them into Mel spectral representations. Then they randomly divided the dataset into training, test and validation datasets and fed them into the model for training and testing. However, the resulting model was not very robust and has a number of limitations. These limitations mainly manifest themselves in the following aspects. First, there is a subjective component in the labeling of the dataset, which may have affected the accuracy of the model. Secondly, the hyperparameters are set more robustly and tuned inadequately. This could lead to the model not realizing its full potential. In addition, the estimation model is relatively homogeneous, which may not allow for a full evaluation of the model's performance and generalization ability. In the future, a number of improvements can be made to improve the model's performance. First, more reasonable ways of labeling datasets can be explored, such as performing multi-labeling to improve labeling accuracy. Second, more accurate and sophisticated model structures can be tried, such as deep convolutional neural networks or combining other audio processing techniques to extract richer feature representations. In addition, rational tuning of hyperparameters, e.g., using techniques such as lattice search or Bayesian optimization, can help to find the best combination of hyperparameters. Finally, techniques such as multiple evaluation metrics and cross validation can be used to comprehensively evaluate the model's performance and generalization ability. By building this simple two-dimensional convolutional neural model, the authors have identified its shortcomings in musical emotion recognition tasks and proposed some ideas for future research. This will advance the field of musical emotion recognition and lead to more accurate and reliable methods for model development and evaluation.

References

[1] J. Geipel, J. Koenig, T. K. Hillecke and F. Resch, The Arts in Psych. **77** (2022).

[2] R. K. Nayyar, S. Nair, O. Patil, R. Pawar and A. Lolage, *Content-based auto-tagging of audios using deep learning*, 2017 International Conference on Big Data, IoT and Data Science (BID), Pune, India, (2017), pp. 30-36.

[3] Y. Feng, Y. Zhuang and Y. Pan, *Music information retrieval by detecting mood via computational media aesthetics*, Proceedings - IEEE/WIC International Conference on Web Intelligence, WI (2003), pp. 235-241.

[4] H. I. James, J. J. A. Arnold, J. M. M. Ruban, M. Tamilarasan and R. Saranya, Inter. Res, J. Engi. & Tech., **6**, 56 (2019).

[5] A. Nair, S. Pillai, G. S. Nair and Anjali T, *Emotion Based Music Playlist Recommendation System using Interactive Chatbot*, Proceedings of the 6th International Conference on Communication and Electronics Systems, ICCES (2021), pp. 1767-1772.

[6] Y. Hwang, H. Cho, H. Yang, D. Won, I. Oh & S. Lee, Mel-spectrogram augmentation for sequence-to-sequence voice conversion (arXivLab, 2020), doi: 10.48550/arXiv.2001.01401.

[7] J. Mehta, D. Gandhi, G. Thakur and P. Kanani, *Music Genre Classification using Transfer Learning on log-based MEL Spectrogram*, Proceedings 5th International Conference on Computing Methodologies and Communication, ICCMC (2021) pp 13-72.

[8] J. Lee, J. Park, K. L. Kim and J. Nam, Appl. Sci., **8**, 150 (2018).

[9] D. Han, Y. Kong, J. Han and G. Wang, Front. Comput. Sci., **12** (2022).

[10] S. Tammina, Inter. J. of Sci. & Res Pub., **9** (2019).

[11] M. Won, A. Ferraro, D. Bogdanov and X. Serra, arXiv:2006.00751v1 (2020).