Indonesian Journal of Electrical Engineering and Computer Science
Vol. 25, No. 2, February 2022, pp. 1186~1198
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v25.i2.pp1186-1198

Music genres classification by deep learning

Yifeng Hu, Gabriela Mogos
Department of Computing, School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou, China

Article Info

Article history:

Received Jun 30, 2021
Revised Nov 24, 2021
Accepted Dec 1, 2021

Keywords:

Deep learning
Feature extraction
Music-genre-classification
Machine learning
Neural networks

ABSTRACT

Since musical genre is one of the most common ways used by people for managing digital music databases, music-genre-classification is a crucial task. There are many scenarios for its use, and the main one explored here is eventually being placed on Spotify, or Netease music, as an external component to recommend songs to users. This paper provides various deep neural networks developed based on python, together with the effect of these models on music genres classification. In addition, the paper illustrates the technologies for audio feature extraction in industrial environment by mel frequency cepstral coefficients (MFCC), audio data augmentation in order to analyze the music species based on local data feature comparison.

Corresponding Author:

Gabriela Mogos
Department of Computing, School of Advanced Technology, Xi'an Jiaotong-Liverpool University
111 Ren'ai Road, SIP, Suzhou, China
Email: Gabriela.Mogos@xjtlu.edu.cn

1. INTRODUCTION

In 2019, the total revenue of the music market has come to $20.2 billion, but according to the report showcased, more than 75% of music artists in China still earn less than 1,000 renminbi (rmb) per month. This means that all the online resources (streaming) are occupied by head users. The existing music recommendation function on major music platforms basically determine the users' preferences from historical usage data [1]. This makes the current phenomenon of streaming polarization even more serious, as songs with high streaming are more likely to be heard and easier to collect data from user [2]. On the other hand, songs with high quality but without high streaming are more difficult to be recommended by the system comparing with the Pop music because usage data are less often to be captured [1].

In another respect, the communication of music is influenced by un-quantifiable factors such as geography, politics, the sentiment conveyed, or the artist himself, making the issue of music recommendation harder. For example, gosh music, a Chinese rap group, is better known and distributed in Chongqing than in other regions due to geography. Such an environment is obviously unreasonable from the perspective of cultural communication, and the imbalance of resource allocation caused by excessive concentration of streaming. As a carrier of music playing, the platform should find to the essence of music, rather than under the control of marketing [3]. By analyzing the content of music will be a core idea of this project.

From the level of data, despite of the tremendous scale of data to collect during the categorizing by labeling, the problematic fact is that the songs may not necessarily be mutually exclusive [4] where the manual classification will be meaningless and inefficient. To narrow down the precise research direction, the application scene is focused on the recommendation system of music platform. Propuses a novel activation function, "Swish", was discovered [5]. The function combines the traditional sigmoid function with hyperparameter where specially designed for the rectified linear unit (ReLU) model. The swish replaces the rectified linear unit (ReLU) activation function and outperform ReLU and other activation function. The simplicity of Swish and its similarity to ReLU allows the replacement of ReLU in Swish. Schmidhuber [6] presents a hybrid model blending from pixel-based multilayer perceptron MLP and context-based convolutional neural networks (CNN) and is proven to outperform both two models for very fine spatial resolution (VFSR) image classification in isolation.

Yu *et al*. [7], demonstrate the effect of adjusting different parameters in the output function (relation) of k x $(m – n + 1)/p$. The result of test illustrates high accuracy requires the quantity of output unit below 600 and have no direct relation with the number of filters. In addition, with the optimal architecture, the simulation time increase for a higher number of filters. Hence, the selection of appropriate parameters for different data sets and different needs is also optimized for project performance. In attention mechanism is illustrated and implemented in neural networks [8]. The attention mechanism explains the fact of human tend to selectively concentrate only a part of information and ignore the rest of it. Inspired by this, the attention has been applied in respect of simulation of human visual processing or reading processing. By combining parallelized attention and serial attention with bidirectional recurrent neural networks (BRNN), the accuracy and performance are improved. And the CNN model built based on the parallelized attention also outperform the basic model. The deep CNN were implemented to predict latent factor, which cannot be obtained from user's historical usage data, from music audio [9]. Research concludes that predicting latent factors from music audio is a viable method for recommending new and unpopular music. Moreover, compared with traditional bag-of-words representation, deep learning translates very well to the music recommendation setting in music information retrieval (MIR). Also, Eklund [10] the optimization of the neural network by dropout is demonstrated. The dropout feature has a large impact on training in the early stage that the number of epochs is relatively small. As the number of training increases, the gradient of error with dropout gradually flattens out but maintain the trend of declining. Moreover, compared to the standard neural networks, this result decreases by nearly 1% in the same test environment. Hu *et al.* [11] provides a conclusion that combination of audio and visual feature as a multimodal space which has a better performance after training than audio or visual feature in isolation. In addition, more information is used to construct a more multi-dimensional space for the process, such as audio, text, and images. By means of downscale reduction such as t-distributed stochastic neighborhood embedding TSNE, better classification is capable to achieve. Britz [12] shows that the first collaborative filtering is freshmen friendly in Spotify. The cold start needs to be done by user ratings which takes time. Second, when the user has an unusual taste, the ratings are few and user does not have many logical "neighbors", so recommendations will be poor. Then, as the user keeps on rating, the user's data will not be representative for the new user. As an advantage of collaborative filtering, recommendations are made by neighbors' preferences, avoiding low precision. To sum up, the main disadvantage of the Spotify recommendation system is that it takes time for the system to perceive the users. Therefore, developing of a content-based-music-classification plug-in, or model, has high commercial value to break the existing barriers and reallocate market resources. Moreover, it is necessary to build a better environment of the platform.

Analyzed from the user's point of view, the environment of a good platform in conducive to enhance the user's experience. From the perspective of music artists, it is helpful to perceive user orientation and help musicians to create. Hence, the result of this design is expected to achieve the following functions:

- To be able to recognize audio signals, extracting features.
- To be able to rain the system and update the packages of features according to the database.
- To classify and organize the rules compared to the data set.

2. METHOD

2.1. Data set

- GTZAN: George tzanetakis GTZAN is a database of 1,000 music clips, specially designed for audio and music use. It contains 10 music genres, consisting of blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. These music clips are only 30 seconds long to prevent copyright disputes. The audio, however, is stored in the database in 220-50Hz, mono, 16-bit format.
- Music home-made: Due to our ability of using digital audio workstation DAW, the result of neural networks testing would be interesting with the music home-made working as the test data set.
- Augmented audio file: The wave form of an audio from the GTZAN would be transformed after audio pre-processing. These processed files kept the same quality and other characteristics with the audio file in the GTZAN. Taking these processed files as the data set prevents the overfitting during the training.

2.1.1. Audio pre-processing phase

The audio files as shown in Figure 1 in the database can be used directly, but after a series of transforming, diversity of data can be increased to expand the data set. There are some steps in data pre-processing: import the libraries, import the dataset, check out the missing values, see the categorical values, splitting the data-set into training and test set, and feature scaling. Simultaneously, the pre-processed audio is used as the training set to prevent the network from overfitting during the training process.

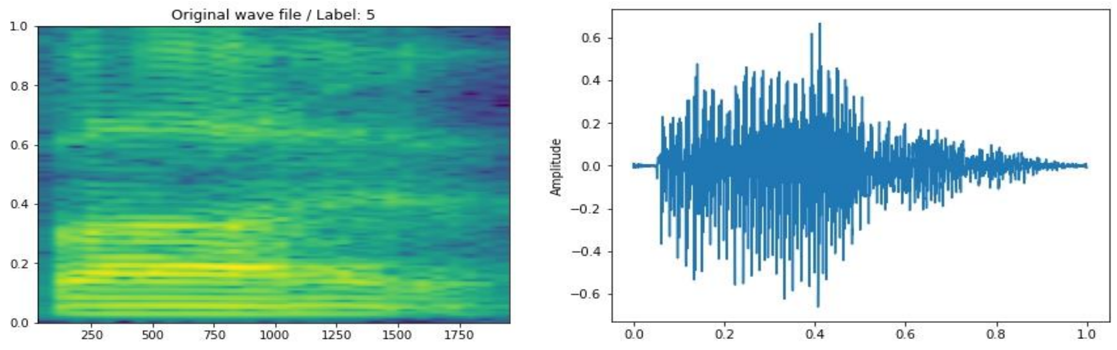


Figure 1. Original wave form

- Data augmentation: The principle of data augmentation is to create a new sample by adjusting the factor in the original sample [13]. This increases the size of the data set during the investigation and avoids the homogenization of the data, which is helpful to avoid phenomena of overfitting in the training. While the conventional data augmentation is acting on several characteristics of sound waves, pitch, loudness, and sound quality. However, these concepts are abstract and based on human perception, not objective features. Therefore, adjusting the corresponding parameters will cause the transformation of a wave form.
- Pitch shifting: For pitch, the corresponding processing is pitch shifting, in the field of music, an octave is divided into 12 semi-tones, which, to put it more plainly, tweaking the parameter of semi-tones determined it sounds higher or lower on the perceptual level. In python, pitch shifting was implemented by calling the function librosa.effect.pitch_shift(signal, sr,n_steps). By adjusting the parameter, n_steps, control the pitch as shown in Figure 2.

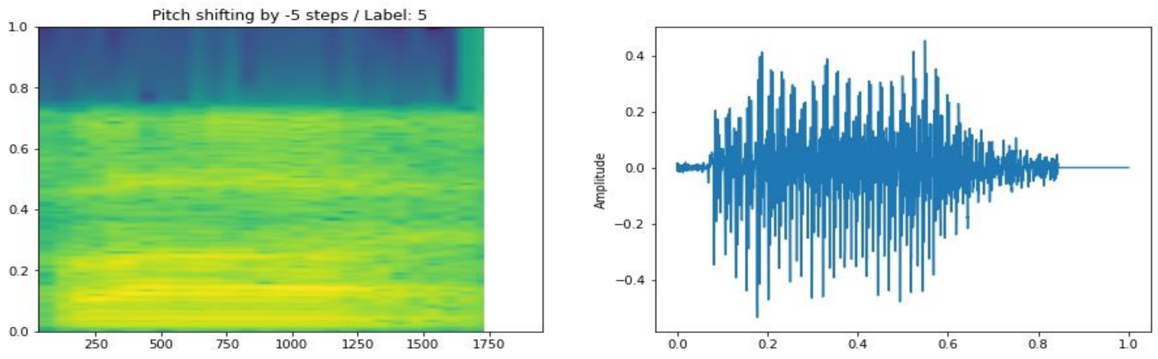


Figure 2. Pitch shifting wave form

- Time stretching: The process of changing the speed/duration of sound without affecting the pitch of the sound as shown in Figure 3. Similarly, in python, the function librosa effects time_stretch (wav, factor) is called to stretch duration of audio file. By tweaking the parameter factor to adjust the specific time duration.
- Noise addition: This process involves addition of noise, i.e., white noise to the sample. White noises are random samples distributed at regular intervals with mean of 0 and standard deviation of 1. The introduction of noise has a great contribution to the robustness of image classification [14]. The function *keras.layers.GuassianNoise(stfdev)(input)* was called to add random Gaussian noise to the input. In place of execution of data augmentation during the pre-processing, this function is called while building the model as shown in Figure 4.

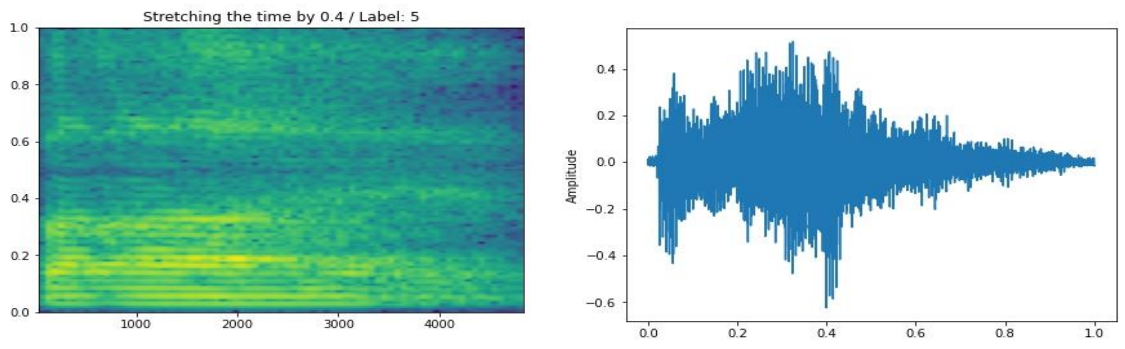


Figure 3. Time stretching wave form

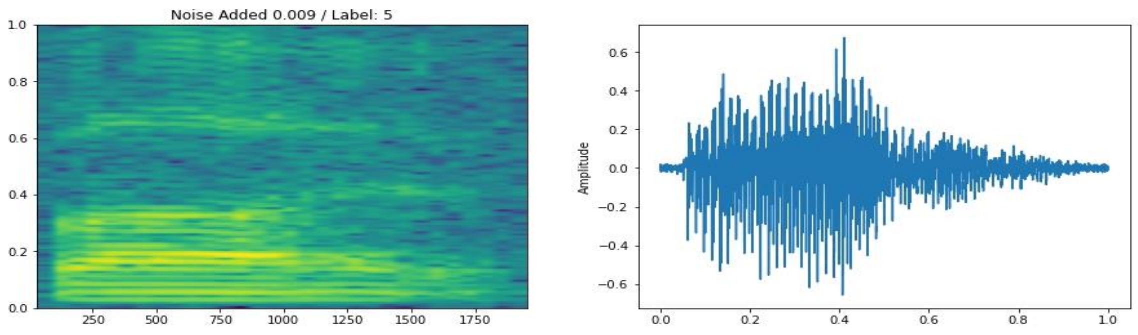


Figure 4. Noise addition wave form

2.1.2. Feature extraction phase

a) Regularization (L1 & L2)

Regularization, a technique commonly used in machine learning to control the complexity of model. The kernel of this algorithm is achieving the confrontation with overfitting by adding penalty in the loss function during machine learning or the optimization in inverse problem. From the DNN level, this algorithm demonstrated excellent performance especially on the data set without label [3], [11]. Common regularization is composed generally by L1 and L2. L2 regularization has the effect of deflating each element of the original optimal solution by a different proportion. L1 regularization produces sparsity by shifting the elements of the original optimal solution by different amounts and making some elements zero.

b) Fast fourier transformation (FFT)

FFT is a classical algorithm for digital signal processing. It is the fast computation of the DFT of a sequence or its inverse transformation, converting the signal from the original domain (time or space) to the frequency domain or vice versa. This algorithm was first proposed in 1965 by J.W. Cooley and T.W. Tukey [15]. The FFT is often utilized in speech recognition, sound simulation, image preservation, filtering, enhancement, and recovery. It is known that sound waves can be decomposed into several combinations of sine waves with different frequency [15]. Therefore, by applying the fast Fourier transform FFT to music fragments, it is possible to analyze the weight of the components at different frequencies, which is an ideal method for content-based music classification.

c) Short time fourier transformation (STFT)

FFT is the conversion of the audio signal from the time domain, as well as loudness, to the frequency domain and Amplitude, so the information of time is sacrificed in this transforming. But sound and music are time-based object. A frequency domain signal that lacks a time sequence can only be used to analyze the value at a certain instant. For this purpose, the two-dimensional results [frequency, amplitude] of the FFT algorithm are converted to three-dimensional results [time, frequency, amplitude] in Short-Time-Fourier's-Transformation STFT. It works by sampling instant values of frequency and amplitude at different moment and save current time as value of result in first dimension.

The X axis represent time and Y axis represent the frequency. The amplitude in FFT will be represented by Grey Level instead. Domain of Grey Level is between 0 to 256 with 0 representing black and 256 representing white. The louder (amplitude) the audio, the more grey level converge to 0. Correspondingly, the more grey level converge to 256, the less loudness. The Figure 5 plotted by such a process is known as a sound spectrum. The result is usually frequent symmetric. For the purpose to simplify the subsequent operation, only half of the frequency will be used.

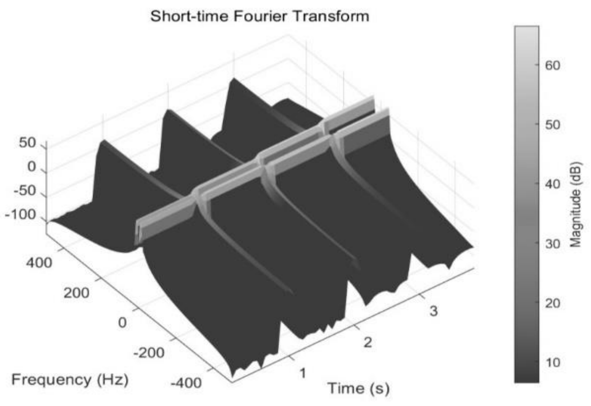


Figure 5. The graphical representation of STFT

d) Mel frequency cepstral coefficients (MFCC)

In speech recognition and speaker recognition, the most used speech feature is mel frequency cepstral coefficients function.

$$Mel(f) = 2595\, x\, lg(1 + \frac{f}{700})$$

According to the research on the hearing mechanism of human ear, it is found that human children have different hearing sensitivity to different frequency sound waves. When two sounds of unequal loudness are applied to the human ear, the higher frequency component will affect the perception of the lower frequency component, making it less detectable, a phenomenon known as masking.

We can learn that the lower frequency is more likely to mask the signal of the higher frequency. Therefore, the frequency band perceived by the human ear is not linearly distributed. Since this nonlinear characteristic does not depend on the nature of the signal, no need for assumptions or restrictions are made on the input signal, and the results of the auditory model are used.

Therefore, MFCC has a remarkable robustness and fits well with the auditory characteristics of the human ear and has an outstanding recognition performance when the signal-to-noise ratio is reduced. The method librosa.feature.mfcc( ) was called in python during the feature extraction stage. To get different precision of audio feature, tweak the parameters in the brackets to determine the length of observation chunk. In this study, the dynamic feature was represented by differential spectrum of static feature. Hence the MFCC feature were stored in local as the result of audio pre-processing.

2.1.3. Training phase

Human interaction-based classification is expensive and difficult [16]. Since late 90's, neuronal network started playing a significant role in image classification in field of remote sensing [17]. On the next, we will focus on MLP, CNN, LSTM-RNN techniques and the results obtained with them.

a) Multilayer perceptron (MLP)

A multi-layer perceptron (MLP) is a network that maps inputs into outputs by forward feed and backward propagation [14]. The MLP structure is typically composed of interconnected neurons in multiple layers (input, hidden and output layers), with each layer fully connected to the preceding layer and the succeeding layer [15]. Following a nonlinear activation function, the output in format of weighted unit can be distinguished the data are not linear separable [16].

b) Convolutional neural networks (CNN)

A convolutional neural networks (CNN) is a variant of multilayer forward feeding neural networks which specifically designed for large scale image processing or sensory data extracting by considering local and global stationary features [17]. CNN had been implemented to achieve classification which explored complex spatial pattern behind the VFSR image [6]. Mechanism of CNN is like MLP which contains a number of layers where the output of previous layer connected to the subsequent layer by a set of learnable weights and biases. The kernel idea in a CNN contains three major operations: convolution, nonlinearity, and pooling/subsampling [18].

The convolutional layers and pooling layers are alternating in CNN framework, until the high-level features obtained followed by a fully connected classification is performed [17]. Unlike MLP, the full-connected architecture is not adopted in CNN which allows the spatial information to be concerned during the convolutional training [7]. Moreover, the weights of convolutional nodes in same map are shared and each convolutional layer contains several feature maps [6]. This feature of CNN allows networks to learn different features with the parameters traceable. CNN is characterized by its pooling/subsampling layer, which generalizes the convolved features by down-sampling and then decrease the complexity of computation during the training. Assuming a pooling layer, the output can be derived from the previous layer.

c) Long short-term memory-recurrent neural network (LSTM-RNN)

Whether it's audio or music, they are both cross the timeline in nature. Models like MLP and CNN both shift the observation window by the same hop length. After extracting the data features in the form of MFCC, the features of the audio signal are transformed into a three-dimensional array consisting of time, frequency, and amplitude. The MLP and CNN feature extraction is more of a summary of the pattern, not a model with a specific sensitivity to time. Therefore, the recurrent neural networks RNN model introduced here is a sequential model, which is usually applied [19] to machine translation, speech recognition, and generating image description.

The advantage of RNN is obvious. A model that looks back to previous data to learn and predict subsequent data is well suited for analysis and categorization with audio. Likewise, its disadvantage is lethal, because the network is built based on the concept of cell, the longer the input data, the deeper the cell will generate the network which cause a situation of excessive computation. In addition, recurrent neural networks RNN does not have long-term memory that it can only work on short interval data which limit the training cannot adapt data with long interval or the data with long duration [5], [12]. To conquer the long-term problem in recurrent neural networks RNN, there is a variation of RNN named long short-term memory (LSTM) [20]. This model realizes the capability of learning long-term dependencies by implementing and optimizing the model of RNN.

3. RESULTS

The implementation of the models of MLP, CNN, and LSTM-RNN are developed in the python environment. To further explore the models suitable for content-based music recommendation, comparison experiments were conducted with the following five dimensions. The dimensions are segments number, layer number, neuron number, activation function, and epoch number, and the accuracy of the results is compared between the same model and across models by means of control variables. In this research, test size accounted for 25% of the total data set and validation target set 20% of the train set.

3.1. Segments number

3.1.1. Variation interval: [10, 20, 30]

Segment number represents the control of the observation points and the accuracy of the observation window movement at the very beginning of the audio pre-processing and signal feature extraction stage. The higher the segment number, the closer the observation points are to each other, the denser the sampling frequency is, the denser the final data will be. In real life, rapid progress of high-resolution information collection, but the information is cheap unless it is either processed or retrievable [4]. In this experiment, 100 epochs are used, and the internal parameters of the model (e.g., number of layers, and activation function) are not adjusted, but only the segment number is adjusted. And dropout had been adapted in realization to decrease the complexity from O(2^n) [7]. The Table 1 shown the average of the results after 3 training sessions.

Table 1. The result of MLP, CNN, LSTM-RNN training on different segments numbers

	Accuracy	Validation Accuracy	Error	Validation Error
MLP - 10 Segments	0.6964	0.5745	1.1177	1.8091
MLP - 20 Segments	0.8231	0.6762	0.7547	1.3065
MLP - 30 Segments	0.8328	0.7116	0.7185	1.1494
CNN - 10 Segments	0.9187	0.7533	0.2494	0.8416
CNN - 20 Segments	0.8351	0.7727	0.4702	0.7022
CNN - 30 Segments	0.8043	0.7602	0.5641	0.7003
LSTM-RNN - 10 Segments	0.8622	0.6707	0.4375	1.2585
LSTM-RNN - 20 Segments	0.8952	0.7050	0.3289	1.1319
LSTM-RNN - 30 Segments	0.8983	0.7282	0.3149	1.0527

3.1.2. MLP and segment number

In the MLP, accuracy increases with the increase of segments number, and the improvement process from 20 segments to 30 segments tends to level off gradually. Validation accuracy also increases with the increase of segments number, and tends to level off, but it is larger than the slope of validation accuracy and the accuracy, which means that there is still room for improvement. For error, the decreasing trend also tends to level off, and the decreasing trend of validation error is gentler. Figure 6 shows how the MLP model can easily reach the extreme value of the train set with the increase of segments number, but it is not very sensitive to the data that the model has not seen. In addition, the MLP model is relatively simple, the training time is also relatively short.

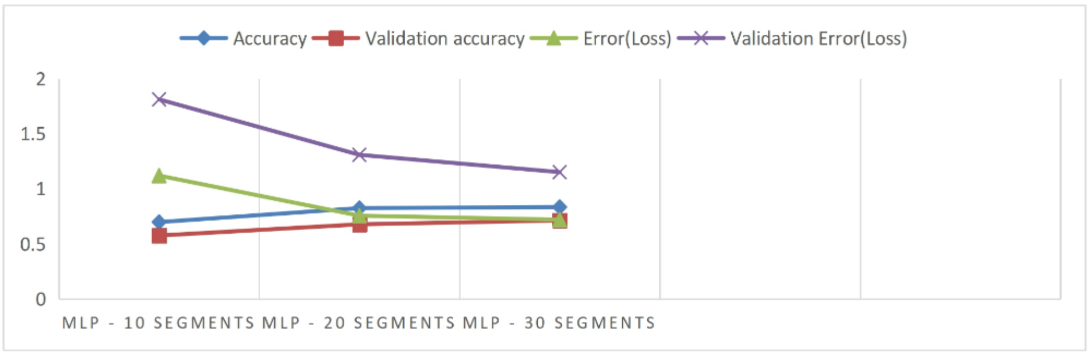


Figure 6. Accuracy, validation accuracy, error, validation error on 10, 20, 30 segment number on MLP

3.1.3. CNN and segment number

Figure 7 shows how in the CNN model, the accuracy in the test set instead decreases with the increase of segments number, and the relationship between validation accuracy and segments number is not obvious, and the fluctuation is small. On the contrary, the error is also the same situation, which decreases with the increase of segment number, and the fluctuation of validation error is not significant. In summary, the change of segment number is only related to the CNN model's familiarity with the train set, and it is not sensitive to the data never seen in the validation set.

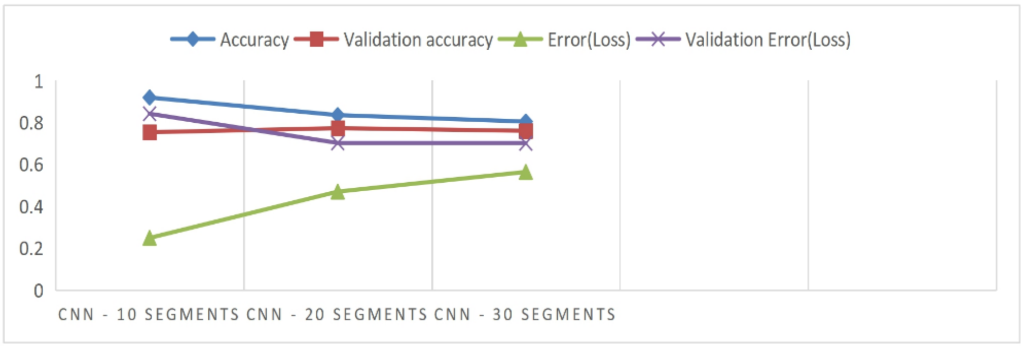


Figure 7. Accuracy, validation accuracy, error, validation error on 10, 20, 30 segment number on CNN

3.1.4. LSTM-RNN and segment number

Figure 8 shows that like CNN, the LSTM-RNN model has harvested a fairly accurate accuracy for the training set when the segment number is 10. However, there is no significant increase in accuracy for the training set as the segment number increases. And the value-added of validation accuracy remains stable. on the contrary, the decrement in whether error or validation error, has remained stable.

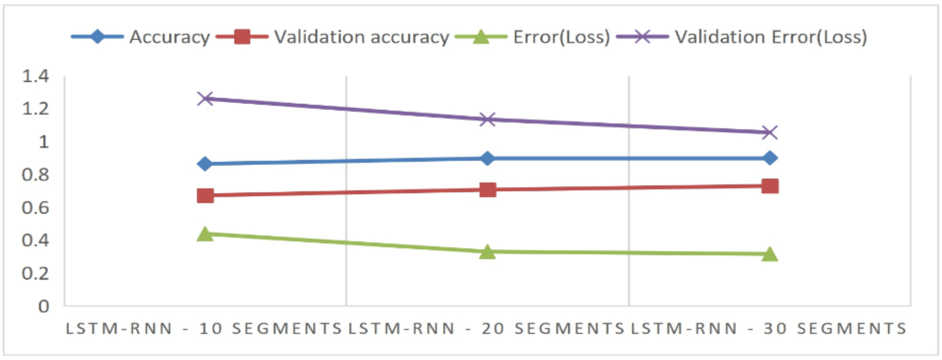


Figure 8. Accuracy, validation accuracy, error, validation error on 10, 20, 30 segment number on LSTM-RN

3.1.5. Cross-sectional comparison with segment number

As can be seen in the Figure 9, the CNN is the best fit for the training set when the segment number is 10, but the highest familiarity with the validation set is when the CNN is at segment number 20. The lowest value occurs in the case of MLP model with segment number 10. By observing the gradient analysis, the MLP model exhibits a higher gradient as the segment number increases, followed by the recurrent neural networks RNN, and only the CNN model decreases as the segment number increases. As for the training time, RNN occupies the longest training time, with the increase of database, the deep network generation is extremely burdensome for computing. the MLP model has a shorter training time because of its relative simplicity [21], while CNN is in the middle of both.

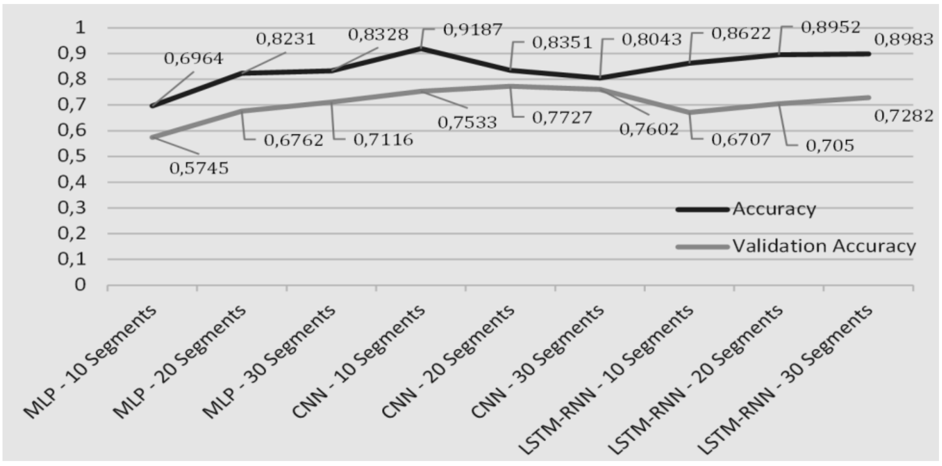


Figure 9. The relationships between Models and segment number

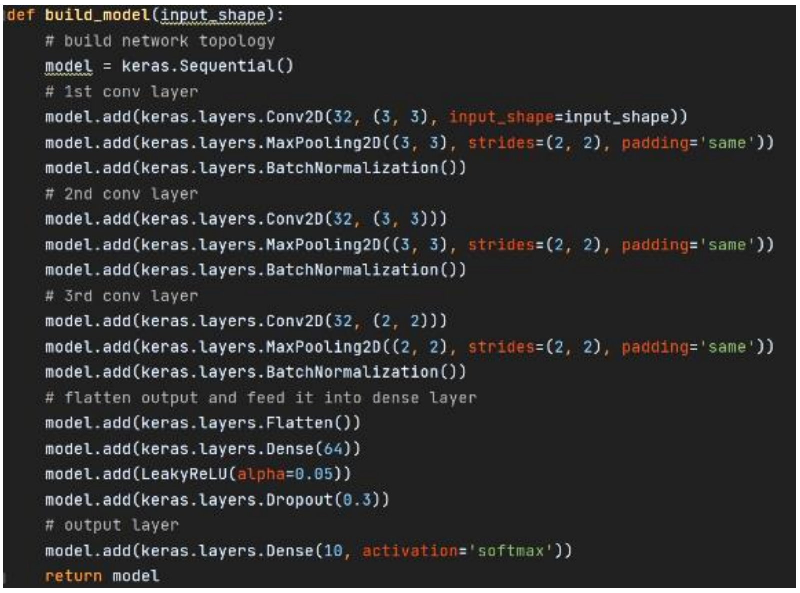
3.2. Activation function

3.2.1. Variation interval: [Sigmoid, tanh, ReLU, Leaky ReLU, ELU]

To find the activation functions most suitable for the model, this part of the experiment focuses on observing the effects of the different activation functions used above the same model on the results. Here are some parameters of the models. Due to the difference between the models, the experiment did not compare between the models when verifying the relationship between the activation function and the models. Leaky ReLU here with the α equals to 0.05.

3.2.2. MLP evaluation

MLP**:** learning rate: 0.0001, hidden layers: [512, 256, 128] , epochs: 50, dropout: 0.3, segment number: 10. In this experiment, we used MLP model in python as shown in Figure 10 and training 10 segments number data with different activation function. The Table 2 shows in the results, that the accuracy of MLP drops from about 50% to 26.8% when using the ReLU function.



```python
def build_model(input_shape):
    # build network topology
    model = keras.Sequential()
    # 1st conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), input_shape=input_shape))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())
    # 2nd conv layer
    model.add(keras.layers.Conv2D(32, (3, 3)))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())
    # 3rd conv layer
    model.add(keras.layers.Conv2D(32, (2, 2)))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())
    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64))
    model.add(LeakyReLU(alpha=0.05))
    model.add(keras.layers.Dropout(0.3))
    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))
    return model
```

Figure 10. MLP model on python

Table 2. MLP training 10 segments number data with different activation function in 50 epochs

Name	Accuracy	Validation accuracy	Error	Validation error
MLP+Sigmoid	0.4923	0.5172	1.7245	1.6842
MLP+tanh	0.4786	0.5042	2.0576	1.9834
MLP+ReLU	0.2632	0.2688	2.5273	2.4743
MLP+Leaky ReLU	0.6513	0.5672	2.3557	2.3557
MLP+ELU	0.5522	0.5512	2.0873	1.9312

Among the MLP models, the overall fluctuations in both the degree of fit to the training set and the Accuracy to the test set are not significant, with the training results of Leaky ReLU reaching the optimum. This fact revealed the downside of taking ReLU as activation function that there a situation so-called “Dead ReLU” which the model cannot be activated once the weight smaller than 0. The slight gradient of the Leaky ReLU model in the negative interval solves this problem.

According to the analysis of the graphs in Figure 11, the stable gradient while the exponential linear unit (ELU) as activation function on MLP after 50 epochs signifies a linear trend during the converge process (Figure 11). Other activation function also performs the linear result same as exponential linear unit (ELU) except the gradient gradually flatten with increment of epochs. The overfitting of MLP connotes an exception of test accuracy above train accuracy. Therefore, seeking for specify reason of this require further investigation.

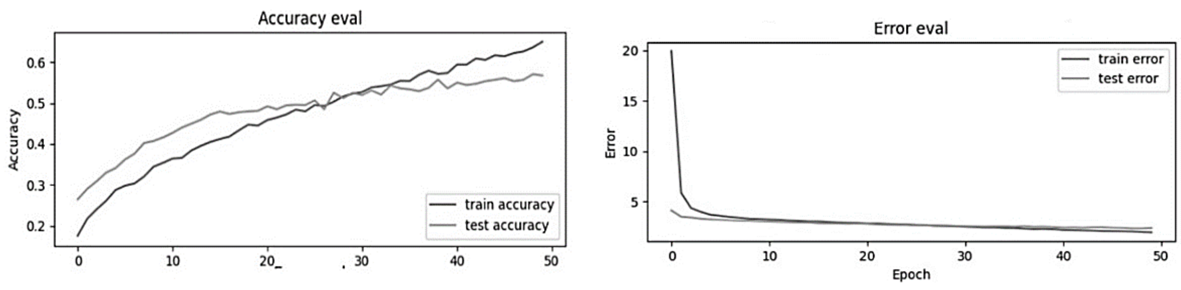


Figure 11. The accuracy and error of MLP model with activation function of Leaky ReLU in 50 epochs

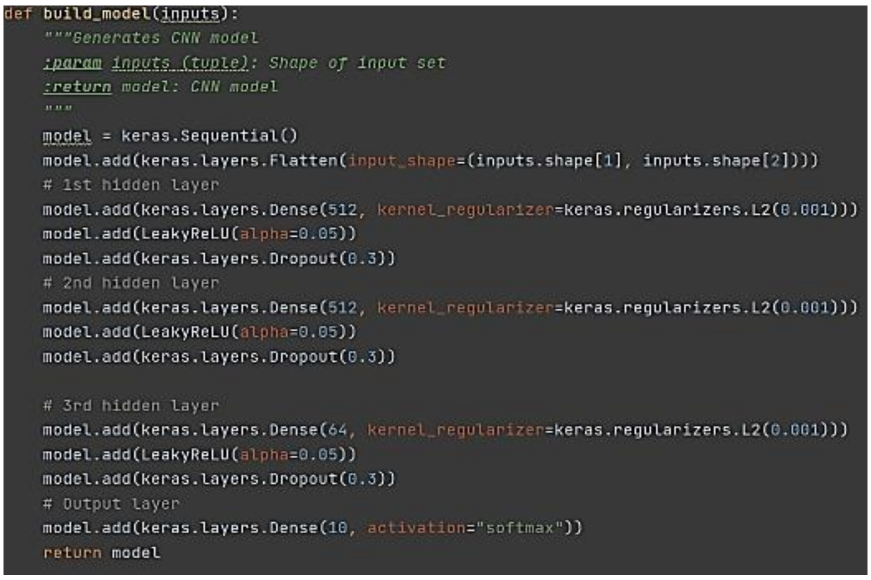
3.2.3. CNNs evaluation

Convolutional neural networks (CNN), learning rate: 0.0001, convolutional layer: [32, 32, 32], full-connected layer [64], epochs: 50, dropout: 0.3, segment number: 10. Due to the advantages of CNN itself in recognizing feature extraction on high-precision images, Table 3 has shown relatively outstanding accuracy on different algorithms [22]. Among them, the activation functions used for the convergence of the training set all played well, and all simulated an accuracy of roughly 82%. For the validation set, i.e., those data not seen by the model, Leaky ReLU shows the best performance [23], with 74% accuracy. Through the evaluation of the image of the training results of the CNN models, CNN demonstrates a stable and efficient performance in the recognition of very fine spatial resolution VFSR image [6] as said by [5]. To implement the CNN model, we used python as shown in Figure 12. Comparing with simple model, MLP, convergence process of CNN presents a non-linear in the Figure 13. This indicates the efficiency of CNN in the early stage with epochs around 10.

The problem between the CNN model and the activation function is also obtained by a simple analysis of the overfitting problem as shown in Figure 14. The activation function usually affects the training process, resulting in higher train accuracy [5], [24]. Among results, tanh has the most severe overfitting in the CNN model.

Table 3. CNN training 10 segments number data with different activation function in 50 epochs

Name	Accuracy	Validation Accuracy	Error	Validation Error
CNN+Sigmoid	0.8406	0.6647	0.4576	1.0268
CNN+tanh	0.8535	0.6120	0.4710	1.2152
CNN+ReLU	0.8244	0.7207	0.4992	0.8509
CNN+Leaky ReLU	0.8569	0.7427	0.4184	0.7589
CNN+ELU	0.8411	0.7167	0.4674	0.8454



```
def build_model(inputs):
    """Generates CNN model
    :param inputs (tuple): Shape of input set
    :return model: CNN model
    """
    model = keras.Sequential()
    model.add(keras.layers.Flatten(input_shape=(inputs.shape[1], inputs.shape[2])))
    # 1st hidden layer
    model.add(keras.layers.Dense(512, kernel_regularizer=keras.regularizers.L2(0.001)))
    model.add(LeakyReLU(alpha=0.05))
    model.add(keras.layers.Dropout(0.3))
    # 2nd hidden layer
    model.add(keras.layers.Dense(512, kernel_regularizer=keras.regularizers.L2(0.001)))
    model.add(LeakyReLU(alpha=0.05))
    model.add(keras.layers.Dropout(0.3))

    # 3rd hidden layer
    model.add(keras.layers.Dense(64, kernel_regularizer=keras.regularizers.L2(0.001)))
    model.add(LeakyReLU(alpha=0.05))
    model.add(keras.layers.Dropout(0.3))
    # Output layer
    model.add(keras.layers.Dense(10, activation="softmax"))
    return model
```

Figure 12. CNN model on python

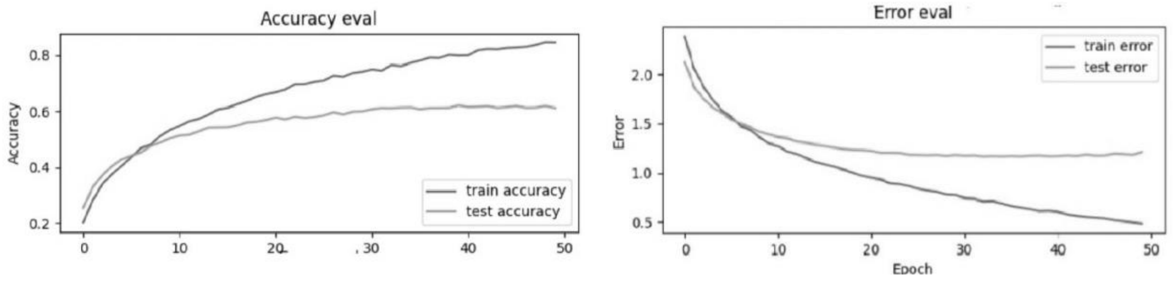
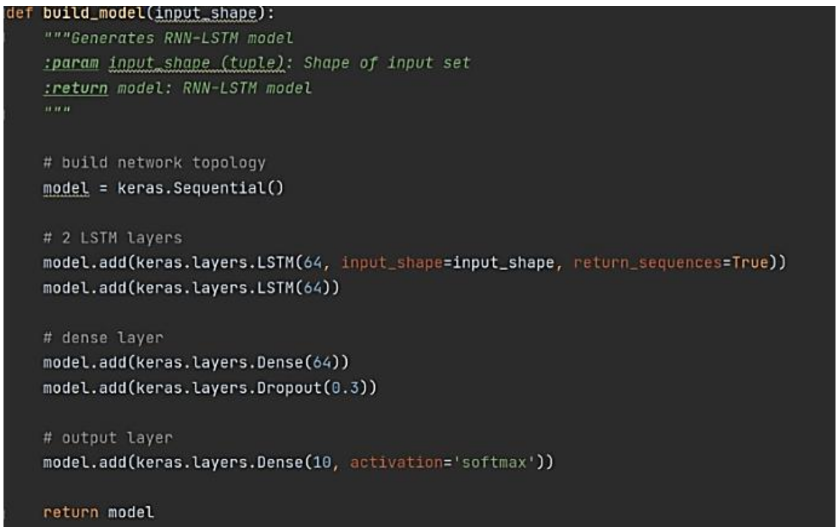


Figure 13. The accuracy and error of CNN model with activation function of tanh in 50 epochs



```python
def build_model(input_shape):
    """Generates RNN-LSTM model
    :param input_shape (tuple): Shape of input set
    :return model: RNN-LSTM model
    """

    # build network topology
    model = keras.Sequential()

    # 2 LSTM layers
    model.add(keras.layers.LSTM(64, input_shape=input_shape, return_sequences=True))
    model.add(keras.layers.LSTM(64))

    # dense layer
    model.add(keras.layers.Dense(64))
    model.add(keras.layers.Dropout(0.3))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))

    return model
```

Figure 14. LSTM-RNN model on python

3.2.4. RNNs evaluation

Recurrent neural networks (RNN) learning rate: 0.0001, LSTM layer: [64], epochs: 50, dropout: 0.3, segment number: 10. Table 4 shows that he RNN is relatively simple in structure, with only one LSTM layer and flatten layer, so it is not difficult to derive from the data in the table that the results are more fluctuated by the activation function. Among them [25], the best performers are the Leaky ReLU and ELU functions, which show a better fit to the training set. However, in contrast, the lowest LSTM-RNN with Sigmoid has a validation accuracy of 60.5%, while the highest LSTM-RNN with Leaky ReLU has only 66.8%, and the difference between the two in accuracy is about 15%. This indicates that to conquer the phenomenon of overfitting, more debugging in either ReLU or a variant function of ReLU are needed. Furthermore, training of LSTM-RNN is relatively too expensive for small scale research. LSTM-RNN average running time of 50 epochs on CPU is 10-15 minutes but CNN takes around 5-10 minute as shown in Figure 15.

Table 4. RNN training 10 segments number data with different activation function in 50 epochs

Name	Accuracy	Validation accuracy	Error	Validation error
RNN + Sigmoid	0.6647	0.6053	0.9856	1.1540
RNN + tanh	0.7839	0.6427	0.6562	1.1148
RNN + ReLU	0.7513	0.6353	0.7868	1.0749
RNN + Leaky ReLU	0.7712	0.6680	0.7400	1.0524
RNN + ELU	0.8020	0.6567	0.6119	1.0962

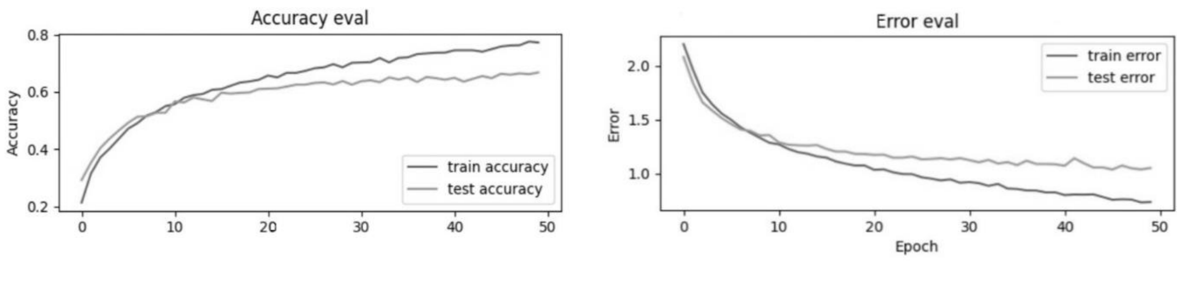


Figure 15. The Accuracy and Error of LSTM-RNN model with activation function of Leaky ReLU in 50 epochs

4. CONCLUSIONS

The goal of this study is to break down the pyramid structure of now day's streaming which helping bottom musician to reap higher revenue. At the very beginning, the expectation is to develop a plugin for stream platform, and such as Spotify. But due to the technical ability limitation, research direction in this project was changed to the analysis and evaluation after basic model developing. The data, GTZAN, used for training also been used in field of audio recognition, and signal feature extracting. As the project was investigated and advanced, “deep learning in audio classification” are redirected to “deep learning in classification of VFSR images” by the powerful function STFT and MFCC in the package librosa. It easily transforms un-analyzable audio signal from time domain and amplitude domain into frequency domain which a new perspective is found for analysis for this purpose. After that, three deep learning models are developed by python and used for image recognition a classification. The paper, besides the implementation of basic function of models, presents the investigation of relation between prediction capability of deep neural network models and factors had roughly done. The experiments are focused on the observation of the relationship between the model and the Segment Number, and between the model and the activation function.

First, the relationship between segment number on different models was explored. It is finally concluded that the higher the precision except for CNN, all demonstrate higher accuracy. Also, the training time needed by the RNN model is increase in due to the effect of its own generated depth by the recurrent structure. Therefore, to make the RNN model harvest efficiency rate during training, from the data accuracy level, the higher the segment number, the higher the accuracy of prediction is accordingly. The same RNN model training time is relatively long and consequently the training cost is expensive, for small-scale surveys and experiments, it is not recommended to use RNN models, especially CPU training. Regarding the activation functions, both Sigmoid and tanh in the saturated activation function have ordinary performance, and only achieve relatively low accuracy and relatively slow convergence on each model. However, due to their relatively stable gradients, better training results may be obtained after increasing the number of epochs.

As for the non-saturated activation functions, ReLU, Leaky ReLU, ELU, all make the training effect of each model particularly obvious in the early stage of the backpropagation process due to the small value of the negative interval of the function and the linear increment in positive interval. However, as the epochs increase, the problem of overfitting also arises. Moreover, the dead ReLU case of the unsaturated activation function ReLU needs to be controlled during the initialization of the weights, otherwise the weight will always be equal to 0 during the backpropagation process, and the neural nodes will not be able to pass any weight. Due to the satisfactory performance of CNN model in this experiment, and strong complementarity between representation of feature by MLP and CNN, and the relatively stable gradient of Sigmoid, as well as the high efficiency of Leaky ReLU function, and the the subsequent work will be based on CNN model, fusion activation function for Sigmoid and Leaky ReLU, and debugging a deep learning model and algorithm more suitable for audio analysis and classification. In addition, the attention model (AM) will be implemented for future work to optimize the model because of its outstanding performance by biological simulating of the activation neurons in human brain.

REFERENCES

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2016.

[2] M. Nielsen, “Neural Networks and Deep Learning,” (2013). Accessed: 4, Apr. 2021 [Online]. Available: http://neuralnetworksanddeeplearning.com

[3] K. T. Al-Sarayreh, K. Meridji, M. Alenezi, M. Zarour, and M. D. Al-Majali, “A sustainable procedural method of software design process improvements,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no 1, 2021, doi: 10.11591/ijeecs.v21.i1.pp440-449.

[4] A. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation,” *Advances in Neural Information Processing Systems*, vol. 26, pp. 2643-2651, 2013.

[5] C. Zhang, H. Li, A. Gardiner, I. Sargent, J. Hare, and P. M. Atkinson, "A Hybrid MLP-CNN Classifier for Very Fine Resolution Remotely Sensed Image Classification,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 133-144, 2018, doi: 10.1016/j.isprsjprs.2017.07.014.

[6] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85-117, 2014, doi: 10.1016/j.neunet.2014.09.003.

[7] Y. Yu, S. Luo, S. Liu, H. Qiao, Y. Liu, and L. Feng, "Deep Attention Based on Music Genre Classification,” *Neurocomputing*, vol. 372, pp. 84-91, 2020, doi: 10.1016/j.neucom.2019.09.054.

[8] S. Oramas, F. Barbieri, O. Nieto, and X. Serra, “Multimodal deep learning for music genre classification,” *Transactions of the International Society for Music Information Retrieval*, pp. 4-21, 2020.

[9] F. Del Frate, F. Pacifici, G. Schiavon, and C. Solimini, "Use of Neural Networks for Automatic Classification from High-Resolution Images," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 4, pp. 800-809, April 2007, doi: 10.1109/TGRS.2007.892009.

[10] V. V. Eklund, “Data Augmentation Techniques for Robust Audio Analysis,” (2019). Accessed: 4, Apr. 2021. [Online]. Available: https://trepo.tuni.fi/bitstream/handle/10024/117251/EklundVille-Veikko.pdf?sequence=2

[11] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 263-272, doi: 10.1109/ICDM.2008.22.

[12] D. Britz, “Recurrent Neural Networks Tutorial, Part 1–Introduction to RNNs,” (2015). Accessed: 4, Apr. 2021. [Online]. Available: http://www.wildml.com/2015/09/recun-ent-neural-networks-tutorial-part-l-introduction-to-rrms/

[13] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised Deep Feature Extraction for Remote Sensing Image Classification," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1349-1362, March 2016, doi: 10.1109/TGRS.2015.2478379.

[14] C. Olah, "Understanding LSTM Networks," (2015). Accessed: 3, Mar. 2021. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[15] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active Learning Methods for Remote Sensing Image Classification," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2218-2232, July 2009, doi: 10.1109/TGRS.2008.2010404.

[16] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and Scalability of GPU-Based Convolutional Neural Networks," *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, pp. 317-324, doi: 10.1109/PDP.2010.43.

[17] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]," in *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13-18, Nov. 2010, doi: 10.1109/MCI.2010.938364.

[18] W. Zhao and S. Du, "Learning multiscale and deep representations for classifying remotely sensed imagery," *Remote Sensing*, no. 113, pp. 155-165, 2016, doi: 10.1016/j.isprsjprs.2016.01.004.

[19] M. Langkvist, A. Kiselev, M. Alirezaie, and A. Loutfi, "Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks," *Remote Sensing*, vol. 8, no. 4, 2019, doi: 10.3390/rs8040329.

[20] M. Kamyab, G. Liu, and M. Adjeisah, "Attention-Based CNN and Bi-LSTM Model Based on TF-IDF and GloVe Word Embedding for Sentiment Analysis," *Applied Sciences*, vol. 11, no. 23, pp. 11255, 2021, doi: 10.3390/app112311255.

[21] P. H. Talaee, "Multilayer perceptron with different training algorithms for streamflow forecastin," *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 695-703, 2014, doi: 10.1007/s00521-012-1287-5.

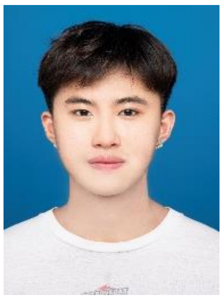
[22] K. Yu, W. Xu, and Y. Gong, "Deep learning with kernel regularization for visual recognition," *Advances in Neural Information Processing Systems*, vol. 21, pp. 1889-1896, 2009.

[23] J. W. Cooley and J. W. Tukey, "An Algorithm for The Machine Calculation of Complex Fourier Series," *Comput.*, vol. 19, no. 2, pp. 297-301, 1965.

[24] X. Li, Z. Deng, Z. Chen, and Q. Fei, "Analysis and Simplification of Three-Dimensional Space Vector PWM for Three-Phase Four-Leg Inverters," in *IEEE Transactions on Industrial Electronics*, vol. 58, no. 2, pp. 450-464, Feb. 2011, doi: 10.1109/TIE.2010.2046610.

[25] M. C. Girish Baabu and M. C. Padma, "Semantic feature extraction method for hyperspectral crop classification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 1, pp. 387-395, 2021, doi: 10.11591/ijeecs.v23.i1.pp387-395.

BIOGRAPHIES OF AUTHORS



Yifeng Hu received his bachelor's degrees in Information and Computer Science. from the University of Liverpool (UK) and Xi'an Jiaotong-Liverpool University (China) in July 2021. His research interests include Music, Machine Learning, Deep Learning and some of the interdisciplinary fields joint together with techniques of computer science. He is also chief musical supervisor, producer, mixing engineer, cofunder of LessThan3 Studio. He can be contacted at email: Hu.Yifeng@hotmail.com

Gabriela Mogos is Associate Professor in the Department of Computing, School of Advanced Technology, at the Xi'an Jiaotong-Liverpool University (XJTLU), Suzhou, China. She received her PhD in Computer Science from the *Alexandru Ioan Cuza* University of Iasi, Romania, and followed this with a postdoctoral research position at the University of Oradea, Romania. Her interests and research activities include Cybersecurity, Machine Learning, Quantum computing with an emphasis on design of new quantum algorithms and quantum cryptographic protocols. She has more than 100 academic publications, books, book chapters, and has acted as principal investigator and co-investigator in international and national research projects. She can be contacted at email: gabi.mogos@gmail.com, Gabriela.Mogos@xjtlu.edu.cn