MLP Coursework 4: Music Genre Recognition Using Triple Generative Adversarial Nets

s1569197, s1679450

Abstract

Recent days, music recommendation system has became important for the online music companies to retain and attract more listeners. In this paper, we present an approach to music genre recognition using Sequence-to-Sequence model and investigate with the state-of-the-art model, Triple GAN(Generative Adversarial Nets), on MagnaTagATune. First, we built a baseline system using Bi-directional LSTM(Long short-term Memory), experimented with different depths and widths of the model. Second, we developed Triple GAN on top of the baseline system, as well as investigate different combinations to the classifier, generator and discriminator. In our result, the baseline model with 3-layer and 100 hidden cell each achieves 85.7% in the test dataset. With 0.7 relatively importance of the generation and classification in Triple GAN, our result is unexpected worse than the baseline model as the best combination in the Triple GAN only achieves 0.183 in hamming loss whereas the baseline model achieves 0.143. We can not simply conclude that Triple GAN is not suitable in music classification and more work is required.

1. Introduction

People love to listen to music and nowadays the easiest access way to music is the digital distribution from online music stores and streaming services, such as Youtube, iTunes, Spotify. Recommending a song to the listeners based on the previous play list is the most important task for those companies to retain and attract more listeners. As a result, music labeling and music classification have been becoming popular in the recent days.

Even though neural network has been developed rapidly and music classification has been making improvement from years to years, there are still difficulties, which can not be resolved totally. The difficulties of this task are caused by the variety of different styles and genres, as well as the social and personal emotions that influences the listener preferences.

Recent researches on music classification have two approaches. One approach uses timbral features by dividing sound signals into frames and computing for each frame and the statistical values (such as the mean and the variance) of those features are calculated. (Li et al., 2003) Another approach uses textural feature by converting the audio signals to the spectrograms and extracts texture features, which are then used for modeling music genres in a classification system, from these time-frequency images by Local Binary Pattern. (Costa et al., 2012)

Both approaches propose the conventional way to address their task using Support Vector Machine(SVM), which classifies groups of data with the help of hyperplanes. The limitation of SVM is that SVM can not resolve a large dataset problem and the outliers in the dataset would greatly affect the performance of the model. Unlike SVM, deep neural network can solve these problems.

The aim of this work is to explore the classification of music genre using adversarial training for a sequential model with continuous data by timbral features approach. Our work consists of two parts, first is to build a baseline system using the architecture of RNN(Recurrent Neural Network), evaluated by varying the hidden cell. Second is to explore the state-of-the-art model, Triple GAN(Generative Adversarial Network), on the dataset. How the use of different combinations in the inner models of Triple GAN affect the performance of the system, the effect of stacked Bi-directional LSTMs are the questions resolved after this work.

The reason of choosing RNN instead of feed-forward network is that RNN has predictive power when used with large amounts of sequenced information and it has been proved by many papers. Furthermore, GAN architecture has few advantages over the others in the following list

- Generates samples faster than fully visible belief networks
- Does not need any Monte Carlo approximations to train in high dimensional spaces
- No requirement that the latent code have any specific dimensionality or that the generator net be invertible comparing the nonlinear ICA

Nevertheless, most researchers have outstanding researches in image classification using GAN, we would like to investigate how the performance of GAN dealing with music data.

In this work, we use MagnaTagATune, which was collected using the TagATune game and music from the Magnatune label. It is not a balanced dataset and more details about the dataset is described in Section 4.

Lastly, we will conclude how the baseline model, which we propose in Section 2, performs in MagnaTagATune dataset, and the performance of the GAN architecture built with the baseline model, mentions in Section 3, after this work.

2. Stacked Recurrent Neural Network

2.1. Recurrent Neural Network

The Recurrent Neural Network is formed by multiple copies of the same network,each passing a message to a successor, this can be seemed as a loop in the network.

2.2. Bi-Directional Recurrent Neural Network

Bi-directional RNNs is a kind of RNN and uses a finite sequence to predict or label each element of the sequence based on the element's past and future contexts. This is done by concatenating the outputs of two RNNs, one processing the sequence from left to right, the other one from right to left. (Schuster & Paliwal, 1997) The combined outputs are the predictions of the teacher-given target signals. Bi-directional RNNs provide higher latency between inputs and corresponding outputs than unidirecionally RNNs and improve classification without incurring much latency.

2.3. Long short-term Memory

LSTM is a deep learning system that avoids the vanishing gradient problem. LSTM is normally augmented by recurrent gates called "forget" gates. LSTM prevents backpropagated errors from vanishing or exploding. Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier. Problem-specific LSTM-like topologies can be evolved. LSTM works even given long delays between significant events and can handle signals that mix low and high frequency components. Here is the compact form of the equations for the forward pass of a LSTM with a forget gate. (Hochreiter & Schmidhuber, 1997)

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denates the Hadamard product. The subscripts $_t$ refer to the time step.

Variables

- $x_t \in R_d$:input vector to the LSTM unit
- $f_t \in R^h$:forget gate's activation vector
- $i_t \in R^h$:input gate's activation vector
- $o_t \in R^h$:output gate's activation vector
- $h_t \in R^h$:output vector of the LSTM unit
- $c_t \in R^h$: $cellstatevector$
- $W \in R^{h*d}$, $U \in R^{h*h}$ and $b \in R^h$:weight matrices and bias vector parameters which need to be learned during training
- σ_g:sigmoid function
- σ_c:hyperbolic tangent function
- σ_h:hyperbolic tangent function

2.4. Stacked Bi-Directional LSTM

As RNN is formed by numbers of neural network combining together, thus we can imagine that stacked RNN is stacking blocks of RNN. The architecture of stacked Bi-directional LSTM is shown in Figure 1.

Deep RNNs have already been proved to be meaningful in researches like(Hermans & Schrauwen, 2013). Our baseline model is implemented as multiple layers depth stacked Bi-directional LSTM as it is inspired by (Sak et al., 2015), which states that stacking RNN would perform better than the shallow models for speech recognition. The basic idea is that deeper RNNs can learn higher level features.

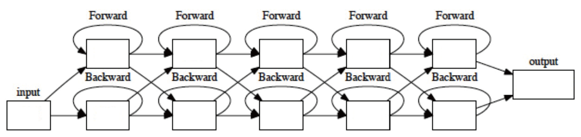

Figure 1. Architecture of stacked Bi-directional LSTM connecting with Forward and Backward Model.

3. Triple Generative Adversarial Nets

3.1. GAN(Generative Adversarial Nets)

GAN consists of two models: A discriminative model, which learns to determine whether a sample is from the model distribution or the data distribution, and A generative model, which produces fake sample and pass the detection from the discriminative model, while the discriminative model is trying to detect the fake sample from the generative model. This competition in this game drives both models to improve until optimal. (Goodfellow et al., 2014) GAN architecture shows in Figure 2. The objective function can be written as

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]$$
$$+\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

where $p_{data}(x)$ is the generator's distribution over dataset x, $p_z(z)$ is the input noise variable, $D(x)$ is the probability that x came from the data rather than $p_data(x)$ and $G(z)$ is a differentiable function.

3.2. Conditional GAN

Extending from the original GAN architecture, adding the condition to the generative model, it is possible to direct the data generation process to what we really want.(Mirza & Osindero, 2014) A sample architecture of conditional GAN shows in Figure 3. According to the idea of the conditional GAN, the objective function can be expressed as

$$\begin{aligned}\min_G \max_D V(D,G) &= \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] \\ &+\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))]\end{aligned}$$

where y is the extra information, such as class labels or data from other modalities.

3.3. Triple GAN

Beside the generative model and discriminative model, there is one more - classier to generate pseudo labels given real data. Adding the classifier in the architecture, this results in a good generator and vice versa in Triple-GAN and the discriminative model can access the label information of the unlabeled data from the classifier and then force the generative model to generate correct data-label pairs.(Li et al., 2017) The new objective function can be expressed as

$$\begin{aligned}\min_{C,G} \max_D V(C,D,G) &= \mathbb{E}_{(x,y)\sim p(x,y)}[\log D(x,y)] \\ &+\alpha\mathbb{E}_{(x,y)\sim p_c(x,y)}[\log(1-D(x,y))] \\ +(1-\alpha)&\mathbb{E}_{(x,y)\sim p_g(x,y)}[\log(1-D(G(y,z),y))] + R_L\end{aligned}$$

$$R_L = \mathbb{E}_{(x,y)\sim p(x,y)}[C(y|x) - y_{real}]^2$$

where $\alpha \in (0,1)$ is a constant that controls the relative importance of generation and classification. R_L is the loss function for the classifier. In the paper, Li suggests to use cross entropy for the classifier. In our work, as the output of the classifier is the timbral features, this is the numerical estimation and the numerical error reduction may not necessarily lead to perceptual improvement, it is good to use mean square error as the loss function. (Yang et al., 2017) The model shows in Figure 4. The development of the triple GAN shows in Algorithm 1.

3.4. Triple GAN - Stacked Bi-Directional LSTM

The combination of RNN and GAN architecture, that is trained with adversarial training to model the whole joint probability of a sequence, and to be able generate sequences of data.

Inspired by (Mogren, 2016), he proposes a GAN with RNN model to generate music data, here we propose our model, Triple GAN with Bi-LSTM. We can simple image that each model in Triple GAN has different depth of Bi-LSTMs to fulfill their own tasks. Mogren has proves that his model can able to generate good music data. In expectation, our model should able to generate correct music label in this manner. The difference between his proposal and ours is that in generator, he proposes to use unidirectional LSTMs and we propose Bi-directional LSTMs in all three models.

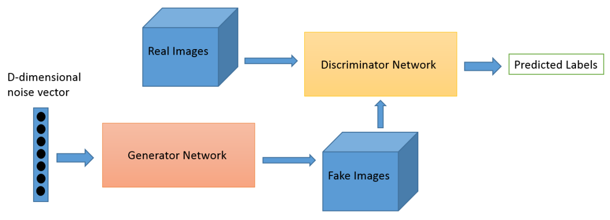

Figure 2. Discriminator acts as a classifier that determinates where a given image looks like a real image from the dataset or like an artificially created image. Generator takes random input values and transforms them into images through a neural network.

Algorithm 1 Minibatch stochastic gradient descent training of Triple-GAN in SSL(Semi-Supervised Learning)

```
for number of training iterations do
    Sample a batch of pairs $(x_g, y_g) \sim p_g(x,y)$ of size $m_g$, a batch of pairs $(x_c, y_c) \sim p_c(x,y)$ of $m_c$ and a batch of labeled data $(x_d, y_d) \sim p(x,y)$ of size $m_d$
    Update D by ascending along its stochastic gradient: $\nabla_{\delta_d}[\frac{1}{m_d}(\sum_{(x_d,y_d)} \log D(x_d,y_d) + \frac{\alpha}{m_c}\sum_{x_c,y_c}\log(1 - D(x_c,y_c)) + \frac{1-\alpha}{m_g}\sum_{x_g,y_g}\log(1-D(x_g,y_g)))]$
    Compute the unbiased estimators $\hat{R}_L$ and $\hat{R}_P$ of $R_L$ and $R_P$ respectively
    Update C by descending along its stochastic gradient: $\nabla_{\delta_c}[\frac{\alpha}{m_c}\sum_{x_c,y_c} p_c(y_c|x_c)\log(1-D(x_c,y_c)) + R_c + \alpha_P R_P]$
    Update G by descending along its stochastic gradient $\nabla_{\delta_g}[\frac{1-\alpha}{m_g}\sum_{x_g,y_g}\log(1-D(x_g,y_g))]$
end for
```

4. Data Processing

4.1. Input - Audio Data Extraction

The dataset, MagnaTagATune, consists of 256863 song clips of 29 seconds each and 188 tags for each song. We have first converted all the songs from MP3 format to WAV format and used HTK(The Hidden Markov Model Toolkit) (Young et al., 2006) to extract MFCCs(Mel-Frequency Cepstral Coecients) acoustic features in this work. As suggested, we have used hamming window with 25ms and 10ms window shift in the configuration file.

MFCCs are designed to capture short-term spectral-based features. After taking the logarithm of the amplitude spectrum based on STFT for each frame, the frequency bins are grouped and smoothed according to Mel-frequency scaling, which is design to agree with perception. MFCCs are

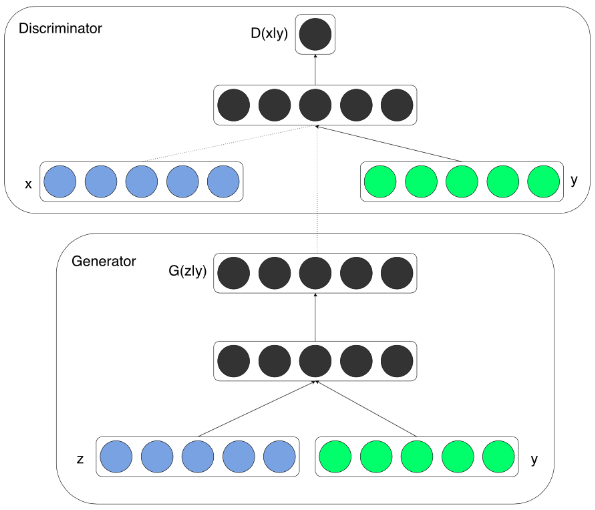

Figure 3. For the generator, 'x' represents the model distribution data that we feed into the model, 'y' is the conditional information that telling the generator what data we want. For the discriminator, 'x' represents the output of generator/ data from the data distribution. 'y' represents the conditional information regarding about the 'x'.

generated by decorrelating the Mel-spectral vectors using discrete cosine transform.

4.2. Output - Label Extraction

The original dataset contains 188 tags, for simplicity, we have picked only top 10 tags based of the frequency of the tags, which contains 16472 songs.

The top 10 tags are 'classical', 'guitar', 'strings', 'drums', 'electronic', 'fast', 'piano', 'slow', 'rock', 'techno'. Their frequency distribution is shown as Figure 5.

4.3. Interpolation

As mentioned in above, the acoustic features are extracted in 25ms hamming windows and 10ms window shift, this

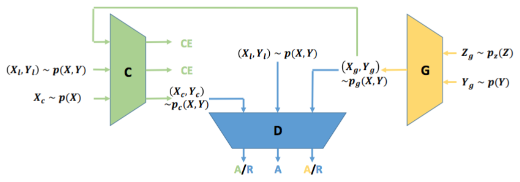

Figure 4. 'C' represents classifier, 'G' represents generator, 'D' represents discriminator with 'R' denoting rejection, 'A' denoting acceptance and 'CE' denoting the cross entropy loss for supervised learning. 'A's and 'R's are the adversarial losses and 'CE's are unbiased regularizations that ensure the consistency between p_g, p_c, and p, which are the distributions defined by the generator, classifier and the true generating process, respectively.

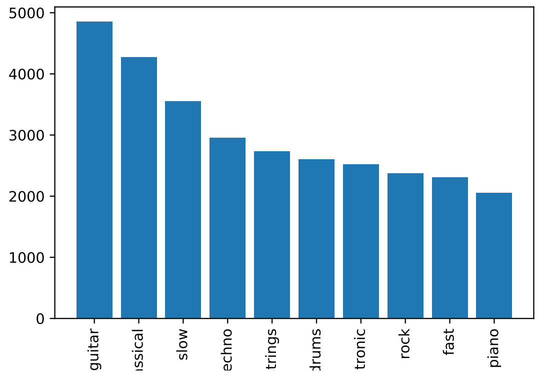

Figure 5. The frequency distribution of top 10 tags.

has resulted in there are 2901 data points with 39 features for each songs. This is too much for the model to learn and map with the labels. Many papers have suggested using K-mean clustering to reduce the number of data points, but there is disadvantage of using it as the resulted data points have been disregarding about the time domain and we think that it is an important matter in speech recognition task.

Spline interpolation (Hermansky & Jain, 1999) is a form of interpolation where the interpolant is a special type of piecewise polynomial and often preferred over polynomial interpolation. Spline interpolation uses low-degree polynomials in each of the intervals, and chooses the polynomial pieces such that they fit smoothly together. Cubic spline interpolation allows to have a smooth curve as the resulted first and second derivatives are continuous.The equation of this cubic spline interpolation is shown below.

$$I_k(V) = I_k + a_k(V - V_k) + b_k(V - V_k)^2 + c_k(V - V_k)^3 \quad (1)$$

where $V = [V_k, V_{k+1}]$ and $k = 0, 1, ..., (n-1)$. To define the coefficients of cubic splines, we need to solve a linear system for concavities $I'' = m_k$ subject to the boundary conditions: $m_0 = m_n = 0$(for natural splines). Provided the solution for m_k is found for $k = 1, 2, ...(n-1)$, the coefficients of the cubic splines are

$$a_k = (I_{k+1} - I_k)/(V_{k+1} - V_k) - (2m_k + m_{k+1})(V_{k+1} - V_k)/6$$

$$b_k = 0.5m_k$$

$$c_k = \frac{1}{6}(m_{k+1} - m_k)/(V_{k+1} - V_k)$$

We have used the implemented 'interpolate' function in Scipy library to fulfill our need. The resulted data has a shape of 30 data points with 39 features for each song. After interpolation, we have applied normalization to the audio acoustic data to make sure that they are in a common scale. We use 'normalize' function in sklearn library in this part.

We can do some visualization like Figure 6 to make sure that we can still keep enough information after dimension reduction for our classification task.

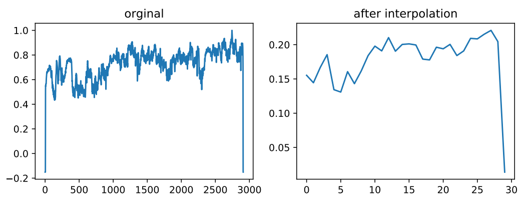

Figure 6. The audio sequence can still keep the shape after down-sampling and is sufficient for classification.

5. Experimental Result

5.1. Experiment set

We split the dataset to training set(56%), validation set(24%), and test set(20%) using 'train_test_split' function in sklearn library with 42 random state.

5.2. Evaluation method

Accuracy In this dataset, we are doing multi-label classification. Different from single-label classification, the accuracy cannot be simply calculated as exact match, because multi-label classification results can be partially correct. For example, assume that the correct label is 0110, if we use exact match, then the output accuracy of 0100 and 0000 will be exactly the same (0%), which is not what we want. (Sorower, 2018)

The equation for accuracy we are going to use is listed as follows. Instead of doing exact match, we are evaluating the accuracy base on the fraction of true label it predicted.

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

Hamming loss Hamming Loss reports how many times on average, the relevance of an example to a class label is incorrectly predicted.(Schapire & Singer, 2000).

$$HL = \frac{1}{kn}\sum_{i=1}^{n}\sum_{l=1}^{k} I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i))$$

5.3. Baseline Model - Stacked LSTM

In this section, we are going to evaluate how different number of hidden units and different number of LSTM layers can affect the loss and accuracy on both training set and validation set.

Model structure The baseline model we are using is a simple sequential model, with multiple bi-directional LSTM layer and a output layer. Considering the domain, our model is solving a multi-label classification task, the output activation we choose is sigmoid function. We choose sigmoid because we don't want the possibility summing up to 0, each class should have its own independent possibility to be positive. The loss function we are minimizing is binary cross-entropy. For optimizer, we choose Adam, a widely used stochastic optimization algorithm.

Regularization In order to reduce the influence of over-fitting, we are applying some regularization techniques.

- Dropout(Srivastava et al., 2014). Dropout is simple way to reduce over-fitting by randomly ignore some neuron's output. We applied Dropout after LSTM layer with dropout rate 0.05.
- Recurrent Dropout suggested in (Lee et al., 2016). Instead of only masking output, this method suggest that using same mask in input, output, and each RNN layer.

Hidden unit number As Figure 7 shows, the curve with 100 hidden units can achieve best validation accuracy. In training set, we can see that as the hidden unit number increases, the training set accuracy also increases. The figure reflected the trend that the accuracy increased linearly and the loss decreases linearly as epoch grows in training set. The more hidden unit number a model contains, the better training set result it will achieve. But in validation set, the accuracy start to decreases after reached a peak. After about 150-200 epoch, the accuracy of training set starts to decrease, and the loss starts to increase. This is the signal of over-fitting. So we can conclude that higher hidden unit number can have greater risk for suffering from over-fitting. In order to achieve a better classification result, we need to find a balance point from under-fitting and over-fitting. We used early-stopping to eliminate over-fitting. The best hidden unit number observed from the figure is 100 hidden unit per layer.

According to test set result shown in Table 1, the model with unit number 100 generalized best, which is consistent with our observation in Figure 7.

Unit number	Accuracy	hamming loss
100	0.8514	0.1486
150	0.8483	0.1517
200	0.8491	0.1509
250	0.8468	0.1531
300	0.8482	0.1517

Table 1. Accuracy and loss of different models with different unit number on test set.

LSTM layer number On top of the result of hidden unit number, we evaluated how different layer number can influence the classification outcome. As we mentioned in Section 2.4, the experiment result in Figure 8 produced similar result from previous work: adding more layers can place some positive influence on classification. The figure reflected that more LSTM layers results in higher classification accuracy and lower loss in training set. But we

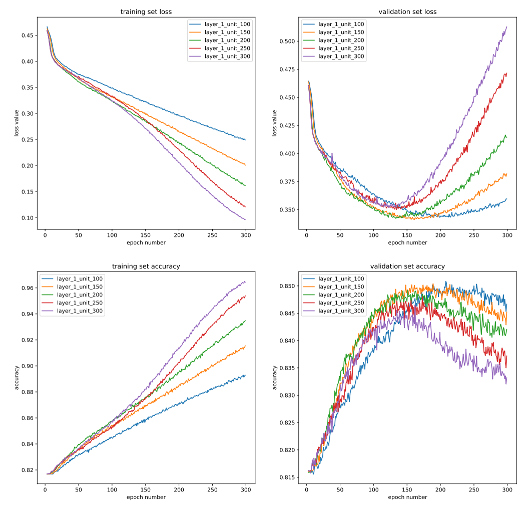

Figure 7. Comparison between 1 layer models with different number of hidden units.

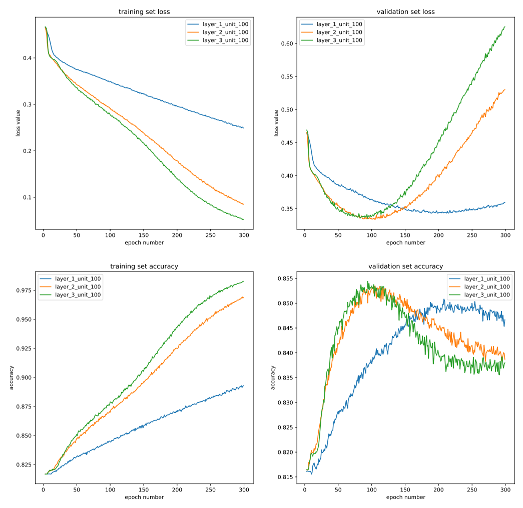

Figure 8. Comparison between models with 100 hidden unit but different number of layers.

can still observe over-fitting phenomenon in the validation curve. After all, 2 layer LSTM can achieve best accuracy and the over-fitting problem is not as severe as the one with 3 layers. In order to minimize the influence of over-fitting, we applied early-stopping.

As Table 2 indicated, the model with 3 layer can achieve best classification result on test set.

Layer number	Accuracy	hamming loss
1	0.8514	0.1486
2	0.8562	0.1438
3	0.8567	0.1433

Table 2. Accuracy and loss of different models with different layer number on test set.

5.4. Triple GAN

In this part of the experiment, we have three models, generator, discriminator and classifier, each model has different purpose in the architecture. Here we are going to investigate how each model in the architecture affects the performance of the system, in other words, we would like to try different levels of depth combinations in LSTMs of each model. Moreover, we would also investigate whether the width of the models leads the performance of the prediction.

Model Structure The setting of each layer of LSTMs in each model is same in the baseline model. Each layer is set with Bi-directional, dropout rate 0.05 and recurrent dropout 0.35. Moreover, the α value, which is the learning ratio between the generator and classifier, is 0.7, and the α_p is 0.01. The reason of setting the α is that we think that generating correct label is more important than generating better music acoustic features and we do not want the classifier to be constrained in the overall architecture.

LSTM layer number in each model Referring to Table 3, we can easily notice that adding more layers in the classifier, this may add more constraints to the overall architecture. The results for 2-Layer LSTMs in the classifier shows that those models have performed worse than those models with only 1-Layer LSTM in the classifier. We can also notice that, with more layers in the generator, the performance increases slightly. Lastly, it seems that adding more layers in discriminator, this does not affect the performance much. Considering the time limit in our work, based on the performance of this experiment set we have chosen a few models to further investigate with 3-Layer Bi-LSTMs. Table 4 with multiple runs of the unexpected results from 3-Layer LSTMs experiment, the results show that the performance of the models decreases largely.

The possible reason of this is that we does not set a proper ratio between the generator and classifier and it is hard to get a optimal point for these three models in the architecture. Moreover, with increasing the layers in each model, the models may require more epochs to get to the converge point. (Pascanu et al., 2012) shows the difficulty in training recurrent neural network and (Arjovsky & Bottou, 2017) tells us the instability of normal GAN architecture, we believe these difficulties would increase in our work as the model we proposed is the combination of these two.

Hidden unit number Observing Table 5, we can easily notice that higher hidden unit number in each layer in each model can have worse than lower hidden unit number. The result shows that this is contradiction about the previous hypothesis stating higher hidden unit ables to achieve better in classification.

Overall, the result from Triple GAN architecture does not outperform the baseline model and this is unexpected as (Li et al., 2017) has shown improvement in the image classification for multiple well-known datasets. One possible reason for this result is that the dataset we chosen for this task may be not suitable for Triple GAN. Although Triple GAN have good performance in image classification task, music genre classification task is quite different from it.

C / (G,D)	1	2
(1,1)	0.184	0.354
(1,2)	0.239	0.241
(2,1)	0.183	0.184
(2,2)	0.183	0.215

Table 3. C:classifier, G:generator, D:discriminator. Loss of different models with different layer number on test set.

(G,D,C)	hamming loss
(3,1,1)	0.298 ±0.1
(3,2,1)	0.408 ±0.13
(3,3,1)	0.451 ±0.11
(3,3,3)	0.352 ±0.2

Table 4. C:classifier, G:generator, D:discriminator.Loss of different models with 3 layer depth on test set.

Hidden Unit / (G,D,C)	100	200	300
(1,1,1)	0.184	0.195	0.284
(2,1,1)	0.183	0.183	0.230
(2,1,2)	0.184	0.249	0.316

Table 5. C:classifier, G:generator, D:discriminator. ixLoss of different models with different hidden number on test set.

6. Conclusions

In this work, we addressed the classification task of music genre using Stacked Bi-LSTMs and Triple GAN - Stacked Bi-LSTMs. Through using HTK toolkit, we extract the acoustic features from the audio files in MagnaTagATune data set and apply some data pre-processing techniques to the extracted features as the input of the model. For the output of the label, we use the top 10 music labels from the dataset. We have investigated the problem of this work through two aspects:1) explore the baseline model with Bi-LSTMs varying the number of layers and the hidden unit, and 2) examine the different combinations of different Bi-LSTMs layers in each model in Triple GAN architecture and the hidden unit in each layer.

With the objective evaluation mentioned in our paper, we have figured out that widening the number of hidden cells in LSTM allows to faster converge and we have proved that stacking RNN shows better performance in test set than single layer. Comparing with (Bergstra et al., 2010), they achieved 84.1% on the top tags and based on the current result of our baseline model, this has showed that RNN with simple setting has preformed better on sequential data than linear classifier. However, with investigations in Triple GAN architecture, the results are unexpected worse than the baseline model we develop in the paper. With the conclusion, Triple GAN increases the difficulty in finding the optimal point for the three models and the ratio of relative importance of the generation and classification is quite important in the architecture as it would directly affect the convergence while optimizing the models. Lastly, the investigation of our work is not successful in term of the result achievement and we can not simply say that Triple GAN is not suitable in music classification as this architecture is still new and more work is needed to put in.

7. Limitation and Future work

We have only used MFCCs in our work, and there are still many feature extraction and engineering techniques can be adopted. (Psutka et al., 2001) show that Perceptual Linear Prediction(PLP) have outperform MFCCs in automatic speech recognition, therefore, experimenting with different type of acoustic features as input data, or even combination of them, would give us different performance results and may lead to any improvement in prediction.

For optimizing the neural network, we have implemented multi-layers, adding bias, widening the hidden units etc, but not initializing different weights. We can use Xavier's initialization in (Glorot & Bengio, 2010) and this has been proved improving accuracy in many papers.

References

Arjovsky, M. and Bottou, L. Towards Principled Methods for Training Generative Adversarial Networks. *ArXiv e-prints*, January 2017.

Bergstra, James, Mandel, Michael, and Eck, Douglas. Scalable genre and tag prediction with spectral covariance. pp. 507–512, 01 2010.

Costa, Y.M.G., Oliveira, L.S., Koerich, A.L., Gouyon, F., and Martins, J.G. Music genre classification using lbp textural features. *Signal Processing*, 92(11):2723 – 2737, 2012. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2012.04.023. URL http://www.sciencedirect.com/science/article/pii/S0165168412001478.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In Teh, Yee Whye and Titterington, Mike (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

Hermans, Michiel and Schrauwen, Benjamin. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 190–198, 2013. URL http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.

Hermansky, Hynek and Jain, Pratibha. Down-sampling speech representation in ASR. In *Sixth European Conference on Speech Communication and Technology, EUROSPEECH 1999, Budapest, Hungary, September 5-9, 1999*, 1999. URL http://www.isca-speech.org/archive/eurospeech_1999/e99_0073.html.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Lee, Daniel D., Sugiyama, Masashi, von Luxburg, Ulrike, Guyon, Isabelle, and Garnett, Roman (eds.). *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016. URL http://papers.nips.cc/book/advances-in-neural-information-processing-systems-29-2016.

Li, C., Xu, K., Zhu, J., and Zhang, B. Triple Generative Adversarial Nets. *ArXiv e-prints*, March 2017.

Li, Tao, Ogihara, Mitsunori, and Li, Qi. A comparative study on content-based music genre classification. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pp. 282–289, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. doi: 10.1145/860435.860487. URL http://doi.acm.org/10.1145/860435.860487.

Mirza, M. and Osindero, S. Conditional Generative Adversarial Nets. *ArXiv e-prints*, November 2014.

Mogren, Olof. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, abs/1611.09904, 2016. URL http://arxiv.org/abs/1611.09904.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012. URL http://arxiv.org/abs/1211.5063.

Psutka, Josef, MÃijller, Ludek, and Psutka, Josef V. Comparison of mfcc and plp parameterizations in the speaker independent continuous speech recognition task. In Dalsgaard, Paul, Lindberg, BÃ˘rge, Benner, Henrik, and Tan, Zheng-Hua (eds.), *INTERSPEECH*, pp. 1813–1816. ISCA, 2001. URL http://dblp.uni-trier.de/db/conf/interspeech/interspeech2001.html#PsutkaMP01.

Sak, H., Senior, A., Rao, K., and Beaufays, F. Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition. *ArXiv e-prints*, July 2015.

Schapire, Robert E. and Singer, Yoram. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000. doi: 10.1023/A:1007649029923. URL https://doi.org/10.1023/A:1007649029923.

Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov 1997. ISSN 1053-587X. doi: 10.1109/78.650093.

Sorower, Mohammad. A literature survey on algorithms for multi-label learning. 02 2018.

Srivastava, Nitish, Hinton, Geoffrey E., Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. URL http://dl.acm.org/citation.cfm?id=2670313.

Yang, S., Xie, L., Chen, X., Lou, X., Zhu, X., Huang, D., and Li, H. Statistical Parametric Speech Synthesis Using Generative Adversarial Networks Under A Multi-task Learning Framework. *ArXiv e-prints*, July 2017.

Young, Steve J., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. *The HTK Book Version 3.4*. Cambridge University Press, 2006.