LEARNING FEATURES OF MUSIC FROM SCRATCH REPRODUCTION

Fatemeh Nejati(30542707) fn1g18@soton.ac.uk
Lin Han(30362377) lh2n18@soton.ac.uk
Jueling Quan(29894351) jq1n18@soton.ac.uk

ABSTRACT

Based on the large-scale dataset, MusicNet, this paper aims to reproduce a multi-label notes prediction task according to the conference paper 'Learning features of music from scratch' published at ICLR in 2017. Music-to-score alignment is applied to derive the pattern of musical scores in recordings which involves different musical instruments and music genre. Multi-layer perceptions, convolutional networks, and linear model are implemented.

1 INTRODUCTION

The original paper introduces a multi-classification problem to predict notes in recordings based on a large-scale music data set, MusicNet. Different approaches are proposed and compared, namely, linear regression for spectrograms and ReLUgrams representation, Multi-layer perceptron (MLP) and Convolution Neural Network (CNN) with different parameters.

MusicNet is an open source data set that records 330 groups of freely-licensed classical music played using various instruments and each group represents a different song. We import this 11GB audio file with '.npz' extension which is mainly used to store data in machine learning scheme as Numpy format. The raw MusicNet database consists of two parts: audio signals and the labels including 'start time', 'end time', 'instrument', 'note', 'start beat', 'end beat', 'note value'. These label classes are defined as a combination of instruments and notes.

The original paper aims to learn features from raw audio data. We want to reproduce these models with different parameters and compare them to the benchmark results mentioned in reference paper. Also, we implement a new feature representation 'Mel' as modification. Finally, we implement our own version of CNN. The codes and data set provided by the paper are based on python2 environment which involves complicated configuration due to the requirement of tensorflow.

2 IMPLEMENTATION

2.1 INPUT REPRESENTATION

For implementing this paper, due to the the specific notes prediction and music feature extraction task, the way we represent input plays a crucial role. Music features as an oscillation of the signal are not directly aligned with traditional machine learning approach and different feature representation will affect the final model performance. Therefore, further feature engineering is meaningful.

2.1.1 AUDIO REPRESENTATION

Music recordings can be represented in different ways. The most common one is the raw audio representation in the time domain. Audio can be interpreted as a function of amplitude against time. It is fairly difficult to extract information form audio to distinguish scores and make predictions of subsequent ones.

2.1.2 SPECTROGRAMS

Spectrogram is an engineered feature representation of the spectrum frequencies of an acoustic signal. In the music information retrieval (MIR) domain, time-frequency representations in the form of

spectrograms appear to have a distinct advantage over the raw audio input [1], because it is easier to distinguish the musical notes which is quite similar to image processing.

(a) fourier transform: The truncated version of original music segment have a linear frequency scale and could be calculated by Fast Fourier Transform (FFT). We use *'librosa'* and *'Scipy.fft()'* in python environment to compute spectrogram with different fourier window sizes.

(b) mel frequency transform: Mel-Frequency analysis of music is based on human perception experiments where human's ear acts as filter and concentrates more on low frequency components. In mel scale, the resolution for higher frequencies and the size of the representation are reduced by adding Mel-Filters. This Mel scale is mentioned in [3] but not implemented in the selected paper, we implement it with Mel window size 1024 and 2048 and compare them with other models.

(c) RELU spectrograms: As is defined in [2], spectrograms can be expressed as follows:

$$Spec_k(x) \equiv \left| \sum_{s=0}^{t-1} e^{\frac{-2\pi iks}{t}} x_s \right| = (\sum_{s=0}^{t-1} cos(\frac{-2\pi iks}{t}) x_s)^2 + (\sum_{s=0}^{t-1} sin(\frac{-2\pi iks}{t}) x_s)^2 \tag{1}$$

where x = (x_1,.....x_t) denotes a segment of an audio signal of length t. Define u_s = cos(2πks/t), v_s = sin(2πks/t), we can deduce a family of features as ReLUgram:

$$f_{k,cos}(x) + f_{k,sin}(x) \equiv \left|u^T x\right| + \left|v^T x\right| = \left| \sum_{s=0}^{t-1} cos(\frac{-2\pi iks}{t}) x_s \right| + \left| \sum_{s=0}^{t-1} sin(\frac{-2\pi iks}{t}) x_s \right| \tag{2}$$

which could be learned precisely by multi-layer perceptrons, a two-layer ReLU network according to [1].

2.1.3 LOGARITHMIC TRANSFORM

This preprocessing step is designed to apply dynamic range compression. Instead of using raw spectrograms, logarithmic frequency scale are more popular among researchers because they reduce the resolution in higher frequencies and the size of the input.The transformation equation is as follows:

$$f_i(x) = log(1 + g(w_i^T x)) \tag{3}$$

where g(x) is mapping function producing different feature representation. We implement 'ReLU($max(0, w_i^T x)$)', 'mel' and 'fft' with different window sizes in logarithmic way as a standard approach to compare model results.

2.1.4 NEURAL NETWORK REPRESENTATION

(a) Multi Layer Perception: We construct a MLP model with a convolutional layer with stride 2. After taking the 'Relu' and 'Log' we do average pooling with size 50 and get the predicted outputs after fully connected layer. We optimize this two-layer network with linear optmiser of L2 norm.

(b) convolutional neural network: We re-implement the CNN architecture introduced in the paper and then modify the structure in our own way as is shown below.

Table 1: Different implementation of CNN

paper:	recep(2048)→	conv(16384,8)→	log(1+Relu)→	a-pool(16,8)→	fc(500)
ours :	recep(2048)→ pool(4)→	conv(16384,8)→ conv2(32*8)→	log(1+Relu)→ pool(4)→	m-pool(16,8)→ fc1,fc2	conv1(32*8)→

Our CNN network architecture is based on [3] using two convolutional layers, 32 filters with size 8, 2 max pooling layers with size 4 and two fully connected layers. We train this model with gradient descent optimizer with 1000 iterations (less than the original paper (100000) due to the long training time). Furthermore, we explored different parameters and evaluation methods to compare with the benchmark version.

2.2 PREDICTION

Prediction of nodes can be interpreted as a multi-variant linear regression problem. The musical scores are expressed by assigning binary labels to a 128-dimension vector as frequency codes for nodes. If a note is presented at the midpoint of a segment, the corresponding element within the vector is 1. With this audio feature mapping, we train a regression model by optimizing square loss and then, predict a sampled subset of nodes and evaluate F1 score.

2.3 REPRODUCTION DETAILS

2.3.1 COMPLEXITY

The audio data set with '.npz' extension is 11GB, which is extremely time consuming to train a model with, especially, for a CNN model. The configuration of computers in ECS building 16 labs is 'i7-7700 CPU, 3.6GHz,RAM16GB'. It took about 40 minutes to finish 100 iterations while the benchmark model requires 100000 iterations for training CNN. When fine tuning our own CNN model, there are even several times that the lab computers crashed when running the model evaluation part. Consequently, we had to choose less number of iterations. As the result, the accuracy of our models is slightly lower. Table 2 shows the time complexity.

Table 2: Time complexity

iter	square_loss	weights_top	weights_bottom	avg_prec	time(s)	eval_time(s)
100	0.806321	0.758286	0.200937	0.524213	4190.703	1619.871
200	0.804295	0.758601	0.201685	0.525786	4240.427	1651.407
300	0.803408	0.758793	0.202085	0.526419	4281.136	1648.493

2.3.2 WINDOW SIZE

Window size in music field is equivalent to receptive filed in computer vision, which is defined as the width of the set of weights in the first layer. There is a trade-off between the capability of capturing relevant signals and lost of temporal resolution for choosing the window size.

2.3.3 REGULARIZATION

Although the author mentions in the paper that using heavy L2-norm regularisation reduces the accuracy, we kept it in our implementation; because we observed that the results do not show any considerable differences.

2.3.4 RESULTS

Related codes were uploaded to github folder:'reproducemusicnet'. All the reproduction and modification results are shown in table 3. The precision curve for typical models are shown in Figure 1.

3 CONCLUSION AND DISCUSSION

The overall reproduction results are close to those of the benchmark, but, with lower average precision (F1-score). This can be put down to the less number of iterations. Actually, we can apply transferred learning method to optimise the model based on the previous weights. By doing this, as we tried, the accuracy will become higher. Regarding the spectrograms, our results seem better, perhaps, it is because we use different packages for fft and getting different samples. In terms of log-spectrograms with bigger window sizes (8192,16384), recall precision is higher. For Mel related work, there is no benchmark, but we can observe that for the same window size, mel feature does not outperform spectrograms but log-mel improves the performance significantly, as high as log-RELUgrams. Additionally, larger window size not always improves the results, which explains the time-frequency tradeoff. Moreover, MLP and CNN models show higher accuracy which means they can extract features well, even for using raw audio data, directly. By building deeper CNNs, we

Table 3: Reproduce Evaluation of different Representation

Representation	Window Size	Precision	Precision (rep.)	Recall	Recall (rep.)	AP	AP (rep.)
log-spectrograms	1024	49.0%	46.9%	40.5%	39.9%	39.8%	37.7%
spectrograms	2048	28.9%	64.2%	52.5%	26.3%	32.9%	42.2%
mel-spectrograms	2048	–	81.1%	–	6.6%	–	37.5%
mel-spectrograms	1024	–	66.1%	–	15.7%	–	34.2%
log-mel-spectrograms	2048	–	60.2%	–	33.3%	–	46.2%
log-spectrograms	2048	61.9%	60.3%	42.0%	31.7%	48.8%	42.4%
log-ReLUgrams	2048	58.9%	58.0%	47.9%	46.7%	49.3%	47.5%
MLP, 500 nodes	2048	50.1%	51.9%	58.0%	61.1%	52.1%	55.6%
MLP, 2500 nodes	2048	53.6%	OOM	62.3%	OOM	56.2%	OOM
AvgPool, 2 stride	2148	53.4%	OOM	62.5%	OOM	56.4%	OOM
log-spectrograms	8192	64.2%	52.2%	28.6%	67.5%	52.1%	55.6%
log-spectrograms	16384	58.4%	46.6%	18.1%	67.4%	45.5%	48.3%
MLP, 500 nodes	16384	54.4%	54.4%	64.8%	60.2%	60.0%	58.8%
CNN, 64 stride	16384	60.5%	52.4%	71.9%	60.7%	67.8%	58.8%

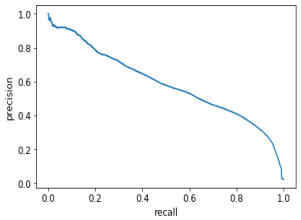

Figure 1: CNN 64 stride

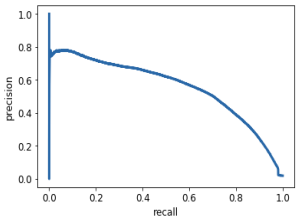

Figure 2: log-spec 8192

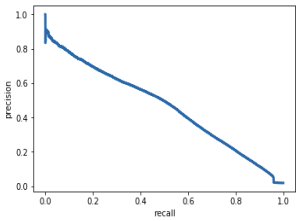

Figure 3: log-mel-spec 2048

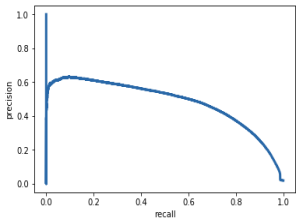

Figure 4: log-spec-16384

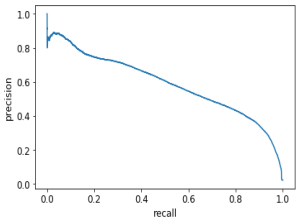

Figure 5: MLP(500)-16384

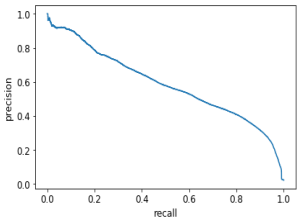

Figure 6: CNN-16384

can expect even better performance. But, there are limitations for the implementation. For instance, limited computing speed without GPU due to the big audio data set.Our modification of CNN with two conv-layers and 64 filters is not really deep. If better devices were available with more sufficient time, it is worth to apply deeper CNNs to spectrograms representation instead of raw audio data to see if the results are improved further. Finally, other machine learning techniques, such as ensemble and clustering, can be applied to CNN for feature extraction. If time permitted, more comments and guidelines should be included for easier understanding. For more complicated CNN architectures in the future, we recommend to use transfer learning, which is more practical to be deployed to other databases and applications.

REFERENCES

[1] Thickstun J, Harchaoui Z, Kakade S. Learning features of music from scratch[J]. arXiv preprint arXiv:1611.09827, 2016.

[2] Mnguez Carretero M. Automatic music transcription using neural networks[J]. 2018.

[3] Dieleman S, Schrauwen B. End-to-end learning for music audio[C]//2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014: 6964-6968.