CS230

Automatic Music Transcription

Ruoyan Chen
Department of Electrical Engineering
Stanford University
ruoyan85@stanford.edu

Yiwen Liu
Department of Electrical Engineering
Stanford University
ywliu24@stanford.edu

Abstract

Automatic Music Transcription (AMT) provides instrumental performers with convenience of performing any compositions they like. It is an interesting but challenging task as it solves the problem of forming a mapping from an audio sequence input to a symbolic representation output. In this paper, we explored several models for AMT problem, including baseline neural network model, Convolutional Recurrent Neural Network and Transformer Model. We experimented with different model architectures and hyperparameters, and the results are analyzed using different metrics.

1 Introduction

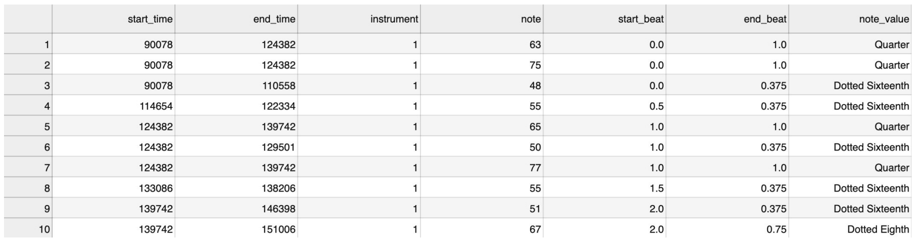
Sheet music is a handwritten or printed form of music notation that uses modern musical symbols. It enables instrumental performers who are able to read music notation or singers to perform a song or piece. For a long time, sheet music has been regarded as one the most effective media for musicians to communicate with each other. It is also an intuitive way for non-professionals to learn how to play a music instrument or sing a song. Nevertheless, sheet music might not be available for all compositions especially for those being protected by strict copyright regulations. Therefore, to better provide music beginners or amateurs with a chance to play these compositions, we came up with a project idea of using deep learning techniques to perform Automatic Music Transcription (AMT). AMT aims to form a mapping from an audio sequence input to a symbolic representation output through supervised learning. The model is trained by raw audios and corresponding music notes as labels.

2 Related work

There exists a multitude of approaches to do AMT. One common technique used in the data preprocessing is down-sampling audio signals to reduce the amount of data and to create its spectrogram, improving the runtime efficiency. In existing researches [1] [2] [3] [4] for the network architectures, with supervised learning based on each music note, the combination of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), as well as some comparisons between Deep Neural Network (DNN) and LongShort-Term Memory (LSTM) are often mentioned. In [5], the paper introduces Convolutional Recurrent Neural Network (CRNN) as well as Connectionist Temporal Classification (CTC) loss function. Inspired by the literature review, we started our project from constructing a baseline model of a three-layer neural network, and then explored more advanced network architectures such as CRNN and Transformer Model.

3 Dataset and Features

For this project, we used raw audios chosen from MusicNet dataset introduced in [6] as data and music notes as labels. MusicNet is a collection of 330 freely-licensed classical music recordings, together with over 1 million annotated labels indicating the precise time of each note in every recording, the instrument that plays each note, and the note's position in the metrical structure of the composition (Figure 1). There are 128 labels for notes and each timestep contains one or more of those labels. The full dataset for this project is divided into two parts: a training set with 320 pieces of raw audio and a validation set with 10 pieces.



	start_time	end_time	instrument	note	start_beat	end_beat	note_value
1	90078	124382	1	63	0.0	1.0	Quarter
2	90078	124382	1	75	0.0	1.0	Quarter
3	90078	110558	1	48	0.0	0.375	Dotted Sixteenth
4	114654	122334	1	55	0.5	0.375	Dotted Sixteenth
5	124382	139742	1	65	1.0	1.0	Quarter
6	124382	129501	1	50	1.0	0.375	Dotted Sixteenth
7	124382	139742	1	77	1.0	1.0	Quarter
8	133086	138206	1	55	1.5	0.375	Dotted Sixteenth
9	139742	146398	1	51	2.0	0.375	Dotted Sixteenth
10	139742	151006	1	67	2.0	0.75	Dotted Eighth

Figure 1: Label example

4 Methods

4.1 Vanilla Neural Network

Similar to [7], to build a baseline model, we took advantage of a three-layer network with a fully-connected layer interposed between the layer-one convolutions and the linear output layer. In this model, we convert the input audio to the spectrogram/frequency domain, thus the intermediate layer captures non-linear relationships between features of this spectrogram. We defined the loss function as the Mean Squared Error and applied Adam optimizer.

4.2 CRNN

In audio problems, it is common to preprocess the raw audio clips into spectrograms, with time as one dimension, and frequency as the other. Here in our problem, we plan to learn features along the frequency axis using 1-D CNNs, and relations between different timesteps using RNNs. This model is called CRNN, and is proposed in [5]. The architecture of the model is shown in Figure 6a.

Convolutional Neural Networks (CNNs) have achieved great results in the computer vision field. It utilizes kernels to detect complex features in data and excels in capturing and transforming high-level information in images. Regarding audio problems, here we treat the sampled frequencies at each timestep as a 1-D array, and use 1-D CNNs to capture the features. Then we can embed the data into a new array with the extracted features. The CNN model we applied to this problem is shown in Table 1.

Recurrent Neural Networks (RNNs) have been widely applied to natural language processing and speech recognition. It has great ability in understanding sequential data by making the hidden state at time t dependent on hidden state at time $t-1$. Therefore, after analyzing the frequency component, RNNs can be used to identify the short term and longer term temporal features in the song. Here we applied Long Short-Time Memory Networks (LSTM), which is shown in Table 2.

After the LSTM, we use two fully-connected layers (Table 4) to generate predicted probability of each note, and use binary cross-entropy loss for optimization. The loss function is:

$$\ell(\hat{y}^{(i)}, y^{(i)}) = -\left(y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right)$$

4.3 Transformer Model

Till now, RNNs are one of the best ways to capture the timely dependencies in sequences. However, the Transformer, which was first proposed in [8], shows great results in translation tasks without any

RNNs. It is an architecture based on attention mechanisms and is used for transforming one sequence into another. The Transformer contains two parts: an encoder and a decoder. Here, on the base of section 4.2, we replace the LSTM with the Transformer encoder to achieve the same functionality [9]. The model architecture is shown is Figure 6b.

The detailed architecture of Transformer encoder is shown in Figure 6c. It consists of two parts: a self-attention layer and a feed-forward neural network. It receives a list of embedded vectors as input, and processes this list by passing these vectors into a the self-attention layer, then into a feed-forward neural network, then sends out the output upwards to the next encoder. The feed-forward network is similar to common fully-connected neural networks, so we will focus on the self-attention layer in the model.

In RNNs, the model maintains a hidden state to incorporate its representation of previous inputs it has processed with the current one it's processing. Self-attention is the method the Transformer uses to look at other positions in the input sequence for clues that can help encode the current input.

Assume our input is a sequence of embeddings X, where each row is an embedding at that timestep. First we create a Query matrix Q, a Key matrix K, and a Value matrix V. These matrix are calculated by multiplying X by three weight matrices that we trained during the training process, like:

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

Then the output of the self-attention layer is calculated by:

$$Z = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

In the equation, $softmax$ can be interpreted as a "score" for values in different timestep. With higher score, the value at that timestep will contribute more to the encoder output at current time.

5 Experiments

5.1 Data Preprocessing

The raw data from MusicNet contains 330 music pieces. We split the data into training set and validation set, each containing 320 and 10 pieces. For data preprocessing, first we down-sample the data from 44.1kHz to 11kHz as introduced in [10] in order to improve the computational efficiency. Then using FFT, we convert the data to complex frequency domain, and split the audios into smaller clips, each contains 64 timesteps. As for labels, we use one-hot encoding to change each label into an array of shape (128,), thus changing the task into a multi-label classification problem.

5.1.1 Vanilla Neural Network

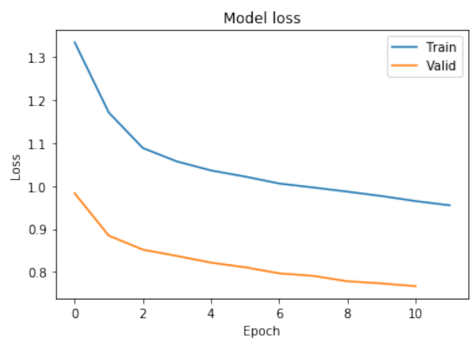
We experimented with two three-layer neural networks, one trained with a fixed log-spaced, cosine-windowed filterbank at layer-one, and the other trained end-to-end from the raw audio, and we see they both have test accuracy above 60% as shown in Figure 2. We also made some observations during the implementation. For example, for the three-layer end-to-end model, because the end-to-end method requires a huge amount of training examples, it resulted in a high variance problem.

5.2 CRNN

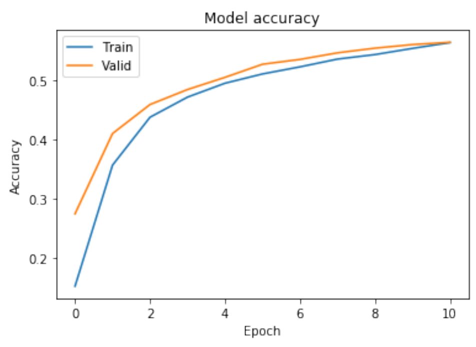
For CRNN, the detailed model architecture is described in Table 1, 2, and 4. We applied cross-entropy loss and Adam optimizer. The loss and accuracy is shown in Figure 3.

5.3 Transformer Model

For Transformer Model, the detailed model architecture is described in Table 1, 3, and 4. We applied cross-entropy loss and Adam optimizer. The loss and accuracy is shown in Figure 6b.

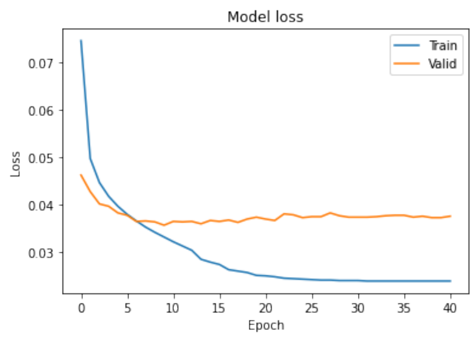


(a)

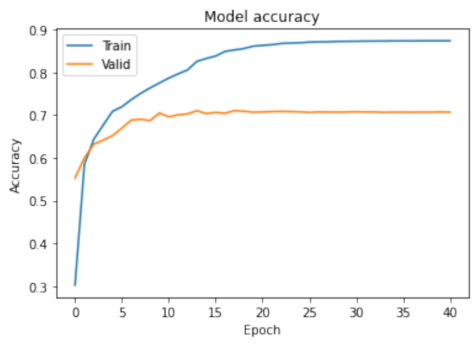


(b)

Figure 2: Three-layer NN: (a) loss; (b) accuracy



(a)



(b)

Figure 3: CRNN: (a) loss; (b) accuracy

5.4 Discussion

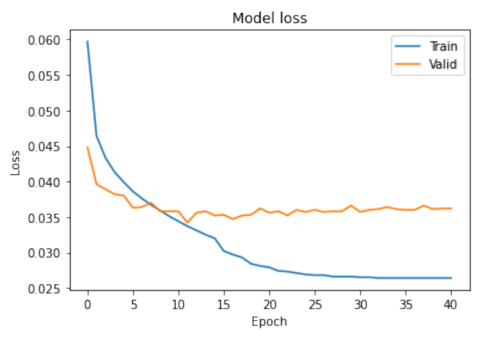
We use ROC curve and precision-recall curve to analyze the results.

The ROC curve (Figure 5a) is a plot false positive rate (fpr) vs. true positive rate (tpr) under different probability thresholds, where:

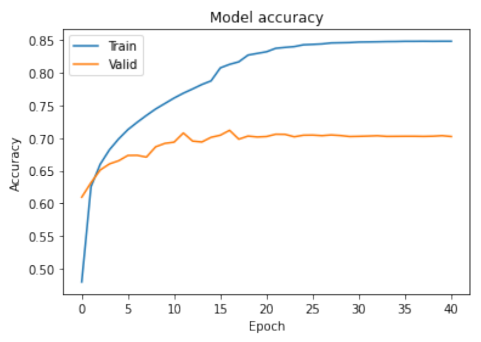
$$fpr = \frac{FP}{FP + TN}$$
$$tpr = \frac{TP}{TP + FN}$$

It shows the model's prediction ability on both balanced and imbalanced binary prediction problems alike because it is not biased to the majority or minority class. A classifier with no discriminative power will form a diagonal line. Here we can see that both our models are far above the diagonal line. What's more, we calculate the ROC-AUC scores for both curves, which is the area under the ROC curve, and they are both above 0.98. This means both our models have great discriminative power regardless of dataset bias.

However, for imbalanced classification with a severe skew and few examples of the minority class, the ROC can be unreliable, for a small number of correct or incorrect predictions can result in a large change in the ROC Curve or ROC AUC score. Therefore, we introduce precision-recall curve as another measure.



(a)



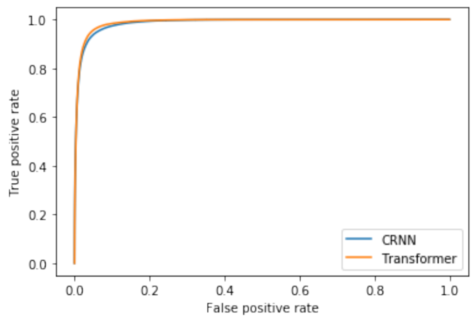
(b)

Figure 4: CNN-Transformer: (a) loss; (b) accuracy

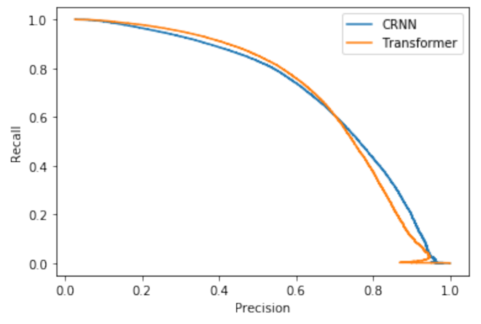
Precision and recall are defined as:

$$precision = \frac{TP}{TP + FP}$$
$$recall = \frac{TP}{TP + FN}$$

The precision-recall curve (Figure 5b) shows the precision-recall pairs for different probability thresholds. It focuses on the model's performance on the minority (positive) class. A model with perfect skill is depicted as a point at (1,1). A no-skill classifier will be a horizontal line at 0. The curves of our models bows towards (1,1), which represents a good classification skill.



(a)



(b)

Figure 5: (a) ROC curve; (b) Precision-recall curve

6 Conclusion

In this project, we explored AMT problem using MusicNet dataset. We experimented with multiple network architectures, including vanilla neural networks, CRNN and Transformer Model. Comparing the performances, both CRNN and Transformer show strong ability in music note recognition and outperform the vanilla neural network, which is consistent with our anticipation and literature reviews.

In the future, we would like to further explore the Transformer Model since we expect it to achieve outstanding performance amongst these network architectures. Other techniques such as multi-head attention may be applied to improve its performance. Besides, this project only focus on note recognition. But to generate a music sheet, we also need to decide the beat and instrument of each note. Therefore, we could further complete the task by including beat calculation and instrument recognition.

7 Contributions

- Topic research and dataset selection - Ruoyan Chen, Yiwen Liu
- Literature review on related works - Ruoyan Chen, Yiwen Liu
- Data preprocessing and Model building - Yiwen Liu
- Baseline model training and Hyperparameters tuning - Ruoyan Chen
- Improved network architectures training - Ruoyan Chen, Yiwen Liu
- Fixing technical issues during the training - Ruoyan Chen, Yiwen Liu
- Project Proposal write up - Ruoyan Chen, Yiwen Liu
- Project Milestone write up - Ruoyan Chen, Yiwen Liu
- Project Final Paper write up - Ruoyan Chen, Yiwen Liu
- Project Video - Ruoyan Chen, Yiwen Liu

The authors are also grateful to our project assistant Jo Chuang for his support on technical problems as well as his valuable insights on architecture development.

References

[1] Yu-Lun Hsu, Chi-Po Lin, Bo-Chen Lin, Hsu-Chan Kuo, Wen-Huang Cheng, and Min-Chun Hu. Deepsheet: A sheet music generator based on deep learning. In *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 285–290, 2017.

[2] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *arXiv preprint arXiv:1710.11153*, 2017.

[3] Jonggwon Park, Kyoyun Choi, Sungwook Jeon, Dokyun Kim, and Jonghun Park. A bi-directional transformer for musical chord recognition. *arXiv preprint arXiv:1907.02698*, 2019.

[4] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[5] Miguel A Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A holistic approach to polyphonic music transcription with neural networks. *arXiv preprint arXiv:1910.12086*, 2019.

[6] John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[7] John Thickstun, Z. Harchaoui, D. Foster, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2241–2245, 2018.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[9] Muqiao Yang, Martin Q Ma, Dongyu Li, Yao-Hung Hubert Tsai, and Ruslan Salakhutdinov. Complex transformer: A framework for modeling complex-valued sequence. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4232–4236. IEEE, 2020.

[10] Julius O. Smith. *Digital Audio Resampling Home Page*. http://www-ccrma.stanford.edu/˜jos/resample/, January 28, 2002.

Appendix: Model Architectures

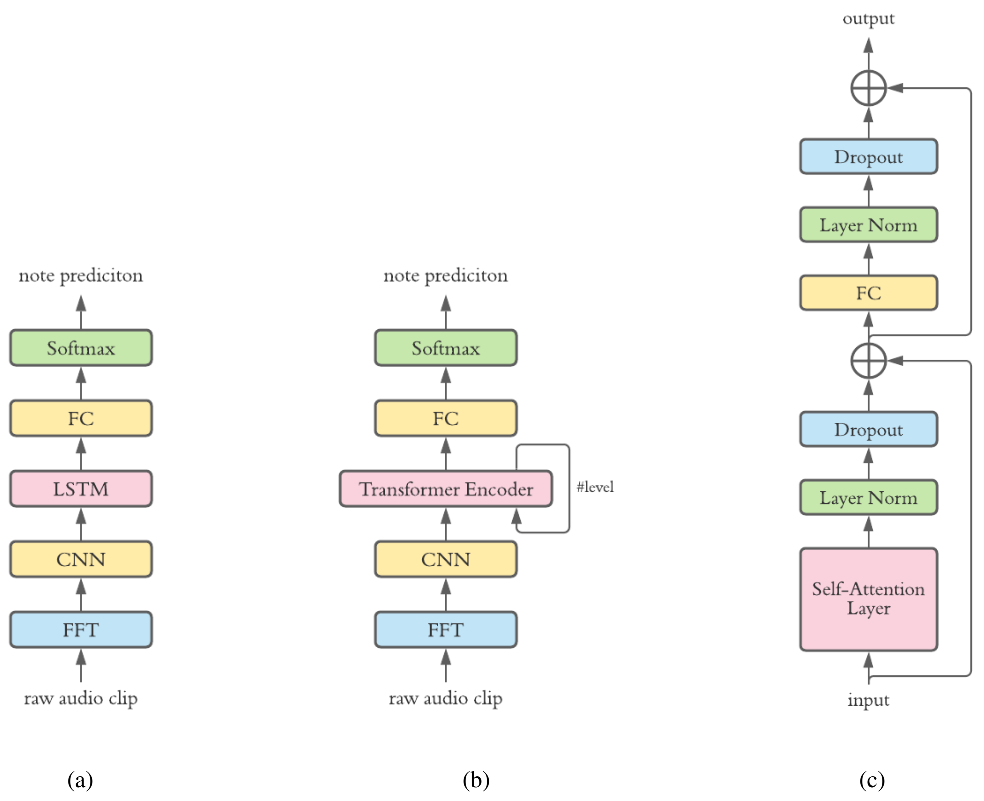


Figure 6: (a) CRNN architecture; (b) CNN-Transformer architecture; (c) Transformer encoder

Table 1: CNN architecture

Layer	Input Shape	Output Shape
Conv1D (6)	$(L \times N, 1, 4096)$	$(L \times N, 16, 4091)$
BatchNorm	$(L \times N, 16, 4091)$	$(L \times N, 16, 4091)$
ReLU	$(L \times N, 16, 4091)$	$(L \times N, 16, 4091)$
MaxPool (2)	$(L \times N, 16, 4091)$	$(L \times N, 16, 2045)$
Conv1D (3)	$(L \times N, 16, 2045)$	$(L \times N, 32, 2043)$
BatchNorm	$(L \times N, 32, 2043)$	$(L \times N, 32, 2043)$
ReLU	$(L \times N, 32, 2043)$	$(L \times N, 32, 2043)$
MaxPool (2)	$(L \times N, 32, 2043)$	$(L \times N, 32, 1021)$
Conv1D (3)	$(L \times N, 32, 1021)$	$(L \times N, 64, 1019)$
BatchNorm	$(L \times N, 64, 1019)$	$(L \times N, 64, 1019)$
ReLU	$(L \times N, 64, 1019)$	$(L \times N, 64, 1019)$
MaxPool (2)	$(L \times N, 64, 1019)$	$(L \times N, 64, 509)$
Conv1D (3)	$(L \times N, 64, 509)$	$(L \times N, 64, 507)$
BatchNorm	$(L \times N, 64, 507)$	$(L \times N, 64, 507)$
ReLU	$(L \times N, 64, 507)$	$(L \times N, 64, 507)$
MaxPool (2)	$(L \times N, 64, 507)$	$(L \times N, 64, 253)$
Conv1D (3)	$(L \times N, 64, 253)$	$(L \times N, 128, 251)$
BatchNorm	$(L \times N, 128, 251)$	$(L \times N, 128, 251)$
ReLU	$(L \times N, 128, 251)$	$(L \times N, 128, 251)$
MaxPool (2)	$(L \times N, 128, 251)$	$(L \times N, 128, 125)$
Reshape	$(L \times N, 128, 125)$	$(L, N, 16000)$
Linear	$(L, N, 16000)$	$(L, N, 320)$

Table 2: LSTM architecture

Layer	Input Shape	Output Shape
LSTM (bi-direction)	$(L, N, 320)$	$(L, N, 256)$

Table 3: Transformer encoder architecture

Layer	Input Shape	Output Shape
Self-attention	$(L, N, 320)$	$(L, N, 320)$
LayerNorm	$(L, N, 320)$	$(L, N, 320)$
Dropout (0.1)	$(L, N, 320)$	$(L, N, 320)$
Residual	$(L, N, 320)$	$(L, N, 320)$
Linear	$(L, N, 320)$	$(L, N, 320)$
ReLU	$(L, N, 320)$	$(L, N, 320)$
Dropout (0.1)	$(L, N, 320)$	$(L, N, 320)$
Linear	$(L, N, 320)$	$(L, N, 320)$
LayerNorm	$(L, N, 320)$	$(L, N, 320)$
Dropout (0.1)	$(L, N, 320)$	$(L, N, 320)$
Residual	$(L, N, 320)$	$(L, N, 320)$

Table 4: Output FC architecture

Layer	Input Shape	Output Shape
Linear	$(L, N, 256/320)$	$(L, N, 2048)$
ReLU	$(L, N, 2048)$	$(L, N, 2048)$
Dropout (0.5)	$(L, N, 2048)$	$(L, N, 2048)$
Linear	$(L, N, 2048)$	$(L, N, 128)$
Softmax	$(L, N, 128)$	$(L, N, 128)$