Term Project Report
Automatic piano transcription based on CNN and RNN combined model

Danye Xu, Zeqi Mao
danye@uchicago.edu, zeqimao@uchicago.edu
Course: TTIC 31110 - Speech Technology

***Abstract*— In this report we explore some potential network configurations for automatic piano music transcription. We train U-Net and gated recurrent unit (GRU) combined models and compare the performance of them with the baseline from the work of Google Brain Team[1].**

I. Introduction

Piano music transcription is a task considered difficult even for humans due to its inherent polyphonic nature. Accurate note identifications are further complicated by the way note energy decays after an onset, so a transcription model needs to adapt to a note with varying amplitude and harmonics. Though the task sounds challenging, it can be helpful in some fields such as music recognition and automatic score generation. The model for piano music may also be extended to many other instruments.

Google Brain Team uses a deep convolutional neural network (CNN) and bidirectional long short-term memory (BiLSTM) combined model to jointly predict onsets and frames of piano music. The prediction is a binary decision task, as the model need to decide whether a frame is the onset of a note, and whether a frame is in the duration of a specific note. For our project, we propose to replace the original model with U-Net and GRU combined model.

U-Net is a kind of convolutional network architecture consisting of a contracting path and an expansive path, introduced in 2015 by Olaf Ronneberger et al[2]. It is originally proposed for segmentation task with very few annotated biomedical images, so it might be useful in accurately recognizing the onsets and frames of different notes with log mel-spectrogram as the input. The model in our project is a simplified version of the original U-Net.

GRU is a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al[3]. The GRU is like a LSTM with a forget gate but has fewer parameters, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM[4]. GRUs have been shown to exhibit even better performance on certain smaller and less frequent datasets[5].

II. Dataset and metrics

We reproduced the model using source code provided at the authors GitHub repository, with slight alteration on how we choose dataset for training, validation and testing.

A. MAPS Dataset

MAPS dataset, which contains audio and corresponding annotations of isolated notes, chords, and complete piano pieces, has a size of 32GB. Due to a lack of computation resources, we select a subset of 716MB and split it into training, validation and testing set of size 7:1:1.

Note that such alteration does not undermine our experiment, for we will be comparing the relative performance of different models.

B. Terms

- Onset: Start of a note
- Offset: End of a note
- Velocity: here we refer 'velocity' as 'the relative note velocity with the maximum velocity in this music piece'

C. Metrics

The metrics used to evaluate a model are frame-level and note-level metrics including precision, recall, and F1 score.

1) Frame-based scores: Frame-based scores are calculated as follows:
   - Precision, recall and F1-score

$$\text{Precision} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t)} \tag{1}$$

$$\text{Recall} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FN(t)} \tag{2}$$

$$F_1 = \frac{2 \times \text{precision } \times \text{recall}}{\text{precision } + \text{recall}} \tag{3}$$

   - Accuracy

$$\text{Accuracy} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t) + FN(t)} \tag{4}$$

     - $TP(t)$: true positives. Calculated for frame t, based on the number F0s that correctly correspond between the ground-truth F0 set and the reported F0 set for that frame.

- $FP(t)$: False positives. Calculated as the number of F0s detected that do not exist in the ground-truth set for that frame.
- $FN(t)$: False negatives. Calculated as the difference between the number of reported negatives at frame t and the number of negatives in the ground-truth at frame t. Therefore, false negatives represent the number of active sources in the ground-truth that are not reported.
- The $TP(t), FP(t)$ and $FN(t)$ are summed across all frames to calculate the total number of TPs, FPs and FNs for a given musical clip.

- Total Error Score E_{tot}

$$E_{tot} = \frac{\sum_{t=1}^{T} \max\left(N_{ref}(t), N_{sys}(t)\right) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \tag{5}$$

- $N_{ref}(t)$: the number of F0s in the ground-truth list for frame t
- $N_{sys}(t)$: the number of reported F0s
- $N_{corr}(t)$: the number of correct F0s for that frame

This total error can be decomposed into the sum of three sub-errors. They are:

- Substitution error

$$E_{sub} = \frac{\sum_{t=1}^{T} \min\left(N_{ref}(t), N_{sys}(t)\right) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \tag{6}$$

- Missed error

$$E_{miss} = \frac{\sum_{t=1}^{T} \max\left(0, N_{ref}(t) - N_{sys}(t)\right)}{\sum_{t=1}^{T} N_{ref}(t)} \tag{7}$$

- False alarm: counts the number of extra F0s returned that are not substitutes

$$E_{fa} = \frac{\sum_{t=1}^{T} \max\left(0, N_{sys}(t) - N_{ref}(t)\right)}{\sum_{t=1}^{T} N_{ref}(t)} \tag{8}$$

2) Note-based metrics: We use the mir_eval[6] library to calculate note-based precision, recall, and F1 scores. We can think of the ground-truth list as a fixed collection of events where each event is defined by three variables, F0, onset and offset. In the first scenario, a returned note event is assumed to be correct if its onset is within a +/-50 millisecond range of a ground-truth onset and its F0 is within +/- a quarter tone (3%) of the ground-truth pitch. Here, the offset times are ignored. In the second scenario, in addition to the previous onset and pitch requirements, the correct returned note is required to have an off-set time within 20% of ground-truth notes duration around the ground-truth notes offset value, or within 50 milliseconds of the ground-truth notes offset, whichever is larger. For these two cases, precision, recall and F_1 are calculated where true positives are defined as the returned notes that conform to the previously mentioned requirements and false positives were defined as the ones that do not.

We also define an additional measure called Overlap Ratio (OR). The OR for a i^{th} correct note in the returned list is defined as

$$\mathrm{OR}_i = \frac{\min\left(t^{ref}_{i,off}, t^{sys}_{i,off}\right) - \max\left(t^{ref}_{i,on}, t^{sys}_{i,on}\right)}{\max\left(t^{ref}_{i,off}, t^{sys}_{i,off}\right) - \min\left(t^{ref}_{i,on}, t^{sys}_{i,on}\right)} \tag{9}$$

where $t^{sys}_{i,off}$ and $t^{sys}_{i,on}$ are the onset and offset times of the correctly returned note. $t^{ref}_{i,off}$ and $t^{ref}_{i,on}$ are the onset and offset times of the ground-truth note.

III. Model Configuration

We will further explore CNN-RNN combined model for our task. The original model in the paper is CNN-BiLSTM. And we decide we shall use 3 different configurations: CNN-GRU, U-Net-BiLSTM and U-Net-GRU.

A. Modified U-Net

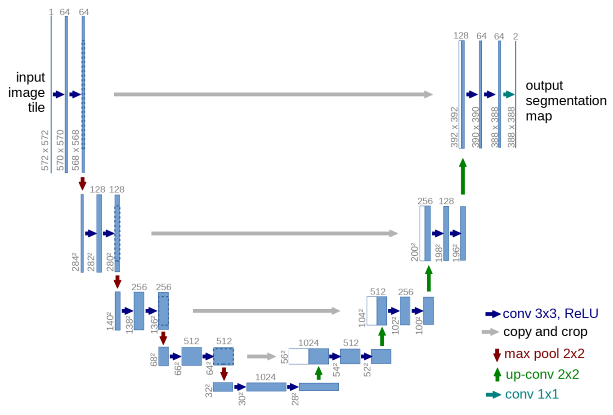


Fig. 1. Architecture of original U-Net

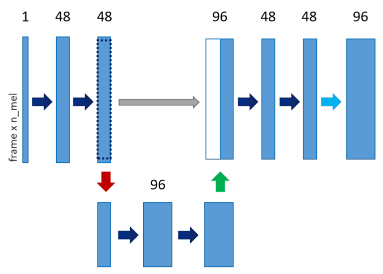


Fig. 2. Architecture of modified U-Net

The network architecture is illustrated in Fig. 2, which is a simplified version of the original U-Net in Fig. 1. As the CNN in the original model only consists of three layers, we have no need to import the whole U-Net architecture in our model. The modified U-Net should be sufficiently complicated and much more memory-saving.

In modified U-Net, the contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions, each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels is doubled. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. At the final layer a 1x1 convolution is used to map each 48-component feature vector to the desired number of channels.

B. GRU

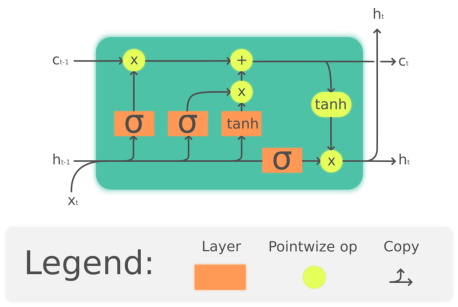


Fig. 3. Architecture of LSTM

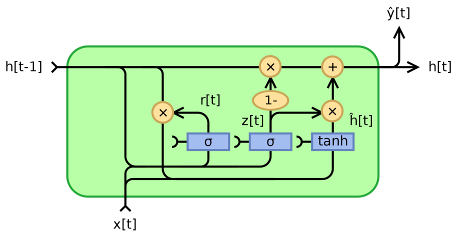


Fig. 4. Architecture of GRU

Initially, for $t = 0$, the output vector is $h_0 = 0$.

- $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$
- $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$
- $h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot tanh(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_h)$

Variables:

- x_t: input vector
- h_t: output vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrics and vector

Comparing the architecture of LSTM in Fig. 3 and GRU in Fig. 4, GRU lacks an output gate. This architecture with less parameters will faster the training procedure and might be more generalizing for a small dataset. The GRU used in our model is also bidirectional.

IV. Experiments

We trained our models and the original models using Pytorch on the MAPS dataset described in Section 2. We compared the performance of 4 different model configuration: CNN+BiLSTM, CNN+GRU, U-Net+BiLSTM, and U-Net+GRU.

A. Hyper-parameter Tuning

As the model CNN+BiLSTM is our baseline, we do not need to do hyper-parameter tuning. We tuned learning rate, number of channels of U-Net, dropout in double-convolution function of U-Net. There are 31 metrics in our final evaluation matrix and we show 5 F_1 score here to illustrate our hyper-parameter tuning process and how we selected the best model. The best model is marked with red bars, with model configuration of U-Net+GRU, hyper-parameters learning rate = 0.0003, U-Net channels = 48 and dropout = 0.2 between 2 convol

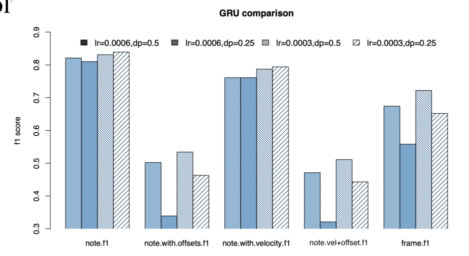


Fig. 5. Result of different CNN+GRU model F1 score. Here *note* means only measure onsets of notes, $note.with.offset$ means measure onsets and offsets of notes together, $note.with.velocity$ means measure onsets and velocity of notes together, $note.vel + offset$ means measure onsets, offsets and velocity all together, i.e. only when the prediction of onset, offset and velocity of a note are right, we confirm this prediction to be correct.

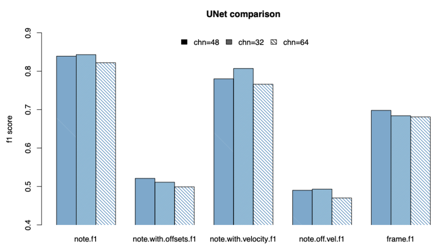


Fig. 6. Result of different U-Net+BiLSTM model F1 score.

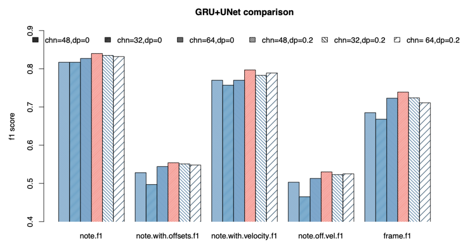


Fig. 7. Result of different U-Net+GRU model F1 score.

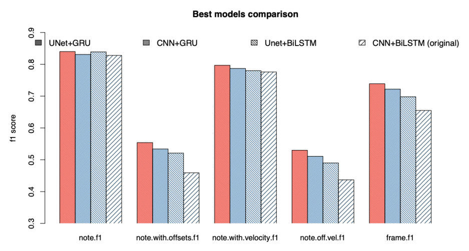


Fig. 8. Best models F1 score comparison.

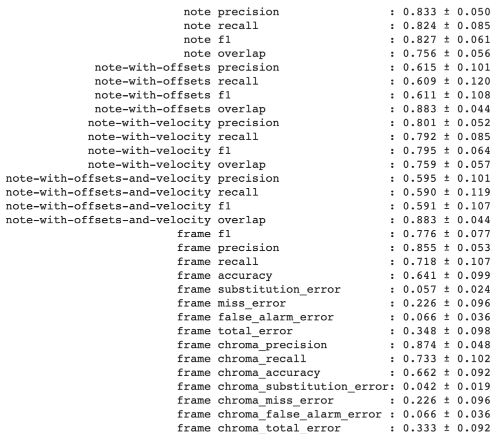
B. Experiment Design

First, we run models (training epochs = 10000) with different hyper-parameters for parameter tuning. After we selected the best model, we re-train the model with training epochs = 100000. Then we evaluate this model on the test set to get the evaluation result. Lastly we send sample piano music slices to the model. The output transcription MIDI files will be generated using the code base provided by the authors of [1].

V. Results

A. The best model

In Fig. 9 we put the full evaluation result of our best model: training epochs = 100000, learning rate = 0.0003, U-Net channels = 48 and dropout = 0.2 between 2 convolution layers. The training time for this model is 11 hours on Google Colab, using GPU acceleration.



```
                         note precision                  : 0.833 ± 0.050
                         note recall                     : 0.824 ± 0.085
                         note f1                         : 0.827 ± 0.061
                         note overlap                    : 0.756 ± 0.056
            note-with-offsets precision                  : 0.615 ± 0.101
            note-with-offsets recall                     : 0.609 ± 0.120
            note-with-offsets f1                         : 0.611 ± 0.108
            note-with-offsets overlap                    : 0.883 ± 0.044
           note-with-velocity precision                  : 0.801 ± 0.052
           note-with-velocity recall                     : 0.792 ± 0.085
           note-with-velocity f1                         : 0.795 ± 0.064
           note-with-velocity overlap                    : 0.759 ± 0.057
note-with-offsets-and-velocity precision                 : 0.595 ± 0.101
note-with-offsets-and-velocity recall                    : 0.590 ± 0.119
note-with-offsets-and-velocity f1                        : 0.591 ± 0.107
note-with-offsets-and-velocity overlap                   : 0.883 ± 0.044
                        frame f1                         : 0.776 ± 0.077
                        frame precision                  : 0.855 ± 0.053
                        frame recall                     : 0.718 ± 0.107
                        frame accuracy                   : 0.641 ± 0.099
                        frame substitution_error         : 0.057 ± 0.024
                        frame miss_error                 : 0.226 ± 0.096
                        frame false_alarm_error          : 0.066 ± 0.036
                        frame total_error                : 0.348 ± 0.098
                        frame chroma_precision           : 0.874 ± 0.048
                        frame chroma_recall              : 0.733 ± 0.102
                        frame chroma_accuracy            : 0.662 ± 0.092
                        frame chroma_substitution_error: 0.042 ± 0.019
                        frame chroma_miss_error          : 0.226 ± 0.096
                        frame chroma_false_alarm_error : 0.066 ± 0.036
                        frame chroma_total_error         : 0.333 ± 0.092
```

Fig. 9. Result of best model full evaluation metrics

B. Sample Output

With the prediction of onsets, frames and velocity of the notes in a piece of piano music, we can get a complete score which can be played by software synthesizers. The detailed theory of velocity can be found in [1]. In Fig. 10 and Fig. 11 we present one original and transcribed sample in our data. We can see that overall the predicted labels resembles original labels in terms of F_0, onsets and relative speed, but with minor differences in the offset of the notes. The predicted labels shows more interruptions within and between ongoing notes. Fig. 12 shows the corresponding transcribed slice.

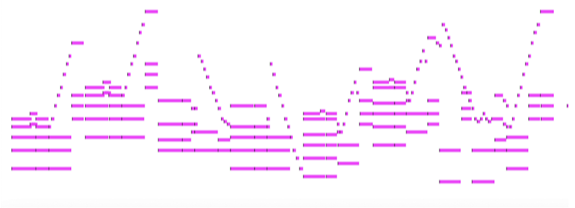


Fig. 10. Original Labels. The black frames are the onsets of notes, the pink frames are the duration of notes.

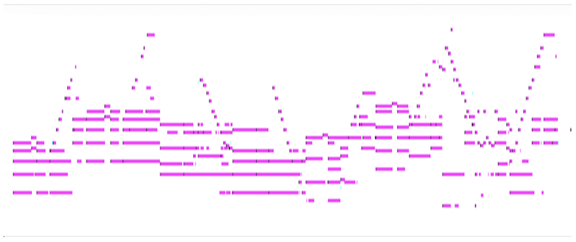


Fig. 11. Predicted labels

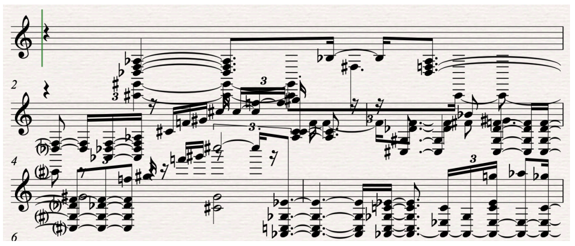


Fig. 12. Transcribed slice

VI. Conclusion & Discussion

A. Conclusion

1) By our experiment, we found that U-Net+GRU model with proper hyper parameters out performed the original CNN+BiLSTM model proposed in [1] on MAPS dataset in terms of our evaluation metrics and especially F_1 scores.
2) We also spotted that by using GRU instead of BiLSTM, we could reach a faster training speed. Table I shows the training time (10000 training epochs) with and without GRU models.

No.	Model configuration & hyper-parameters	Training time (min)
1	CNN+BiLSTM (Original setting)	62
2	CNN+GRU (lr = 0.0003, dropout = 0.5)	35
3	UNet+BiLSTM (lr = 0.0003, dropout = 0.2,Channel = 48)	77
4	UNet+GRU (lr = 0.0003, dropout = 0.2, Channel = 48)	68

TABLE I
Training time comparison

B. Discussion

1) We can see that correctly predicting the offset time is still a difficult job in our model. This is perhaps due to the natural decay of energy when the note goes on. If given more time, we could explore further on that.
2) The transferability of the model across different instruments is worth further exploration. The annotated music pieces of other instruments or different styles of piano music can also be trained in our model to improve the robustness. Whether our model can beat others on larger dataset is also a problem to explore.

References

[1] 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.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[4] Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[6] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.