

AUTOMATIC PIANO TRANSCRIPTION WITH HIERARCHICAL FREQUENCY-TIME TRANSFORMER

jkhkh

Keisuke Toyama¹

Taketo Akama²

Yukara Ikemiya¹

Yuhta Takida¹

Wei-Hsiang Liao¹

Yuki Mitsufuji¹

¹ Sony Group Corporation, Japan

² Sony Computer Science Laboratories, Japan

keisuke.toyama@sony.com

ABSTRACT

Taking long-term spectral and temporal dependencies into account is essential for automatic piano transcription. This is especially helpful when determining the precise onset and offset for each note in the polyphonic piano content. In this case, we may rely on the capability of self-attention mechanism in Transformers to capture these long-term dependencies in the frequency and time axes. In this work, we propose *hFT-Transformer*, which is an automatic music transcription method that uses a two-level hierarchical frequency-time Transformer architecture. The first hierarchy includes a convolutional block in the time axis, a Transformer encoder in the frequency axis, and a Transformer decoder that converts the dimension in the frequency axis. The output is then fed into the second hierarchy which consists of another Transformer encoder in the time axis. We evaluated our method with the widely used MAPS and MAESTRO v3.0.0 datasets, and it demonstrated state-of-the-art performance on all the F1-scores of the metrics among *Frame*, *Note*, *Note with Offset*, and *Note with Offset and Velocity* estimations.

1. INTRODUCTION

Automatic music transcription (AMT) is to convert music signals into symbolic representations such as piano rolls, Musical Instrument Digital Interface (MIDI), and musical scores [1]. AMT is important for music information retrieval (MIR), its result is useful for symbolic music composition, chord progression recognition, score alignment, etc. Following the conventional methods [1–15], we estimate the frame-level metric and note-level metrics as follows: (1) *Frame*: the activation of quantized pitches in each time-processing frame, (2) *Note*: the onset time of each note, (3) *Note with Offset*: the onset and offset time of each note, and (4) *Note with Offset and Velocity*: the onset, offset time, and the loudness of each note.

For automatic piano transcription, it is important to analyze several harmonic structures that spread in a wide range of frequencies, since piano excerpts are usually polyphonic. Convolutional neural network (CNN)-based methods have been used to aggregate harmonic structures as acoustic features. Most conventional methods apply multi-layer convolutional blocks to extend the receptive field in the frequency axis. However, the blocks often include pooling or striding to downsample the features in the frequency axis. Such a downsampling process may reduce the frequency resolution [6]. It is worth mentioning, many of these methods use 2-D convolutions, which means the convolution is simultaneously applied in the frequency and time axes. The convolution in the time axis works as a pre-emphasis filter to model the temporal changes of the input signals.

Up to now, recurrent neural networks (RNNs), such as gated recurrent unit (GRU) [16] and long short-term memory (LSTM) [17], are popular for analyzing the temporal sequences of acoustic features. However, recently some of the works start to use Transformer [18], which is a powerful tool for analyzing sequences, in AMT tasks. Ou et al. [2] applied a Transformer encoder along the time axis and suggested that using Transformer improves velocity estimation. Hawthorne et al. [3] used a Transformer encoder-decoder as a sequence-to-sequence model for estimating a sequence of note events from another sequence of input audio spectrograms. Their method outperformed other methods using GRUs or LSTMs. Lu et al. [19] proposed a method called SpecTNT to apply Transformer encoders in both frequency and time axes and reached state-of-the-art performance for various MIR tasks such as music tagging, vocal melody extraction, and chord recognition. This suggests that such a combination of encoders helps in characterizing the broad-scale dependency in the frequency and time axes. However, SpecTNT aggregates spectral features into one token, and the process in its temporal Transformer encoder is not independent in the frequency axis. This inspires us to incorporate Transformer encoders in the frequency and time axes and make the spectral information available for the temporal Transformer encoder.

In addition, we usually divide the input signal into chunks since the entire sequence is often too long to be



© K. Toyama, T. Akama, Y. Ikemiya, Y. Takida, W. H. Liao, and Y. Mitsufuji. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Toyama, T. Akama, Y. Ikemiya, Y. Takida, W. H. Liao, and Y. Mitsufuji, “Automatic Piano Transcription with Hierarchical Frequency-Time Transformer”, in *Proc. of the 24th Int. Society for Music Information Retrieval Conf.*, Milan, Italy, 2023.

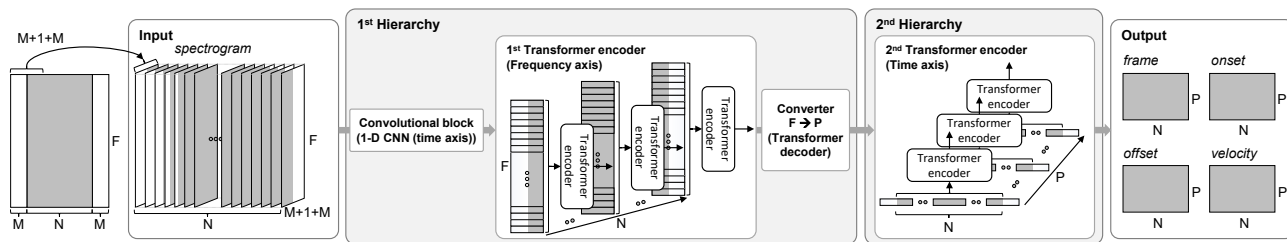


Figure 1. hFT-Transformer (N: number of frames in each processing chunk, M: length of margin, F: number of frequency bins, P: number of pitches)

dealt at once. However, this raises a problem that the estimated onset and offset accuracy fluctuates depending on the relative position in the processing chunk. In our observation, the accuracy tends to be worse at both ends of the processing chunk. This motivates us to incorporate extra techniques during the inference time to boost the performance.

In summary, we propose *hFT-Transformer*, an automatic piano transcription method that uses a two-level hierarchical frequency-time Transformer architecture. Its workflow is shown in Figure 1. The first hierarchy consists of a one-dimensional (1-D) convolutional block in the time axis, a Transformer encoder in the frequency axis, and a Transformer decoder in the frequency axis. The second hierarchy consists of another Transformer encoder in the time axis. In particular, the Transformer decoder at the end of the first hierarchy converts the dimension in the frequency axis from the number of frequency bins to the number of pitches (88 for piano). Regarding the issue of the location dependent accuracy fluctuation in the processing chunks, we propose a technique which halves the stride length at inference time. It uses only the result of the central part of processing chunks, which will improve overall accuracy. Finally, in Section 4, we show that our method outperforms other piano transcription methods in terms of F1 scores for all the four metrics.

A PyTorch implementation of our method is available here¹.

2. RELATED WORK

Neural networks, such as CNNs, RNNs, generative adversarial networks (GANs) [20], and Transformers have been dominant for AMT. Since Sigtia et al. [4] proposed the first method to use a CNN to tackle AMT, CNNs have been widely used for the methods of analyzing the spectral dependency of the input spectrogram [2, 6–10, 12–15]. However, it is difficult for CNNs to directly capture the harmonic structure of the input sound in a wide range of frequencies, as convolutions are used to capture features in a local area. Wei et al. [5] proposed a method of using harmonic constant-Q transform (CQT) for capturing the harmonic structure of piano sounds. They first applied a 3-Dimensional CQT, then applied multiple dilated convolutions with different dilation rates to the output of

CQT. Because the dilation rates are designed to capture the harmonics, the performance of *Frame* and *Note* accuracy reached state-of-the-art. However, the dilation rates are designed specifically for piano. Thus, the method is not easy to adapt to other instruments.

For analysis of time dependency, Kong et al. [6] proposed a method that uses GRUs. Howthorner et al. [7], Kwon et al. [8], Cheuk et al. [9], and Wei et al. [5] proposed methods that use bi-directional LSTMs for analysis. Ou et al. [2] used a Transformer encoder to replace the GRUs in Kong et al.’s method [6], and showed the effectiveness of the Transformer. Usually, the note onset and offset are estimated in each frequency and time-processing frame grid, then paired as a note for note-level transcription by post-processing algorithms such as [6]. However, compared to heuristically designed algorithms, end-to-end data-driven methods are often preferred. For example, Keltz et al. [10] applied a seven-state hidden Markov model (HMM) for the sequence of attack, decay, sustain, and release to achieve note-level transcription. Kwon et al. [8] proposed a method of characterizing the output of LSTM as a five-state statement (onset, offset, re-onset, activate, and inactivate). Hawthorne et al. [3] proposed a method of estimating a sequence of note events, such as note pitch, velocity, and time, from another sequence of input audio spectrograms using a Transformer encoder-decoder. This method performs well in multiple instruments with the same model [11]. Yan et al. [12] proposed a note-wise transcription method for estimating the interval between onset and offset. This method shows state-of-the-art performance in estimating *Note with Offset* and *Note with Offset and Velocity*. However, the performance in estimating *Frame* and *Note* is worse than that of Wei et al.’s method [5].

3. METHOD

3.1 Configuration

Our proposed method aims to transcribe N frames of the input spectrogram into N frames of the output piano rolls (*frame*, *onset*, *offset*, and *velocity*) as shown in Figure 1, where N is the number of frames in each processing chunk. Each input frame is composed of a log-mel spectrogram having size $(F, M + 1 + M)$, where F is the number of frequency bins, and M is the size of the forward margin and that of the backward margin. To obtain

¹ <https://github.com/sony/hFT-Transformer>

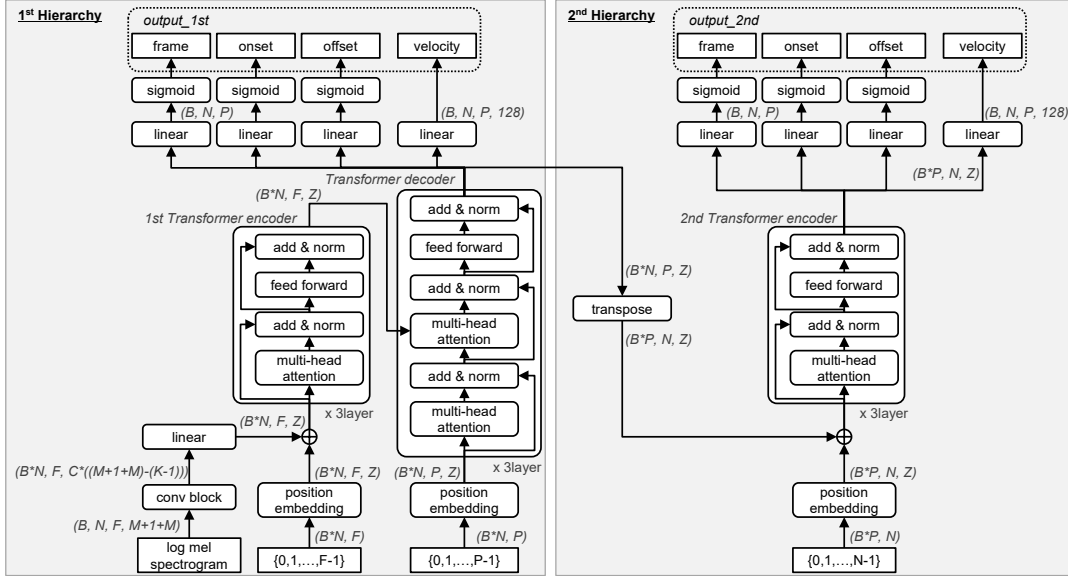


Figure 2. Model architecture of hFT-Transformer

the log-mel spectrogram, we first downmix the input waveform into one channel and resample them to 16 kHz. Then, the resampled waveform is transformed into a mel spectrogram with `transforms.MelSpectrogram` class in the `Torchaudio` library [21]. For the transformation, we use *hann* window, setting the window size as 2048, fast-Fourier-transform size as 2048, F as 256, padding mode as *constant*, and hop-size as 16 ms. The magnitude of the mel spectrogram is then compressed with a log function.

3.2 Model Architecture and Loss Functions

The model architecture of our proposed method is shown in Figure 2. We first apply a convolutional block to the input log-mel spectrogram, the size of which is $(B, N, F, M+1+M)$ where B is the batch size. In the convolutional block, we apply a 1-D convolution in the $M+1+M$ dimension. After this process, the data are embedded with a linear module.

The embedded vector is then processed with the first Transformer encoder in the frequency axis. The self-attention is processed to analyze the dependency between spectral features. The positional information is designated as $[0, 1, \dots, F-1]$. These positional values are then embedded with a trainable embedding. These are processed in the frequency axis only, thus completely independent to the time axis (N dimension).

Next, we convert the frequency dimension from F to the number of pitches (P). A Transformer decoder with cross-attention is used as the converter. The Transformer decoder calculates the cross-attention between the output vectors of the first Transformer encoder and another trainable positional embedding made from $[0, 1, \dots, P-1]$. The decoded vectors are then converted to the outputs of the first hierarchy with a linear module and a sigmoid function (hereafter, we call these outputs *output_1st*).

Regarding the loss calculation for the outputs, *frame*, *onset*, and *offset* are calculated with binary cross-entropy,

and *velocity* is calculated with 128-category cross-entropy. The losses can be summarized as the following equations:

$$L_{\text{bce}}^{<m>} = \sum_{n=0}^{N-1} \sum_{p=0}^{P-1} l_{\text{bce}}(y_{n,p}^{<m>}, \hat{y}_{n,p}^{<m>}), \quad (1)$$

$$L_{\text{cce}}^{\text{velocity}} = \sum_{n=0}^{N-1} \sum_{p=0}^{P-1} l_{\text{cce}}(y_{n,p}^{\text{velocity}}, \hat{y}_{n,p}^{\text{velocity}}), \quad (2)$$

$$L = L_{\text{bce}}^{\text{frame}} + L_{\text{bce}}^{\text{onset}} + L_{\text{bce}}^{\text{offset}} + L_{\text{cce}}^{\text{velocity}}, \quad (3)$$

where $<m>$ is the placeholder for each output (*frame*, *onset*, and *offset*), l_{bce} and l_{cce} denote the loss function for binary cross-entropy and categorical cross-entropy, respectively, and y and \hat{y} denote the ground truth and predicted values of each output (*frame*, *onset*, *offset*, and *velocity*), respectively. Although it is intuitive to apply the mean squared error (MSE) for *velocity*, we found that using the categorical cross-entropy yields much better performance than the MSE from a preliminary experiment.

Finally, the output of the converter is processed with another Transformer encoder in the time axis. The self-attention is used to analyze the temporal dependency of features in each time-processing frame. A third positional embedding made from $[0, 1, \dots, N-1]$ is used here. Then, similar to the first hierarchy, the outputs of the second hierarchy are obtained through a linear module and a sigmoid function. We call these outputs of the second hierarchy as *output_2nd* hereafter. The losses for the *output_2nd* are evaluated in the same way as those for *output_1st*. These losses are summed with the coefficients $\alpha_{1\text{st}}$ and $\alpha_{2\text{nd}}$ as follows:

$$L_{\text{all}} = \alpha_{1\text{st}} L_{1\text{st}} + \alpha_{2\text{nd}} L_{2\text{nd}}. \quad (4)$$

Although both outputs are used for computing losses during training, only *output_2nd* is used in inference. As Chen et al. [22] suggested that the performance of their method of calculating multiple losses outperformed the method

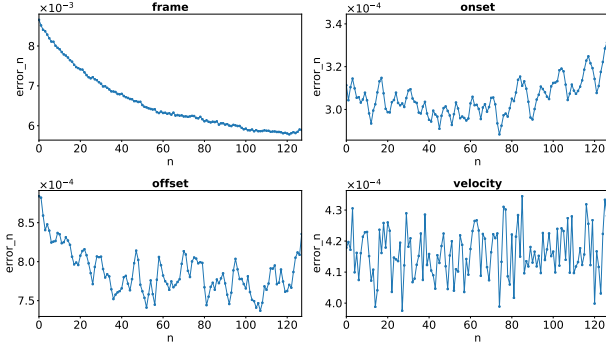


Figure 3. Estimation error (Eqn (5)) on location in each time-processing frame

that uses single loss only, it hints us that utilizing both *output_1st* and *output_2nd* in training has the potential to achieve better performance.

3.3 Inference Stride

As mentioned in Section 1, chunk-based processing is required because the input length is limited due to system limitations, such as memory size and acceptable processing delay. We found that the estimation error tends to increase at certain part within each processing chunk. This can be demonstrated by evaluating the error for each instance of time n within the chunks:

$$error_n^{<m>} = \frac{1}{IP} \sum_{i=0}^{I-1} \sum_{p=0}^{P-1} (y_{i,n,p}^{<m>} - \hat{y}_{i,n,p}^{<m>})^2, \quad (5)$$

where $<m>$ is the placeholder for each output (*frame*, *onset*, *offset*, and *velocity*), and I is the number of processing chunks over the test set. The result using our proposed model trained using the MAESTRO training set (described in Section 4) is shown in Figure 3. Here, the error $error_n^{<m>}$ is calculated using the MAESTRO test set. In the figure, we observe a monotonic decrease for *frame* and a similar but much weaker trend for *onset* and *offset*. However, for *velocity*, no such trend can be observed. This hints us to use only the middle portion of a processing chunk as the output to reduce the error rate. We call this as the half-stride strategy, since a 50% overlap is required for processing chunks, as shown in Figure 4 (B).

4. EXPERIMENTS

4.1 Datasets

We use two well-known piano datasets for the evaluation. The MAPS dataset [23] consists of CD-quality recordings and corresponding annotations of isolated notes, chords, and complete piano pieces. We use the full musical pieces and the train/validation/test split as stated in [4, 7]. The number of recordings and the total duration in hours in each split are 139/71/60 and 8.3/4.4/5.5, respectively. The MAESTRO v3.0.0 dataset [13] includes about 200 hours of paired audio and MIDI recordings from ten years of the International Piano-e-Competition. We used the

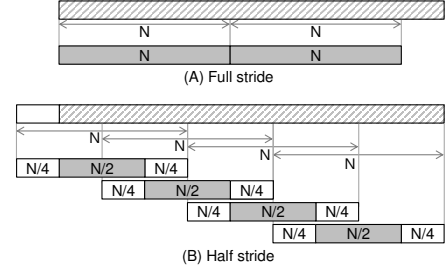


Figure 4. Inference stride: (A) full stride, (B) half stride

train/validation/test split configuration as provided. In each split, the number of recordings and total duration in hours are 962/137/177 and 159.2/19.4/20.0, respectively. For both datasets, the MIDI data have been collected by Yamaha Disklaviers concert-quality acoustic grand pianos integrated with a high-precision MIDI capture and playback system.

4.2 Model Configuration

Regarding our model architecture depicted in Figure 2, we set N as 128, M as 32, F as 256, P as 88, the CNN channels (C) as 4, size of the CNN kernel (K) as 5, and embedding vector size (Z) as 256. For the Transformers, we set the feed-forward network vector size as 512, the number of heads as 4, and the number of layers as 3. For training, we used the following settings: a batch size of 8, learning rate of 0.0001 with Adam optimizer [24], dropout rate of 0.1, and clip norm of 1.0. ReduceLROnPlateau in PyTorch is used for learning rate scheduling with default parameters. We set α_{1st} and α_{2nd} as 1.0, which were derived from a preliminary experiment (see Section 4.6).

We trained our models for 50 epochs on MAPS dataset and 20 epochs for MAESTRO dataset using one NVIDIA A100 GPU. It took roughly 140 minutes and 43.5 hours to train one epoch with our model for MAPS and MAESTRO, respectively. The best model is determined by choosing the one with the highest F1 score in the validation stage.

In order to obtain high-resolution ground truth for *onset* and *offset*, we followed the method in Kong et al. [6]. We set J , the hyper-parameter to control the sharpness of the targets, to 3. Also, the label of *velocity* is set only when an *onset* is present. We set the threshold as 0.5, which means if the *onset* is smaller than 0.5, the *velocity* is set as 0.

4.3 Inference

At inference time, we use *output_2nd* as the final output. We set the threshold for *frame* as 0.5. For note-wise events (*onset*, *offset*, and *velocity*), the outputs in each pitch-frame grid are converted to a set containing note-wise onset, offset, and velocity following Kong et al.'s *Algorithm 1* [6] in five steps shown below:

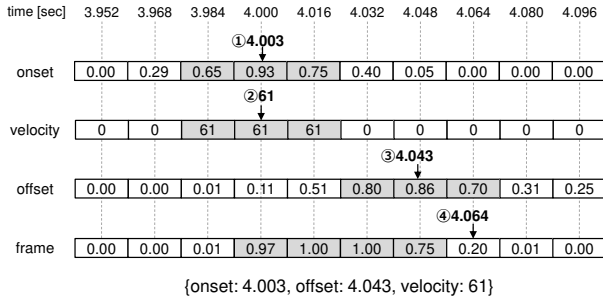
Step 1. onset detection: find a local maximum in *onset* with a value at least 0.5. Then calculate the precise onset time using the values of the adjacent three frames [6].

Step 2. velocity: If an onset is detected in Step 1, extract the *velocity* value at the frame. If the value is zero, then

Method	half stride	Params	Frame			Note			Note w/ Offset			Note w/ Offset&Velocity		
			P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Onsets&Frames [7]		26M	<u>88.53</u>	70.89	78.30	84.24	80.67	82.29	51.32	49.31	50.22	35.52	30.80	35.59
ADSR [10]		0.3M	90.73	67.85	77.16	90.15	74.78	81.38	61.93	51.66	56.08	-	-	-
hFT-Transformer		5.5M	83.36	<u>82.00</u>	<u>82.67</u>	86.63	<u>83.75</u>	<u>85.07</u>	<u>67.18</u>	<u>65.06</u>	<u>66.03</u>	<u>48.75</u>	<u>47.21</u>	<u>47.92</u>
hFT-Transformer	✓	5.5M	83.68	82.11	82.89	<u>86.72</u>	83.81	85.14	67.51	65.36	66.34	49.05	47.48	48.20

Table 1. Evaluation results on MAPS test dataset (P: precision, R: recall, **bold**: best score, underline: second best score)

Method	half stride	Params	Frame			Note			Note w/ Offset			Note w/ Offset&Velocity		
			P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Seq2Seq [3]		54M	-	-	-	-	-	96.01	-	-	83.94	-	-	82.75
HPT-T [2]		-	-	-	90.09	97.88	96.72	96.77	84.13	82.31	83.20	82.85	81.07	81.90
Semi-CRFs [12]		9M	93.79	88.36	90.75	98.69	93.96	96.11	90.79	86.46	88.42	89.78	85.51	87.44
HPPNet-sp [5]		1.2M	92.79	<u>93.59</u>	<u>93.15</u>	98.45	<u>95.95</u>	97.18	84.88	82.76	83.80	83.29	81.24	82.24
hFT-Transformer		5.5M	92.62	93.43	<u>93.02</u>	<u>99.62</u>	95.41	<u>97.43</u>	<u>92.32</u>	88.48	<u>90.32</u>	<u>91.21</u>	<u>87.44</u>	<u>89.25</u>
hFT-Transformer	✓	5.5M	<u>92.82</u>	93.66	93.24	99.64	95.44	97.44	92.52	88.69	90.53	91.43	87.67	89.48

Table 2. Evaluation results on MAESTRO v3.0.0 test dataset

Figure 5. Example of conversion from grid-wise values to note-wise values

discard both onset and velocity at this frame.

Step 3. offset detection with *offset*: find a local maximum in *offset* with a value at least 0.5. Then calculate the precise offset time using the values of the adjacent three frames [6].

Step 4. offset detection with *frame*: choose the frame that is nearest to the detected onset which has a *frame* value below 0.5.

Step 5. offset decision: choose the smaller value between the results of Step 3 and 4.

An example is shown in Figure 5. The *onset* is 4.003, and the *velocity* is 61. For *offset*, the direct estimation from *offset* is 4.043, and that estimated via *frame* is 4.064. Thus, we choose 4.043 as *offset*. Finally, we obtain a note with {*onset*: 4.003, *offset*: 4.043, *velocity*: 61} in the output.

4.4 Metrics

We evaluate the performance of our proposed method with frame-level metrics (*Frame*) and note-level metrics (*Note*, *Note with Offset*, and *Note with Offset & Velocity*) with the standard precision, recall, and F1 scores. We calculated these scores using `mir_eval` library [25] with its default settings. The scores were calculated per recording, and the mean of these per-recording scores was presented as the final metric for a given collection of pieces, as explained

in Hawthorne et al. [7].

4.5 Results

Tables 1 and 2 show the scores on the test sets of MAPS and MAESTRO datasets. The numbers of parameters in these Tables are referred from [5, 10]. For the MAPS dataset, our proposed method outperformed the other methods in F1 score for all metrics. For the MAESTRO dataset, our proposed method outperformed the other methods in F1 score for *Note*, *Note with Offset*, and *Note with Offset & Velocity*. Furthermore, our method with the half-stride strategy which is mentioned in 3.3 outperformed other methods in all metrics. In contrast, the two state-of-the-art methods for MAESTRO, which are Semi-CRFs [12] and HPPNet-sp [5], performed well only on a subset of the metrics.

The results suggest that the proposed two-level hierarchical frequency-time Transformer structure is promising for AMT.

4.6 Ablation Study

To investigate the effectiveness of each module in our proposed method, we trained various combinations of those modules using the MAPS training set and evaluated them using the MAPS validation set. The variations are shown in Table 3. In this study, we call our proposed method *I-F-D-T*, which means it consists of the *I-D* convolution block, the first Transformer encoder in the *Frequency* axis, the Transformer Decoder, and the second Transformer encoder in the *Time* axis. Table 4 shows evaluation results for each variation.

Second Transformer encoder in time axis. To verify the effectiveness of the second Transformer encoder, we compared the *I-F-D-T* and the model without the second Transformer encoder (*I-F-D-N*). For the *I-F-D-N* model, we use *output_1st* in both training and inference stages as the final output. The result indicates that the second Transformer encoder improved *Note with Offset* performance, in

Model	1st-Hierarchy			2nd-Hierarchy		Output
	Convolutional block	1st Transformer encoder	Converter	2nd Transformer encoder		
1-F-D-T†	1-D (time axis)	Frequency axis	Transformer Decoder	Time axis		output_2nd
1-F-D-N	1-D (time axis)	Frequency axis	Transformer Decoder	n/a		output_1st
2-F-D-T	2-D	Frequency axis	Transformer Decoder	Time axis		output_2nd
1-F-L-T	1-D (time axis)	Frequency axis	Linear	Time axis		output_2nd

Table 3. Model variations for ablation study (†: the proposed method, hFT-Transformer)

Model	Params	Frame			Note			Note w/ Offset			Note w/ Offset&Velocity		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
1-F-D-T†	5.5M	93.61	88.71	91.09	98.81	94.81	96.72	86.18	82.81	84.42	77.47	74.55	75.95
1-F-D-N	3.9M	92.85	87.49	90.09	99.01	93.24	95.95	82.67	78.06	80.23	73.89	69.90	71.78
2-F-D-T	6.1M	75.49	61.08	67.52	97.03	19.68	31.10	64.07	13.28	20.88	42.11	8.57	13.50
1-F-L-T	3.4M	93.71	88.42	90.99	99.11	92.90	95.79	85.77	80.56	82.98	71.66	67.32	69.34

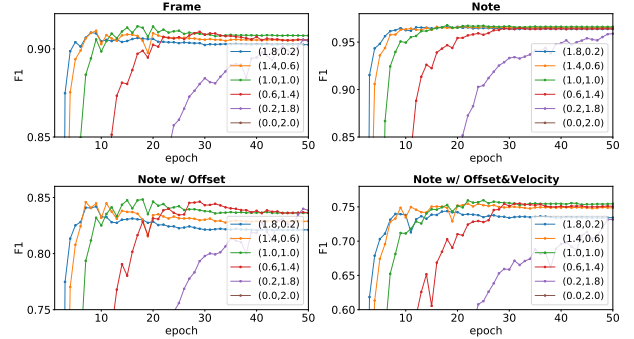
Table 4. Evaluation results of ablation study on MAPS validation dataset

which the F1 score is 84.42 for 1-F-D-T and 80.23 for 1-F-D-N. This shows the effectiveness of the second Transformer encoder as it provides an extra pass to model the temporal dependency of acoustic features, which is presumably helpful in offset estimation.

Complexity of convolutional block. To investigate how the complexity of the convolutional block affects the AMT performance, we compared the 1-F-D-T model and the model that replaces the 1-D convolutional block with a 2-D convolutional block (2-F-D-T). Surprisingly, the result shows that the performance of the 2-F-D-T model is significantly worse than that of the 1-F-D-T model. This is probably because the two modules working on the spectral dependency do not cohere with each other. The 2-D convolutional block may over aggregate the spectral information thus resulting into an effectively lower frequency resolution. Then, the Transformer encoder can only evaluate the spectral dependency over an over-simplified feature space, causing the performance degradation.

Converter. We used a Transformer decoder to convert the dimension in the frequency axis from F to P . In contrast, almost all of the existing methods used a linear module to achieve this. We compared the performance of the 1-F-D-T model to a model with the Transformer decoder replaced with a linear converter (1-F-L-T). The result indicates that the 1-F-D-T model outperformed the 1-F-L-T model in F1 score for all four metrics. Especially, the difference in *Note with Offset and Velocity* is large (75.95 for the 1-F-D-T model and 69.34 for the 1-F-L-T model in F1 score). This suggests that using a Transformer decoder as converter is an effective way of improving the performance, although the side effect is the increase of model size.

We also investigated how the coefficients for the loss functions, α_{1st} and α_{2nd} in Eqn (4), affect the performance. We investigated six pairs of coefficients of loss functions (α_{1st} , α_{2nd}) in Eqn (4), i.e., (1.8, 0.2), (1.4, 0.6), (1.0, 1.0), (0.6, 1.4), (0.2, 1.8), and (0.0, 2.0), for the 1-F-D-T model. Figure 6 shows the F1 scores of *frame*, *onset*, and *velocity* evaluated on the MAPS validation set in each epoch. These results indicate that the (1.0, 1.0) pair


Figure 6. Performance of 1-F-D-T model trained with six-pairs of coefficients of loss functions

yields the best score. It also shows that the training converges faster when α_{1st} is larger than α_{2nd} . Importantly, if we omit the *output_1st*, which is the case when training with the pair (0.0, 2.0), the training loss did not decrease much. Therefore, the F1 score stays around 0% and thus cannot be seen in Figure 6. This suggests that it is crucial to use both losses, *output_1st* and *output_2nd* in our proposed method.

5. CONCLUSION

In this work, we proposed *hFT-Transformer*, an automatic piano transcription method that uses a two-level hierarchical frequency-time Transformer architecture. The first hierarchy consists of a 1-D convolutional block in the time axis, a Transformer encoder and a Transformer decoder in the frequency axis, and the second hierarchy consists of a Transformer encoder in the time axis. The experiment result based on two well-known piano datasets, MAPS and MAESTRO, revealed that our two-level hierarchical architecture works effectively and outperformed other state-of-the-art methods in F1 score for frame-level and note-level transcription metrics. For future work, we would like to extend our method to other instruments and multi-instrument settings.

6. ACKNOWLEDGMENTS

We would like to thank Giorgio Fabbro and Stefan Uhlich for their valuable comments while preparing this manuscript. We are grateful to Kin Wai Cheuk for his dedicated support in preparing our github repository.

7. REFERENCES

- [1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, “Automatic music transcription: An overview,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [2] L. Ou, Z. Guo, E. Benetos, J. Han, and Y. Hang, “Exploring transformer’s potential on automatic piano transcription,” in *Proc. of the 47th Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 776–780.
- [3] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, “Sequence-to-sequence piano transcription with transformers,” in *Proc. of the 22th Int. Society for Music Information Retrieval Conf.*, 2021, pp. 246–253.
- [4] S. Sigtia, E. Benetos, and S. Dixon, “An end-to-end neural network for polyphonic piano music transcription,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 5, pp. 927–939, 2016.
- [5] W. Wei, P. Li, Y. Yu, and W. Li, “Hppnet: Modeling the harmonic structure and pitch invariance in piano transcription,” in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, 2022, pp. 709–716.
- [6] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, “High-resolution piano transcription with pedals by regressing onsets and offsets times,” *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 29, pp. 3707–3717, 2021.
- [7] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” in *Proc. of the 19th Int. Society for Music Information Retrieval Conf.*, 2018, pp. 50–57.
- [8] T. Kwon, D. Jeong, and J. Nam, “Polyphonic piano transcription using autoregressive multi-state note model,” in *Proc. of the 21st Int. Society for Music Information Retrieval Conf.*, 2020, pp. 454–460.
- [9] K. W. Cheuk, Y.-J. Luo, E. Benetos, and D. Herremans, “The effect of spectrogram reconstruction on automatic music transcription: An alternative approach to improve transcription accuracy,” in *Proc. of the 25th International Conference on Pattern Recognition (ICPR)*, 2020, pp. 9091–9098.
- [10] R. Kelz, S. Böck, and G. Widmer, “Deep polyphonic adsr piano note transcription,” in *Proc. of the 44th Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 246–250.
- [11] J. Gardner, I. Simon, E. Manilow, and C. H. J. Engel, “Mt3: Multi-task multitrack music transcription,” in *Proc. of the Int. Conference on Learning Representations (ICLR)*, 2022.
- [12] Y. Yan, F. Cwitkowitz, and Z. Duan, “Skipping the frame-level: Event-based piano transcription with neural semi-crfs,” in *Proc. of the 36th Int. Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
- [13] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the maestro dataset,” in *Proc. of the 7th International Conference on Learning Representations (ICLR)*, 2019.
- [14] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer, “On the potential of simple frame-wise approaches to piano transcription,” in *Proc. of the 17th Int. Society for Music Information Retrieval Conf.*, 2016, pp. 475–481.
- [15] J. W. Kim and J. P. Bello, “Adversarial learning for improved onsets and frames music transcription,” in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, 2019, pp. 670–677.
- [16] C. Kyunghyun, van Merriënboer Bart, G. Caglar, B. Dzmitry, B. Fethi, S. Holger, and B. Yoshua, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.
- [17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Iosifidou, “Attention is all you need,” in *Proc. of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6000–6010.
- [19] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, “Spectnt: A time-frequency transformer for music audio,” in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, 2021, pp. 396–403.
- [20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [21] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhersch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala,

- V. Quenneville-Bélair, and Y. Shi, “Torchaudio: Building blocks for audio and speech processing,” *arXiv preprint arXiv:2110.15018*, 2021.
- [22] Y.-H. Chen, W.-Y. Hsiao, T.-K. Hsieh, J.-S. R. Jang, and Y.-H. Yang, “Towards automatic transcription of polyphonic electric guitar music: A new dataset and a multi-loss transformer model,” in *Proc. of the 47th Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 786–790.
- [23] V. Emiya, R. Badeau, and B. David, “Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle,” *IEEE ACM Transactions on Audio Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2009.
- [24] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.
- [25] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nietro, D. Liang, and D. Ellis, “mir_eval: A transparent implementation of common mir metrics,” in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, 2014, pp. 367–372.