SpecTNT: A TIME-FREQUENCY TRANSFORMER FOR MUSIC AUDIO

Wei-Tsung Lu[1] **Ju-Chiang Wang**[1] **Minz Won**[1,2] **Keunwoo Choi**[1] **Xuchen Song**[1]
[1] ByteDance, Mountain View, California, United States
[2] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{weitsung.lu, ju-chiang.wang, minzwon, keunwoo.choi, xuchen.song}@bytedance.com

ABSTRACT

Transformers have drawn attention in the MIR field for their remarkable performance shown in natural language processing and computer vision. However, prior works in the audio processing domain mostly use Transformer as a temporal feature aggregator that acts similar to RNNs. In this paper, we propose SpecTNT, a Transformer-based architecture to model both spectral and temporal sequences of an input time-frequency representation. Specifically, we introduce a novel variant of the Transformer-in-Transformer (TNT) architecture. In each SpecTNT block, a spectral Transformer extracts frequency-related features into the frequency class token (FCT) for each frame. Later, the FCTs are linearly projected and added to the temporal embeddings (TEs), which aggregate useful information from the FCTs. Then, a temporal Transformer processes the TEs to exchange information across the time axis. By stacking the SpecTNT blocks, we build the SpecTNT model to learn the representation for music signals. In experiments, SpecTNT demonstrates state-of-the-art performance in music tagging and vocal melody extraction, and shows competitive performance for chord recognition. The effectiveness of SpecTNT and other design choices are further examined through ablation studies.

1. INTRODUCTION

Deep learning models have been actively used in recent music information retrieval (MIR) research. Although the spirit of deep learning is end-to-end learning, however, various assumptions are made during making design choices of deep learning models.

Regarding assumptions on spectrograms, the most popular form of music audio representation in deep learning, the time-axis is often considered to be the axis of *sequence* while the frequency-axis is the axis of *feature*. For example, in [1, 2], recurrent layers were applied to model a spectrogram as a sequence of spectra. In [3], convolutional layers were used to aggregate features over time after the first convolutional layer models multiple frames of spectra as feature. On the other hand, in [4], two-dimensional convolutional layers were used, equating the frequency- and time-axes. There are also hybrid approaches, such as convolutional recurrent neural networks (CRNN) [5] and convolutional Transformer [6], in which recurrent layers or Transformer are applied along the time axis.

In spectrograms, it is well known that there are meaningful spectral patterns. Different music components exist in different frequency ranges, and there is a very strong spectral correlation called harmonics. Since a normal convolutional layer can model local patterns only, several approaches have been proposed to model harmonics along the frequency axis. Harmonic Constant-Q Transform (HCQT) is a novel multi-channel time-frequency representation that was proposed to overcome the limitation by improving the input representation of audio [7]. Harmonic CNNs (convolutional neural networks) [8] are designed to model the harmonic pattern by modifying the convolutional filters. However, these solutions only model some of the spectral patterns, reminding the need for a more general solution with higher flexibility.

Transformers have successfully demonstrated their ability to model the sequential data with long-term (inter-) dependency and invariance. This is achieved by multiple aspects of Transformers. First, the key-query mechanism enables modeling the relationship of every combination of the instances. Second, positional encoding helps the model to take the order of instances into account. Through stacked attention layers, the input sequence is transformed into a sequence of representations that are based on the inter-dependency of the input. The prior works in audio analysis are mostly based on a similar, naive approach where Transformer is used as a temporal feature aggregator that acts similar to RNNs (recurrent neural networks), with few exceptions such as [9].

Recently, Transformer in Transformer (TNT), a variant of Transformer that arranges two Transformers in a hierarchical manner, was proposed [10] for image recognition. In TNT, an inner (lower-level) Transformer is applied to extract the local pixel-level embeddings, and then the pixel-level embeddings are projected to the patch-level embedding space which is later handled by an outer (higher-level) Transformer to summarize a global representation. One can simply apply TNT for audio by treating a song as an image, which is comprised of a sequence of frames (patches) while the frequency bins within frames are con-

sidered as pixels. However, from our pilot study, we find this approach results in unstable training and only achieves similar performance compared to using the original Transformer. This is possibly because the amount of training data is insufficient or the interpretation of frequency sequences is different from that of pixel sequences. Therefore, non-trivial modifications from the original idea of TNT should be made.

In this paper, we propose *SpecTNT*, a time-frequency transformer that models spectrograms as a sequence along both time- and frequency-axes. Similar to TNT, SpecTNT uses two Transformers hierarchically. However, the temporal local embeddings extracted from the inner Transformer are not directly sent to the outer Transformer. Instead, a special token called *frequency class token* (FCT) is appended to aggregate the important spectral features of each frame. The FCT is then projected to the global (temporal) embedding space to enable the information exchange across the time axis. This design allows the important local information is passed to the outer Transformer through FCT while reducing the dimensionality of the data flow compared to the original TNT. As a result, it helps SpecTNTs to perform well on audio-related tasks even with smaller datasets.

Our contributions can be summarized as follows: (1) to the best of our knowledge, our work is the first attempt to leverage TNT-based architecture to learn the representations for audio; (2) we propose SpecTNT, a novel modification of TNT to better fit the music data for MIR tasks; (3) we conduct extensive experiments to demonstrate the capability of SpecTNT in various MIR tasks – vocal melody extraction, music auto-tagging, and chord recognition.

2. RELATED WORK

In this section, we review the literature in music tagging, vocal melody extraction, and chord recognition – three well-defined MIR tasks adopted in the experiments to evaluate SpecTNT. Due to space limitation, we focus on the recent trends since the adoption of deep learning approaches.

Music tagging is a multi-label classification task that annotates a music audio clip with various types of labels such as genres (rock, jazz), instruments (vocal, guitar, drums), and mood (happy, sad) [11]. Since a CNN-based approach has been first introduced [3], various advanced architectures have been used including a two-dimensional CNN [4], a sample-level CNN [12], and a two-dimensional ResNet [13]. Due to the open nature of the tag set, among MIR tasks, music tagging is relatively a *vague* task – The exact mechanism of annotating tags is not fully known. This aspect suits well for the fundamental motivation of deep learning, which is, to reduce inductive bias and let the data speak [14].

The goal of vocal melody extraction is to estimate the F0 frequency of the (dominant) vocal track in given mixtures. Various deep learning methods have been adopted: a fully-connected neural network with Hidden Markov Model [15], a bidirectional long short-term memory network [1], a CNN [7], encoder-decoder networks [16, 17] and a CRNN [18]. Recently, a frequency-temporal attention module was introduced in [19] to learn the relevant regions for predictions. Some special representations are proposed including HCQT [7], a combination of frequency and periodicity [20], and source-separated tracks [21, 22].

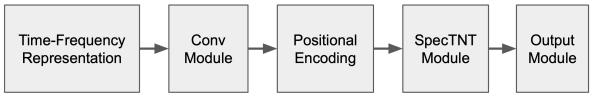

Figure 1. The block diagram of the whole SpecTNT. The details of positional encoding and SpecTNT module are illustrated in Figure 2 and 3, respectively.

Chord recognition is a MIR task to "produce a time-varying symbolic representation of the signal in terms of chord labels" [23]. Compared to music tagging, we clearly understand how chords of music signals can be decided – They are based on the combination of the present musical notes. Therefore, models have been designed to take advantage of note representations such as constant-Q transform (CQT) or chromagram. The early deep learning-based chord recognition models are based on a RNN [24] and a CNN [25]. Later, a CRNN has been used in [23] to combine the merits of RNNs and CNNs. More recently, (bi-directional) Transformer was used, achieving state-of-the-art performance [26, 27].

3. METHODS

As illustrated in Figure 1, the proposed SpecTNT architecture consists of a convolutional module, positional encoding, SpecTNT module, and output module.

The input time-frequency representation is first processed with a stack of convolutional layers for local feature aggregation. Then, the positional information is added to the data. In the SpecTNT module, the intermediate representation is fed into a stack of SpecTNT blocks. Lastly, the output module projects the final embedding into the desired dimension for different tasks. We detail each module in the following subsections.

3.1 Convolutional module

The purpose of this convolutional module is to employ different strategies for generating intermediate representations with pooling or striding convolution techniques depending on the nature of the task. Let the input time-frequency representation be $S \in \mathbb{R}^{T \times F \times K}$ where T is the number of time-steps, F is the number of frequency bins, and K is the number of channels. S is first passed into a stack of convolutional layers. We utilize the residual unit proposed in [28] to be the basic building block of the convolutional module. The representation after the convolutional module is denoted as $S' = [S'_1, S'_2, ..., S'_{\hat{T}}] \in \mathbb{R}^{\hat{T} \times \hat{F} \times \hat{K}}$, where $\hat{F}$, $\hat{T}$, and $\hat{K}$ are the numbers of frequency bins, time-steps, and channels, respectively.

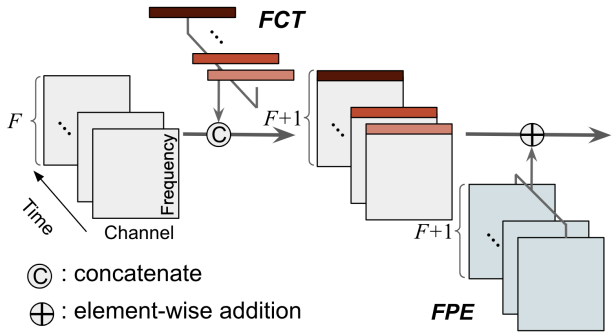

Figure 2. An illustration of the application of Frequency class token (FCT) and frequency positional encoding (FPE). F refers to the number of frequency bins of the input time-frequency representation.

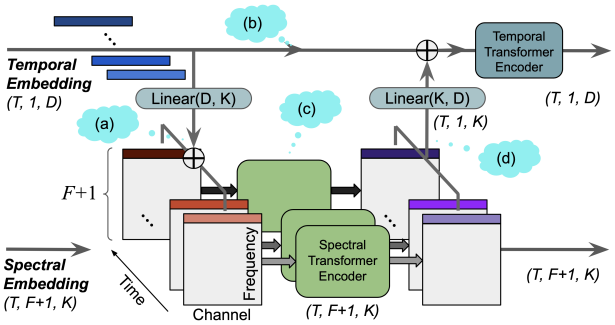

Figure 3. The block diagram of a SpecTNT block. Tensors and modules are illustrated with non-rounded and rounded rectangles, respectively. We specify the non-batch shape of tensors for clarity, and explain (a) – (d) in the main text.

3.2 Frequency Class Token

As depicted in Figure 2, *Frequency class token* (FCT) is an embedding vector initialized with all zeros to serve as the placeholder and defined as $c_t = \mathbf{0}^{1\times\hat{K}}$. Let $S'_t \in \mathbb{R}^{\hat{F}\times\hat{K}}$ denote the input data at each time-step t. The input data and FCT are concatenated as following:

$$S''_t = \text{Concat}[c_t, S'_t]. \tag{1}$$

Here, the role of FCT c_t is similar to the classification token [29]. It is expected to extract spectral features from each frequency bin of the t-th frame during the spectral self-attention in the later stages.

3.3 Positional encoding

In the original Transformer paper, a sinusoidal positional encoding was added to the input sequence to make the following layers aware of the order of input elements [30]. From a similar motivation, we adopt a learnable positional embedding to encode the sequence order of frequency bins.

We encode the positional information of frequencies by adding the frequency positional embedding (FPE) to the data S''. FPE is a learnable matrix $E^{\phi} \in \mathbb{R}^{(\hat{F}+1)\times\hat{K}}$. The addition process is done at each time-step t:

$$\hat{S}_t = S''_t \oplus E^{\phi}, \tag{2}$$

where $\oplus$ is the element-wise addition, and the resulting FCTs are denoted by $\hat{C} = [\hat{c}_1, \hat{c}_2, ..., \hat{c}_{\hat{T}}]$. Then, the resulting representation $\hat{S}_t$ is able to carry information about pitch and timbre to the following attention layers. For example, a pitch in the signal can lead to high energy at a specific frequency bin, and the positional embedding makes FCT aware of the position of that frequency

3.4 Transformer in Transformer (TNT)

Inspired by the architecture in [10], we design a SpecTNT block to handle audio data, as depicted in Figure 3. The SpecTNT block holds two data flows: *spectral embedding* (SE) and *temporal embedding* (TE). The two data flows are respectively processed with two Transformer encoders, namely *temporal Transformer* and *spectral Transformer*. Because the SpecTNT block is repeated multiple times in the SpecTNT module, we introduce a notation l to specify the layer index for both SE and TE.

In the following sections, we explain each component of a SpecTNT block (Section 3.4.1 through Section 3.4.3) and the entire procedure (Section 3.4.4).

3.4.1 Temporal Embedding

In the proposed model, we introduce the temporal embedding (TE) to distribute the information of FCTs across the time axis. We can write the TE at layer l as:

$$E^l = [e^l_1, e^l_2, ..., e^l_{\hat{T}}], \tag{3}$$

where $e^l_t \in \mathbb{R}^{1\times D}$ is a TE vector at time t and D is the number of features. In practice, TE is a learnable matrix and is initialized randomly as $E^0 \in \mathbb{R}^{\hat{T}\times D}$ prior to entering the first SpecTNT block.

There are two bridges between the spectral and temporal data flows. We use FCTs, the first frequency bin of SEs, for this communication. First, TE sends information to FCTs by passing e^l_t to a linear projection layer. Then, the projected D-dimensional vectors are added to FCTs (Figure 3-(a)). Second, after spectral transformer encoder (Figure 3-(c)), FCTs (purple arrays) are projected back to K-dimension (Figure 3-(d)). Note that TE also has a skip-connection (Figure 3-(b)).

3.4.2 Spectral Embedding

The output from the positional encoding, $\hat{\mathcal{S}}$, will serve as an input SE for the first SpecTNT block and is denoted as $\hat{\mathcal{S}}^0$. As mentioned above, SE includes FCTs, which help aggregate useful spectral information from the local. As a general notation, we write the data flow of SE as:

$$\hat{\mathcal{S}}^l = \left[[\hat{c}^l_1, \hat{S}^l_1], [\hat{c}^l_2, \hat{S}^l_2], ..., [\hat{c}^l_{\hat{T}}, \hat{S}^l_{\hat{T}}]\right], \tag{4}$$

where $l = 0, 1, ..., L$, and c^l_t and $\hat{S}^l_t$ are respectively the FCTs of l-th layer and spectral data at time-step t. Then, SE can interact with the TE through FCTs, so the local spectral features can be processed in a temporal and global manner.

3.4.3 Transformer Encoder

A Transformer encoder is composed of three components: multi-head self-attention (MHSA), feed-forward network (FFN), and layer normalization (LN).

Self-attention (SA) [30] plays the pivotal role in a Transformer encoder. It takes three inputs: $Q \in \mathbb{R}^{T \times d_q}$, $K \in \mathbb{R}^{T \times d_k}$ and $V \in \mathbb{R}^{T \times d_v}$ which represent the queries, keys, and values, respectively. T is the number of time-steps, and d_q, d_k and d_v indicate the dimension of features for Q, K, and V, respectively. The output is the weighted sum over the values based on the dot product similarity between queries and keys at the corresponding time-step.

The MHSA module [30] is an extension of SA. It splits the three inputs Q, K and V along their feature dimension into h number of "heads" and performs multiple SA's, each on a head, in parallel. The outputs of heads are then concatenated and linearly projected into the final output. The FFN module has two linear layers with a GELU activation function in the middle. We also adopt the pre-norm residual units [31] to stabilize the training.

With the three components, the Transformer encoder (either spectral or temporal) is built and denoted by

$$X_l = \mathrm{Enc}(X_{l-1}), \tag{5}$$

where the operations within it can be written as

$$\begin{aligned} X'_{l-1} &= X_{l-1} + \mathrm{MHSA}(\mathrm{LN}(X_{l-1})), \\ X_l &= X'_{l-1} + \mathrm{FFN}(\mathrm{LN}(X'_{l-1})). \end{aligned} \tag{6}$$

3.4.4 Stacking SpecTNT Blocks

We stack three SpecTNT blocks for the SpecTNT module. The module starts with inputting the initial SE, $\hat{\mathcal{S}}^0$, and the initial TE, E^0, to the first SpecTNT block.

For a SpecTNT block, there are four steps. First, each FCT vector in $\hat{\mathcal{S}}^{l-1}$ is updated by adding the linear projection of the associated TE vector (Figure 3-(a)):

$$\tilde{c}_t^{l-1} = \hat{c}_t^{l-1} \oplus \mathrm{Linear}(e_t^{l-1}), \tag{7}$$

where $\mathrm{Linear}(\cdot)$ is a shared linear layer. Second, the SE $\tilde{\mathcal{S}}^{l-1}$ (with the updated FCTs $[\tilde{c}_t^{l-1}]_{t=1}^{\hat{T}}$ at the first row) is passed through the spectral Transformer (Figure 3-(c)):

$$\hat{\mathcal{S}}^l = \mathrm{SpecEnc}(\tilde{\mathcal{S}}^{l-1}). \tag{8}$$

Third, each FCT vector in $\hat{\mathcal{S}}^l$ is linearly projected and added back to the corresponding TE vector (Figure 3-(d)):

$$\tilde{e}_t^{l-1} = e_t^{l-1} \oplus \mathrm{Linear}(\hat{c}_t^l). \tag{9}$$

Finally, we propose to encode only the updated TE (i.e., $\tilde{E}^{l-1} = [\tilde{e}_t^{l-1}]_{t=1}^{\hat{T}}$), instead of TE + SE, with the temporal Transformer:

$$E^l = \mathrm{TempEnc}(\tilde{E}^{l-1}). \tag{10}$$

This operation builds up the relationship along the time axis and is the key role that leads to better model and data efficiency. We consider the temporal Transformer only needs to see the information of the frequency bins which are attended by the FCT and such design largely reduces the size of the model and also improves the performance on smaller datasets in preliminary experiments.

Task	(p_f, p_t)	(k, d)	(h_k, h_d)	o_d
Music tagging	(1, 4)	(96, 96)	(4, 8)	50
Vocal melody extraction	(4, 1)	(128, 128)	(8, 8)	$(T, 481)$
Chord recognition	(1, 1)	(64, 256)	(4, 8)	$(T, 25)$

Table 1. Settings of SpecTNT for different tasks, where p_f and p_t represent the pooling ratio we apply along the frequency and time axis in the convolutional module, k and d are the feature dimension, h_k and h_d denotes the number of heads for the spectral and temporal transformer encoder respectively, and finally, o_d represents the output dimension, T indicates frame-wise predictions.

3.5 Output Module

The output TE of the 3rd SpecTNT block, E^3, can be used towards the final output. For frame-wise prediction tasks such as vocal melody extraction and chord recognition, we feed each TE vector e_t^3 into a shared fully-connected layer with sigmoid or softmax function for final output. For song-level prediction tasks such as music tagging, we initialize a temporal class token ϵ^l ($l = 0$) concatenated at the front of E^l:

$$\hat{E}^l = [\epsilon^l, e_1^l, e_2^l, ..., e_{\hat{T}}^l], \tag{11}$$

Note that ϵ^l does not have an associated FCT in SE, but is for aggregating TE vectors along the time axis. Finally, we feed ϵ^3 to a fully-connected layer, followed by a sigmoid layer, to get the probability output.

4. EXPERIMENTS

In this section, we evaluate SpecTNT on various types of MIR tasks to demonstrate its effectiveness and versatility. We choose three MIR tasks – music tagging, vocal melody extraction, and chord recognition.

4.1 Implementation

SpecTNT is implemented using Pytorch [32]. Due to the difference in dataset sizes and the natures of tasks, we use different hyper-parameters for the tasks as shown in Table 1. All models include dropout with a rate of 0.15 in the Transformers of the TNT modules. We use AdamW [33] as the learning optimizer. The initial learning rates are set to 10^{-3} for vocal melody extraction, 5×10^{-4} for music tagging and chord recognition, and a weight decay of 5×10^{-3} is set for all the tasks.

For the input representation of music tagging, we re-sample the audio at the 22,050 Hz and use an input length of 4.54 second. Log-magnitude mel-spectrograms are computed with 128 mel filter banks, 1024 samples of Hann window, and a hop size of 512 samples. For vocal melody extraction, input waveforms are re-sampled at the 16,000 Hz sample rate. We take 3-second segments input and their log-magnitude spectrograms are computed with 2048 samples of Hann window and a hop size of 320 samples. For chord recognition, we try two types of input representation. The first input type is 24-dimensional chroma features with a frame rate of 46 ms [34]. Out of the whole

track, we use 400 frames as an input. The second input type is CQT, which is computed from a 18.2 second audio at the 22,050 Hz sample rate. The CQT includes six octaves starting from C1 (32.70 Hz) with 24 bins per octave, and is based on a hop size of 2048.

4.2 Ablations Study

To validate the design choices we make, we consider three various models by progressively removing the components of SpecTNT as follows.

A1: Remove the operation of (a) in Figure 1 (i.e., Eq. 7) and initialize the FCTs as learnable vectors.

A2: Neglect the FCTs but use the full spectral embeddings for operations (a) and (c) in Figure 1 (i.e., Eq. 7 and Eq. 9). The resulting model can be seen as using the original TNT block [10].

A3: Remove the data flow of spectral embedding, so the model is reduced to the original Transformer [30] for aggregating the input sequence in a traditional way.

In the following evaluations of different tasks, we will include the results of the three variants for comparison.

4.3 Music Auto-tagging

Datasets Million song dataset (MSD) [35] consists of one million audio previews and a subset of it has crowd-sourced music tags. Typically, a subset with the 50 most frequent tags are used with randomly split train/validation/test sets [4]. However, these tags are noisy and the random split without considering artist overlaps may cause unintended information leakage. Therefore, we take advantage of manually cleaned 50 tags from a previous work [36] and split the dataset based on artist names so that there is no overlapped artists among the training/validation/test sets. As a result, we use 233,147 tracks, of which 70%, 15%, and 15% are allocated for training, validation, and test sets, respectively. During training, we apply random data augmentation to the input waveform following the pipeline introduced in [37].

Baseline Models Two baselines methods are compared. The first is CNNSA [6], which employs a convolutional front-end and a transformer encoder to aggregate the temporal feature. The second baseline [13] uses 7-layer short-chunk CNN with residual connection, followed by a fully-connect layer for final output. This model has shown state-of-the-art performance in music auto-tagging. We utilize the original implementation of [13] to train the baseline under the same configuration as our proposed model.

Evaluation Metrics Area Under Precision Recall Curve (PR-AUC) and Area Under Receiver Operating Characteristic curve (ROC-AUC) are used.

Results The results of music auto-tagging are summarized in Table 2. SpecTNT outperforms prior state-of-the-art models in both metrics. In the ablation study, A1 performs the worst, while A2 and A3 show similar results to SpecTNT. This can be explained from the perspective of data distribution: the top 50 tags of MSD dataset are mostly related to genre and style, both of which need enough temporal information to characterize. In A1, the process of updating FCTs with TEs is removed and this may interfere the temporal information flow being shared with the spectral data and cause the performance drop. By looking into the precision scores of individual tags where SpecTNT outperforms A3, we observe that instrumental tags such as "piano" and "guitar" can benefit from SpecTNT, because they may require more spectral information to model well. This shows the benefit of adding the spectral transformer. Also, the smaller performance difference among SpecTNT, A2, and A3 indicates that the size of MSD dataset might be enough to support architectures with less prior knowledge. That is, A2 and A3 are able to sufficiently learn from MSD the useful information without further utilizing FCTs to interact with the temporal embeddings.

Method	ROC-AUC	PR-AUC
Short-chunk CNN + Res	91.55	37.08
CNNSA	91.57	37.09
SpecTNT	**92.08**	**38.62**
A1	91.92	37.85
A2	92.07	38.59
A3	92.06	38.46

Table 2. Results (in %) for automatic music tagging.

4.4 Vocal Melody Extraction

Datasets We use two datasets to train the models: MIR1K [38], which includes 1000 Chinese karaoke clips, and a 48-song subset of MedleyDB [39] that includes vocal tracks. Since the training sets are relative small, we adopt a pipeline with four steps of augmentation techniques. There is a chance for each step to be applied to a training sample: *i*) pitch-shifting by up to ± 2 semitones (with 100% chance), *ii*) replacing the original background track with a randomly selected, different background track (with 50% chance), *iii*) changing the gain of the vocal within $[-4, 2]$ dB (with 100 % chance), and *iv*) completely removing the background track (with 10% chance).

We choose three test sets for evaluation: ADC2004, MIREX05, and MedleyDB. For ADC2004 and MIREX05, we only use the samples that have melody sung by human voice. This results in 12 samples from ADC2004 and 9 samples from MIREX05. For MedleyDB, we only use the songs that have singing voice included in their "MELODY2" annotations, yielding 12 songs. The ground-truth pitches are obtained from the MELODY2 annotations within the intervals marked as "female singer" or "male singer." These 12 songs are not included in training.

Baseline Models We compare our model with two baseline models. The first baseline is the joint detection and classification model (JDC) [18] based on CRNN. We use the most representative architecture, called "Main" in [18]. The second baseline is the frequency-temporal attention

Dataset	ADC2004			MIREX05			MedelyDB		
Method	OA	RPA	VR	OA	RPA	VR	OA	RPA	VR
JDC	71.2	68.1	73.1	86.0	80.7	85.8	77.0	64.8	73.9
FTANet	71.2	69.3	72.9	89.9	86.5	91.2	**79.4**	66.0	72.0
SpecTNT	**85.3**	**85.0**	88.3	**91.7**	**90.4**	**95.2**	78.4	**77.9**	**87.4**
A1	84.8	84.2	**89.1**	90.2	88.3	94.1	77.9	75.0	85.4
A2	84.9	84.5	88.3	89.7	87.7	93.1	79.3	75.6	84.0
A3	84.5	83.7	87.2	88.9	87.2	92.0	74.5	72.7	83.1

Table 3. Results (in %) for vocal melody extraction.

network (FTANet) [19], which is a CNN-based model that employs attention mechanism along the frequency and time axis. We re-implemented JDC and FTANet using Pytorch and used the suggested hyper-parameters in [18, 19]. Both models are trained under the same configuration (e.g., data split and augmentation process) as our model.

Evaluation Metrics Overall Accuracy (OA), Raw Pitch Accuracy (RPA), and Voice Recall (VR) are adopted for evaluation. We use mir_eval library [40] to compute the performance values with a tolerance range of 50 cents.

Results Table 3 shows the results for vocal melody extraction. SpecTNT outperforms the baselines by a large margin in terms of RPA and VR. To the best of our knowledge, this is the first attempt to apply Transformers to this task and the results demonstrate its superiority over the CNN and CRNN counterparts. It is worth noting that FTANet is trained with an input representation specifically designed for pitch detection [20], but our model works well with spectrogram input. In addition, A3 shows the largest performance drop, and this demonstrates the usefulness of spectral Transformer when training on smaller data.

4.5 Chord Recognition

Datasets We use the Billboard dataset to evaluate SpecTNT for the chord recognition task. The dataset contains 890 pieces selected from the Billboard chart slots [34]. Following [27], duplicates pieces are first removed to leave 739 unique pieces in total. The official release of the dataset only comes with 24-D chroma vectors, which might be insufficient to fully demonstrate the effectiveness of SpecTNT. Therefore, we manually collected the audio files based on the provided meta-data. Due to the potential version mismatch between our audio files and that for official chord annotations, we applied dynamic time warping (DTW) [41] to validate each song. Specifically, we first replicated the chroma features of the official release using Sonic Annotator [42] on our audio files, and then calculated the alignment cost between the two versions of chromagrams for each song using DTW. We selected 462 songs with the lowest alignment costs. The songs with ID's smaller than 1000 are used for training and the remaining for testing. To augment the training data (chroma and audio), we shifted the pitches by up to ± 6 semitones. For evaluation, we adopt the "maj/min" label set with 25 classes, where 24 are major and minor triads across the 12 semitones plus an additional "no chord" class.

Method	Chroma	CQT
CR2	78.92	73.38
BTC	77.98	73.92
HT	**82.68**	-
SpecTNT	80.47	**75.62**
A1	80.10	74.83
A2	78.76	74.44
A3	77.69	74.99

Table 4. Results (in %) for chord recognition task

Baseline Models We compare to three baseline models: *i*) CR2 model from [23], which is a CRNN-based model, *ii*) a bi-directional Transformer (BTC) [26], and *iii*) Harmony Transformer (HT) [27]. BTC and HT are known to be the current state-of-the-art models for chord recognition. For CR2 and BTC, we use the official implementations with the suggested default settings for both chromagram and CQT inputs. For HT, we report the chromagram-based results in [27], since the train/test data split and data augmentation are very similar to us. We did not conduct experiments using HT with CQT input because non-trivial modifications are required for the model.

Evaluation Metrics The Weighted Chord Symbol Recall (WCSR) score is reported as evaluation metric. WCSR is the percentage of correctly identified frames and can be computed by $\frac{t_c}{t_a} \times 100(\%)$, where t_c is the duration of the correctly predicted segments, and t_a is the total duration of the test segments.

Results Table 4 shows the results for chord recognition. For "Chroma" case, the full Billboard dataset is used. For "CQT" case, the 462 songs with audio are used. From the results, SpecTNT can outperform all the baselines except HT (with chromagram input). However, HT may benefit from joint training with an additional segmentation loss, so the comparison could be unfair. Compared to BTC and CR2, SpecTNT achieves better performance for both types of input. For the ablation study, since we used less data for CQT input, A2, which is the largest model, may suffer from over-fitting and thus performs the worst.

5. CONCLUSION

We proposed SpecTNT, a novel Transformer architecture that models spectrograms along both the time and frequency axes. The introduction of FCT enables effective communication between the spectral embeddings and temporal embeddings, maximizing the benefit of Transformer encoder for flexible, local, and global modeling. In experiments, SpecTNT has demonstrated state-of-the-art performance in music tagging and vocal melody extraction and shown competitive performance in chord recognition. For future work, we plan to apply SpecTNT to other MIR tasks, such as beat tracking and structure segmentation.

6. REFERENCES

[1] F. Rigaud and M. Radenen, "Singing voice melody transcription using deep neural networks." in *17th International Society of Music Information Retrieval conference (ISMIR), New York, USA*, 2016, pp. 737–743.

[2] S. Leglaive, R. Hennequin, and R. Badeau, "Singing voice detection with deep recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 121–125.

[3] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6964–6968.

[4] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *17th International Society of Music Information Retrieval conference (ISMIR), New York, USA*, 2016.

[5] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2392–2396.

[6] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[7] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music." in *18th International Society of Music Information Retrieval conference (ISMIR), Suzhou, China*, 2017, pp. 63–70.

[8] M. Won, S. Chun, O. Nieto, and X. Serra, "Data-driven harmonic filters for audio representation learning," *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[9] A. Zadeh, T. Ma, S. Poria, and L.-P. Morency, "Wildmix dataset and spectro-temporal transformer model for monoaural audio source separation," *arXiv:1911.09783*, 2019.

[10] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *arXiv preprint arXiv:2103.00112*, 2021.

[11] P. Lamere, "Social tagging and music information retrieval," *Journal of new music research*, vol. 37, no. 2, pp. 101–114, 2008.

[12] J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," in *Sound and Music Computing Conference (SMC)*, 2017.

[13] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," *In Proc. of Sound and Music Computing (SMC)*, 2020.

[14] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE signal processing magazine*, vol. 36, no. 1, pp. 41–51, 2018.

[15] S. Kum, C. Oh, and J. Nam, "Melody extraction on vocal segments using multi-column deep neural networks." in *17th International Society of Music Information Retrieval conference (ISMIR), New York, USA*, 2016, pp. 819–825.

[16] W. T. Lu, L. Su *et al.*, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning." in *19th International Society of Music Information Retrieval conference (ISMIR), Paris, France*, 2018, pp. 521–528.

[17] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A streamlined encoder/decoder architecture for melody extraction," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 156–160.

[18] S. Kum and J. Nam, "Joint detection and classification of singing voice melody using convolutional recurrent neural networks," *Applied Sciences*, vol. 9, no. 7, p. 1324, 2019.

[19] S. Yu, X. Sun, Y. Yu, and W. Li, "Frequency-temporal attention network for singing melody extraction," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[20] L. Su and Y.-H. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 10, pp. 1600–1612, 2015.

[21] A. Jansson, R. M. Bittner, S. Ewert, and T. Weyde, "Joint singing voice separation and f0 estimation with deep u-net architectures," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[22] Y. Gao, X. Zhang, and W. Li, "Vocal melody extraction via hrnet-based singing voice separation and encoder-decoder-based f0 estimation," *Electronics*, vol. 10, no. 3, p. 298, 2021.

[23] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition." in *18th International Society of Music Information Retrieval conference (ISMIR), Suzhou, China*, 2017, pp. 188–194.

[24] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks." in *14th International Society of Music Information Retrieval conference (ISMIR), Curitiba, Brazil*. Citeseer, 2013, pp. 335–340.

[25] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *2012 11th International Conference on Machine Learning and Applications*, vol. 2. IEEE, 2012, pp. 357–362.

[26] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, "A bi-directional transformer for musical chord recognition," in *20th International Society for Music Information Retrieval Conference (ISMIR), Delft, The Netherlands*, 2019.

[27] T.-P. Chen and L. Su, "Harmony transformer: Incorporating chord segmentation into harmony recognition," *neural networks*, vol. 12, p. 15, 2019.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *31st Conference on Neural Information Processing Systems*, 2017.

[31] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 524–10 533.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[33] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[34] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An expert ground truth set for audio chord recognition and music analysis." in *12th International Society of Music Information Retrieval conference (ISMIR), Miami, USA*, vol. 11, 2011, pp. 633–638.

[35] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of International Conference on Music Information Retrieval (ISMIR)*, 2011.

[36] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada*, 2020.

[37] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *arXiv preprint arXiv:2103.09410*, 2021.

[38] C.-L. Hsu and J.-S. R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 310–319, 2009.

[39] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *15th International Society of Music Information Retrieval conference (ISMIR), Taipei, Taiwan*, vol. 14, 2014, pp. 155–160.

[40] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval conference (ISMIR), Taipei, Taiwan*. Citeseer, 2014.

[41] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[42] C. Cannam, M. O. Jewell, C. Rhodes, M. Sandler, and M. d'Inverno, "Linked data and you: Bringing music research software into the semantic web," *Journal of New Music Research*, vol. 39, no. 4, pp. 313–325, 2010.