YOURMT3+: MULTI-INSTRUMENT MUSIC TRANSCRIPTION WITH ENHANCED TRANSFORMER ARCHITECTURES AND CROSS-DATASET STEM AUGMENTATION

Sungkyun Chang[⋆] *Emmanouil Benetos*[⋆] *Holger Kirchhoff*[†] *Simon Dixon*[⋆]

[⋆] Centre for Digital Music, Queen Mary University of London [†] Huawei

2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP) | 979-8-3503-7225-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/MLSP58920.2024.10734819

ABSTRACT

Multi-instrument music transcription aims to convert polyphonic music recordings into musical scores assigned to each instrument. This task is challenging for modeling as it requires simultaneously identifying multiple instruments and transcribing their pitch and precise timing, and the lack of fully annotated data adds to the training difficulties. This paper introduces `YourMT3+`, a suite of models for enhanced multi-instrument music transcription based on the recent language token decoding approach of MT3. We enhance its encoder by adopting a hierarchical attention transformer in the time-frequency domain and integrating a mixture of experts. To address data limitations, we introduce a new multi-channel decoding method for training with incomplete annotations and propose intra- and cross-stem augmentation for dataset mixing. Our experiments demonstrate direct vocal transcription capabilities, eliminating the need for voice separation pre-processors. Benchmarks across ten public datasets show our models' competitiveness with, or superiority to, existing transcription models. Further testing on pop music recordings highlights the limitations of current models. Fully reproducible code and datasets are available with demos at `https://github.com/mimbres/YourMT3`.

Index Terms— Multi-instrument, automatic music transcription (AMT), music information retrieval (MIR), transformers, data augmentation, mixture of experts (MoE), music tokens

1. INTRODUCTION

Automatic music transcription (AMT) [1] is a fundamental task in music information retrieval where the goal is to transform music audio input into a sequence of musical notes, with each note possessing properties such as onset, offset, pitch, and sometimes velocity. The output is typically presented in the form of MIDI or piano-roll notation. The significance of AMT extends to a wide range of applications, including interactive music systems [2], automatic accompaniment generation [3], and music performance assessment.

The key challenge of this research is multi-instrument AMT: identification and transcription of various instruments with vocals from music recordings. Recently, there has been notable progress in this field: MT3 [4] utilized a MIDI-like decoding transformer, while PerceiverTF [5] employed a spectral attention transformer that generates conventional piano-roll. Unfortunately, the absence of fully reproducible code for these models has been a significant limitation for replication and further research. Our replication of MT3, trainable from scratch, is dubbed as `YourMT3` [6]. Based on this, we propose `YourMT3+`, a hybrid architecture that incorporates advanced architectures and training methods for further enhancements. `YourMT3+` and its variants differ from prior work [4, 5] in the following key aspects:

- **Enhanced Encoder**: PerceiverTF [5], which generated piano-rolls, is now trained with the MT3 framework to generate note event tokens. We replaced MT3's encoder with PerceiverTF featuring spectral cross attention (SCA). Additionally, replacing its feedforward network (FFN) with a mixture of experts (MoE) [7], denoted as `YPTF.MoE`, demonstrates promising results.
- **Multi-channel Decoder**: In addition to *General MIDI* tokens, singing transcription tokens have been further defined. We introduce a multi-channel decoder that replaces MT3's single-channel decoder [4]. This enables task-query based training and the use of partially annotated data, improving performance.
- **Augmentation**: The proposed online data augmentation framework incorporates intra-stem and cross-stem mixing across datasets and pitch-shifting. In particular, *cross-stem augmentation* allows for transcribing singing with other instruments without the need for a voice separation front-end.
- **Evaluation**: Our models were extensively validated on various multi-instrument and single-instrument datasets. One of the main applications of multi-instrument AMT can be transcribing pop music. We provide refined annotations for the existing pop music dataset [8], presenting the first study to investigate multi-instrument AMT performance on commercial pop music.

2. RELATION TO PRIOR WORK

While substantial research exists in AMT, multi-instrument transcription has recently seen significant developments. The field often faces challenges due to the scarcity of fully annotated datasets for all instruments, making it *low-resourced*. Strategies such as multi-task learning [4, 9], unsupervised learning methods [10] and iterative re-alignment techniques [11] have offered partial remedies, with most models producing piano-roll outputs at the frame level.

Compared to the conventional AMT models based on onsets and frames [12], MT3 [4] is a sequence-to-sequence model that mainly distinguished itself in decoding outputs. It decodes a note-level representation similar to language tokens derived from MIDI, deviating from the traditional frame-level piano-rolls. In Section 3.3, we discuss the advantages of using these output tokens in `YourMT3`.

The transcription of singing within multi-instrument AMT remains largely unexplored, despite potential overlaps with source separation [13] and melody extraction [14]. *PerceiverTF* [5], a model with piano-roll output, has significantly advanced the transcription

This research utilized Queen Mary's Andrena HPC facility supported by QMUL Research-IT, and the AI Industrial Convergence Cluster supported by the Ministry of Science and ICT of Korea, and Gwangju Metropolitan City. EB is supported by RAEng/Leverhulme Trust Research Fellowship LTRF2223-19-106.

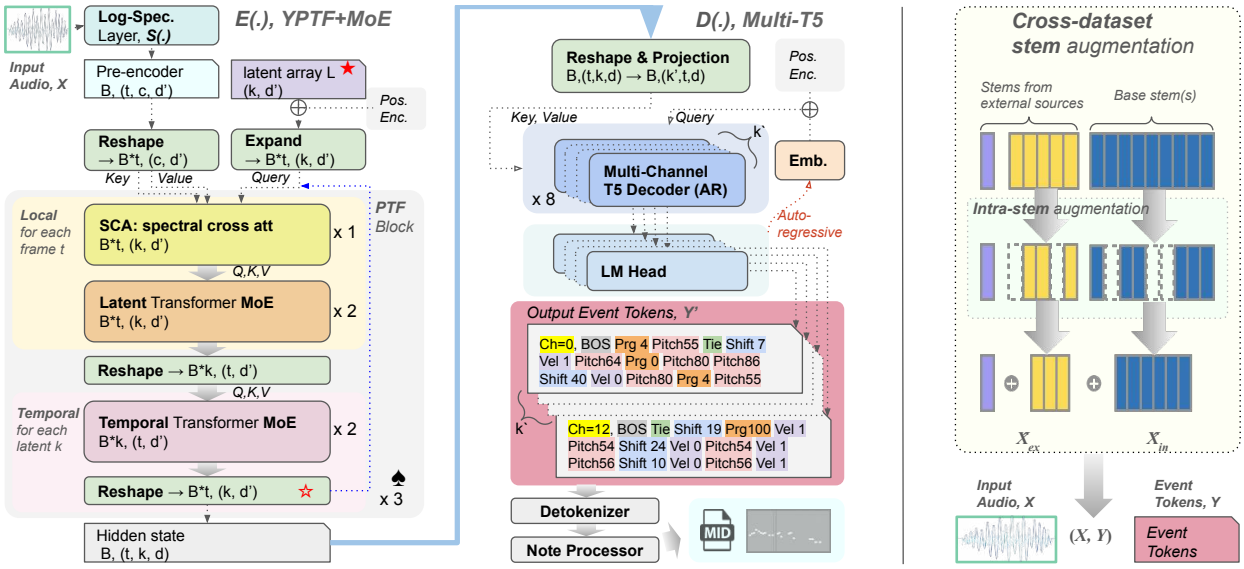

Fig. 1. Overview of *YourMT3+*. (left) Our encoder $E(\cdot)$ takes as input a log mel spectrogram S derived from audio X. (center) An autoregressive decoder $D(\cdot)$ with the language model (LM) head is conditioned by $E(S)$, and output event tokens Y'. (right) Cross-dataset stem augmentation, described in Section 4.

of multiple instruments and vocals by introducing spectral cross-attention (SCA) and stem dataset mixing. We propose an augmentation method, denoted by a plus (+) sign, that formalizes the earlier stem mixing approach [5] within an online multi-dataset pipeline.

3. MODEL

In the `YourMT3+` taxonomy, `YMT3` models match MT3's [4] architecture and training. `YPTF+Single` models use PerceiverTF (PTF) encoder with MT3's single-channel decoder and stem augmentation (+). Our empirical finding demonstrates that PTF's hierarchical attention with instrument-group sub-task queries enhances multi-instrument AMT in complex mixtures. `YPTF.MoE` replaces the encoder's FFN with mixture of experts (MoE), enabling task-specific encodings in multi-dataset training. These models efficiently process MIDI tokens instead of piano-roll. Our multi-channel decoder assigns instrument groups per channel and masks loss for unannotated instruments, allowing training with incomplete labels. The final `YPTF.MoE+Multi` model integrates all these features.

The left panel of Figure 1 provides a detailed overview of our final extended model, `YPTF.MoE+Multi`. The subsequent subsections will detail the components of our model variants, including the audio input, encoder, decoder, and output tokens.

3.1. Input

In Figure 1, X represents a 2.048-second audio segment. In `YMT3`, X is transformed into a *log-magnitude mel-spectrogram* $S \in \mathbb{R}^{t \times f}$ with 256 time steps and 512 mel-frequency bins. In `YPTF`, X is initially transformed into a *log-magnitude spectrogram* with 110 time steps and 1,024 frequency bins. Subsequently, a convolutional feature S_{conv} is produced by 2D ResNet pre-encoder [5], resulting in $S_{conv} \in \mathbb{R}^{t \times c \times f'}$, where both c and f' are set to 128. The multi-resolution input of `YPTF` mirrors *PerceiverTF* [5], including an additional channel dimension C, and differs from PerceiverTF only in the input length, using 2.048 seconds instead of 6 seconds.

3.2. Encoder

The encoder $E(\cdot)$ takes S as an input, where the last dimension of S typically matches the encoder's hidden dimension d. Our baseline encoder of `YMT3` is based on the T5-small v1.1 [15] encoder composed of 8 standard transformer blocks with 6-head self-attention and gated FFNs. The proposed `YPTF` replaces the encoder with PerceiverTF (PTF) [5] blocks as depicted in Figure 1 (left).

PTF block: Each PTF block in our model comprises local and temporal transformer sub-blocks. The local transformer first employs spectral cross attention (SCA), derived from Perceiver [16], using a learnable latent array $L \in \mathbb{R}^{k \times d'}$ and S_{conv} as inputs. Here, k is typically set to twice the number of target instrument groups, where $k < c$ and specifically $k = 26$ for 12 instruments plus singing, with each pair of latents serving as a *query* for the corresponding instrument groups. The latent and temporal transformer sub-blocks, featuring 8-head self-attention, FFNs and residual connections for queries, differ functionally: the former processes spectral information independently of time t, by attending to k and c, whereas the latter handles only temporal information relevant to t and d, independent of k. Overall, the PTF block (♠, Figure 1) performs three iterations. Initially, ★ acts as the *query* in SCA during the first iteration. In the second and third iterations, ☆ serves as the *query*.

MoE: `YPTF.MoE` models replace FFNs in latent and temporal transformer blocks with MoE layers [7], routing attention to two of eight experts. Using two experts gave better results than one or four; see Supplemental B.5. In our experiments, MoE increased the model complexity by about 5% while improving performance across various datasets. Unlike PerceiverTF, we use RoPE [17] in every sub-block of the encoder to integrate positional information through rotation matrices, replacing trainable position embedding (PE), and pre-LayerNorm with pre-RMSNorm. However, these modifications only offered minor benefits in memory and computation without significantly impacting performance.

3.3. Output Tokens

The center panel of Figure 1 shows the output sequence Y' with a maximum N time steps, and the tokens representing MIDI-like events are listed in Supplemental F. As noted in Section 5.2, models trained with more fine-grained vocabulary consistently perform better. Therefore, we use `MT3_FULL_PLUS` for training and `MT3_MIDI_PLUS` only for comparison tests with previous work. Following the note sequence structure in MT3 [4], we made two modifications to the MT3 tokens: (a) unused velocity tokens, except 0 and 1, were removed, and (b) programs 100 and 101 were reserved for singing voice (melody) and singing voice (chorus), respectively.

Compared to traditional piano-rolls [12, 18, 10, 9, 5], MIDI-like tokens [4] offer several advantages: they are more memory-efficient by representing note onset, shift, and offset with tokens rather than hundreds of frames; they simplify multi-instrument data handling by expanding the program vocabulary without significant memory increase, whereas piano-rolls need large separate matrices for each instrument; and they explicitly represent linked note onsets and offsets, avoiding extra post-processing required for piano-rolls.

3.4. Decoder

We use an auto-regressive decoder $D(\cdot)$, conditioned on the encoder's last hidden state, to generate note sequences. The baseline decoder, based on T5-small v1.1 and denoted as `Single`, produces a single sequence with events from multiple instruments.

When annotations are available for only one or some instruments in the audio, we need to mask the loss for unannotated instruments. The `Single` decoder's output blends multiple programs, making it hard to mask specific instruments due to token dependencies. To address this, we propose a `Multi` decoder. It can provide separately maskable supervision for each latent L of the PTF encoder, allocated into channels for each program group.

In our implementation, the PTF encoder's output hidden states are grouped by allocating two latents per channel—with group-linear projection, $k = 26$ latents result in $k' = 13$ projected channels. The `Multi` decoder then independently decodes each of the k' inputs, producing k' sequences for each program using parallel decoders with shared parameters. We set the maximum sequence length to $N_{\text{single}} = 1024$ (as in MT3 [4]) and $N_{\text{multi}} = 256$. Potential truncation loss is discussed further in Supplemental B.6.

4. DATA AUGMENTATION

This section describes an augmentation method for training with multiple datasets. Our strategy is to maximize the diversity of the training examples by randomly mixing selected stems from across multiple datasets. *Intra-stem augmentation* described in Section 4.1 involves selectively muting stems within a multi-track recording to generate several variations, as demonstrated with MT3 [4] and the Slakh dataset. The concept of *cross-dataset stem augmentation*, as discussed in Section 4.2, draws inspiration from PerceiverTF [5]. It aims to create a new mixture of stems from multiple datasets. Additionally, we employ pitch-shifting as described in Section 4.3.

4.1. Intra-stem Augmentation

This refers to the process of randomly dropping instruments from a segment containing multiple stems. From any dataset we sample X, a 2.048-second segment starting from a random point. Assuming that X is composed of N stems denoted $x_1, x_2, \ldots, x_N$, we define a set $\hat{X}_{\text{in}}$ of randomly selected or dropped stems as:

$$\hat{X}_{\text{in}} = \{x_i : x_i \in X, \text{ with } x_i \sim \text{Bernoulli}(p)\} \tag{1}$$

with $i \in \{1, 2, ..., N\}$ where $N > 1$. Here, p=0.7 by default, is the probability of each stem being selected. Each x_i is chosen with p, creating $\hat{X}_{\text{in}}$ with various combinations of stems from X. A larger p increases active stems and task difficulty. The sweet spot was between 0.6 and 0.8, increasing with model size and training time.

Algorithm 1 Cross-dataset Stem Augmentation

Require: $X, U, L, J, \Psi, \tau, p$ {
X: A segment $X \in U$, with stems $x \in X$.
U: Cached segment batches from various datasets.
L: Maximum length of sequence. 1,024 by default.
J: Maximum number of iterations w.r.t j. 5 by default.
Ψ: Stem mixing policy.
τ: Exponential decay parameter. 0.3 by default.
p: Probability for intra stem selection. 0.7 by default. }

1: $\hat{X}_{\text{in}} \leftarrow x_i : x_i \in X$, selected with $x_i \sim \text{Bernoulli}(p)$
2: $\hat{X}_{\text{ex}} \leftarrow \varnothing$
3: $j \leftarrow 0$
4: **while** $r \sim \textit{Uniform}(0,1) < e^{-\tau j}$ and $|\hat{X}_{\text{ex}}| < L$ and $j < J$ **do**
5: $X' \leftarrow$ a randomly sampled segment from $U \setminus X$
6: $X' \leftarrow \text{Filter}(X'; \Psi)$ // retain stems meeting criteria
7:
8: **if** $X' \neq \varnothing$ **then**
9: $\hat{X}_{\text{ex}} \leftarrow \hat{X}_{\text{ex}} \cup X'$ // add stems
10: $j \leftarrow j + 1$
11: **end if**
12: **end while**
13: $\hat{X} \leftarrow \hat{X}_{\text{in}} \cup \hat{X}_{\text{ex}}$
14: $\text{Mix}(\hat{X})$ // apply stem mixing

4.2. Cross-dataset Stem Augmentation

Procedure: In Algorithm 1, we designate U as a collection of cached segment batches across diverse datasets, with its size required to be at least equal to the batch size and preferably larger, if permitted by memory constraints. The base segment X is a sampled segment from U, and the elements of X are stems denoted by x. Here, x signifies a stem ID, including related token and audio information.

Intra-stem augmentation is first applied to X as in Equation 1, yielding a processed base segment $\hat{X}_{\text{in}}$. Next, we enter a loop to mix the base stems of $\hat{X}_{\text{in}}$ with the stems coming from other segments. $U \setminus X$ represents the set of all segments in U excluding X. Each iteration begins by randomly sampling a segment X' from $U \setminus X$. Stems in X' that do not satisfy policy Ψ (detailed in Supplemental D.2) are then filtered out. Subsequently, $\hat{X}_{\text{ex}}$ is updated by merging X'. This loop persists until at least one stopping criterion described in the following subsection is satisfied. Once the aggregation is complete, the Mix$(\cdot)$ function executes the actual mixing of tokens and audio content in a batch-wise manner.

Stopping criteria In Line 4 of Algorithm 1, three criteria are established to stop the iterative mixing among stems. The first criterion is an exponential decay $S(j)$ that serves as the survival function defined as $S(j) = e^{-\tau j}$, where τ controls the surviving curve with respect to j-th iteration. The second criterion restricts $\hat{X}_{\text{ex}}$ to a length L, measured as sequence length post-tokenisation. The last criterion, $j > J$ with $J = 5$ allows mixing up to 5 segments per base segment.

Train	*Test*
MusicNet-EM, GuitarSet, **MIR-ST500**, **ENST-Drums**, **Slakh**, EGMD, Maestro, CMedia, **URMP**, SMT-Bass	**MusicNet**, **MusicNet-EM**, GuitarSet,**MIR-ST500**, **ENST-Drums**, **Slakh**, Maestro, MAPS, **URMP**, **RWC-Pop (refined)**

Table 1. Summary of datasets for train/test. Multi-instrument datasets with full annotation and stems are highlighted in light blue, while those with partially annotated instruments are highlighted in pink. (refined) We offer updated annotations for RWC-Pop [8].

4.3. Pitch-shifting

We apply GPU-based phase vocoder pitch-shifting adapted from `TorchAudio`[1] after cross-dataset stem augmentation, using default settings except nFFT=512 for time-stretching. Batch elements are randomly assigned to five groups, each shifted by -2, -1, 0, +1, or +2 semitones. Notably, as will be discussed in Section 5.2, pitch shifting's inconsistent benefits across datasets were resolved by MoE models' increased capacity.

5. EXPERIMENTS

5.1. Experimental Setup

Data Preparation: Table 1 lists the datasets used for training and evaluating our model. We offer a software package for dataset setup and split information to ensure reproducibility of our results. Audio data was converted into 16 kHz mono WAV format. Stems were stored as arrays, and mix-tracks as WAV files, also treating stemless tracks as mix-tracks. For training our `Single` decoder models on MIR-ST500 [19] and CMedia [20] , we produced singing and accompaniment stems using a pre-trained separation model [13]. With the `Multi` decoder, we also incorporated the original mix tracks from these datasets.

Evaluation Metrics: To evaluate transcription accuracy for each instrument, we employ the *Instrument Note Onset F1* metric [5]. This metric, valid for any instruments including drums, requires matching the onset, pitch, and instrument to the reference within a tolerance of ±50 ms. For multiple non-drum instruments, we additionally utilize the *Instrument-Agnostic Onset F1 and Offset F1* necessitating exact matches for only onset or both onset and offset. These metrics parallel the standard *Note F1* metrics [21] for single-instrument datasets. Furthermore, we used the *Multi (instrument offset) F1* metric [4] for evaluating multi-instrument AMT systems, where correct predictions require matching onset-offset pairs, pitch, and instrument type, excluding drum offsets. Our Multi F1 metric is notably more stringent than the Multi Onset F1 reported for PerceiverTF [5].

Vocabulary: Our models were trained using `MT3_FULL_PLUS` and tested on `MT3_MIDI_PLUS`, detailed in Section F of the Supplemental Document. Despite testing exclusively with the `MIDI` vocabulary, results in Table 3, labeled *+full vocab*, show that training with the more fine-grained `FULL` vocabulary enhanced performance compared to training and testing solely with `MIDI`.

Training: Our models were trained with two NVIDIA A100 GPUs using BFloat16 mixed-precision. In the implemented online data pipeline, four CPU processes per GPU were allocated to efficiently load and augment data without causing streaming bottlenecks. In our preliminary experiments, we tested three optimizers at a constant learning rate of 1e-03: AdaFactor [30], AdamW [31], and AdamWScale [32]. AdamWScale, a variant of AdamW that normalizes gradients using root-mean-square (RMS) energy, provided the most efficient training. Our models were trained using AdamWScale and a cosine scheduler for 300K steps, with initial and final learning rates of [1e-02, 1e-05] and a 1,000-step warm-up from 1e-03. We set the dropout rate at 0.05.

[1] `https://pytorch.org/audio`

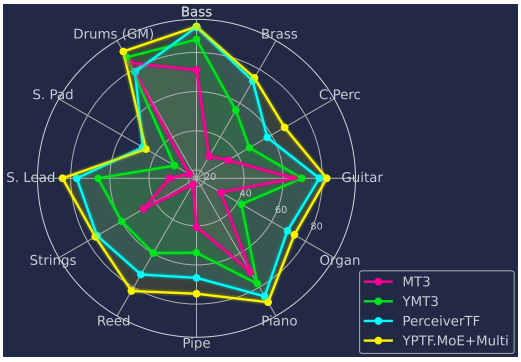

Fig. 2. Instrument Onset F1 on Slakh [29].

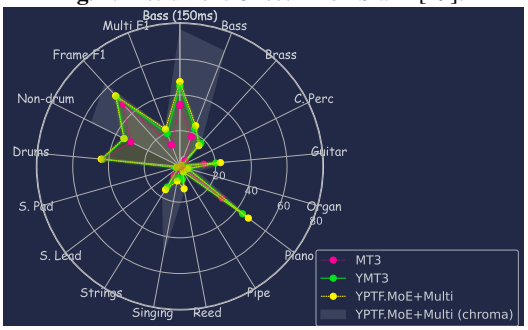

Fig. 3. Instrument-Onset/Frame/Multi F1 on RWC-Pop [8].

5.2. Results and Discussion

In Table 2, our models are compared with other state-of-the-art models across datasets. From MAPS to GuitarSet, evaluations use Instrument Note Onset F1, while URMP and Slakh are assessed using Instrument-agnostic Note Onset F1 and Multi F1. Due to space constraints, only the top-performing baselines (*) are listed on the table's rightmost column. Details of all models are available in our project repository.

Our models prefixed by `Y-` outperformed MT3 [4] across all datasets. Notably, our models and the unseen baseline [23], trained without MAPS [22], outperformed the baseline [24] trained on MAPS. This is likely due to the Maestro [33] dataset being about nine times larger, providing significantly more in-domain knowledge. Among our models, `YPTF.MoE+Multi` matched or exceeded the performance of the latest baseline models in most datasets. It showed exceptional performance on both refined and unrefined datasets in MusicNet strings, particularly in tests with refined labels (EM [11]). However, a noticeable under-performance was observed in singing transcription compared to the baseline [5]. As evidenced by about 10% higher F1 on the MIR-ST500 (100ms), many onset timing errors exceeded the acceptable 50ms range and fell within 100ms. Given that our model and the baseline [5] share

Test Set	*Instrument*	YMT3	YMT3+	YPTF+S	YPTF+M	YPTF.MoE+M	*MT3 [4]*	*AMT*
		noPS	*noPS* | *PS*	*noPS* | *PS*	*noPS* | *PS*	*noPS* | *PS*	(colab)	Baseline *
MAPS [22] (unseen)	Piano	81.44	85.92 | 87.73	88.37 | **88.73**	87.84 | 86.88	87.88 | 86.25	80.62	88.40 [23]♣
MAPS [22] (seen)		-	-	-	-	-	-	85.14 [24]♣
Maestro v3		94.78	94.80 | 94.31	96.28 | 95.85	95.59 | 94.54	96.98 | 96.52	94.86	**97.44** [24]♣
MusicNet ext.	Strings	81.69	89.04 | 88.34	88.39 | 89.39	88.52 | 87.04	**91.32** | 90.07	-△	80.00 [11] ♯
(EM) [11]	Winds	74.95	82.91 | 80.53	77.72 | 79.59	77.18 | 76.54	83.46 | 78.50	-△	**85.50** [11] ♯
MusicNet ext.	Strings	58.20	64.67 | 63.94	64.63 | 65.40	64.17 | 64.08	**66.14** | 66.09	-△	63.90 [11] ♯
[10, 11]	Winds	50.76	55.58 | 55.05	52.55 | 54.27	51.82 | 51.42	55.95 | 55.33	-△	**60.90** [11] ♯
MIR-ST500 [19] (SVS)		67.98	70.39 | 70.69	70.82 | 70.56	71.07 | 71.32	71.60 | **72.05**	-◊	70.73 [25]
MIR-ST500 [19]	Singing	3.62	64.03 | 65.69	66.75 | 67.11	69.67 | 70.26	70.59 | 71.07	-◊	**78.50** [5]
MIR-ST500 (100ms [20])		3.64	71.15 | 72.08	73.26 | 73.89	79.29| 80.63	81.14| **82.08**	-◊	-
ENSTdrums (DTP [26])	Drums	87.77	87.60 | 87.40	89.72 | **90.65**	88.68 | 90.61	88.79 | 89.48	77.82	84.50 [26] ♣
ENSTdrums (DTM [26])		78.64	81.84 | 83.09	85.65 | 86.41	85.14 | 87.18	85.92 | **87.27**	70.31	79.00 [26] ♣
GuitarSet [27] (MT3 [4])	Guitar	88.53	91.39 | 88.49	91.61 | 88.32	88.92 | 86.74	**91.65** | 88.87	89.10	91.10 [5]
URMP [28] Onset F1 [4]	Agnostic	77.10	80.00 | 81.47	81.11 | 81.54	74.56 | 75.72	81.05 | **81.79**	76.65	77.0 [4]
URMP [28] Multi F1 [4]	Ensemble	58.23	62.13 | 62.03	64.34 | 65.89	57.25 | 59.82	67.22 | **67.98**	58.71	59.0 [4]
Slakh [29] Onset F1 [4]	Agnostic	64.83	77.96 | 75.28	80.70 | 76.32	79.39 | 75.68	84.14 | **84.56**	75.20	81.9 [5]
Slakh [29] Multi F1 [4]	All	61.77	65.92 | 63.61	69.52 | 65.13	69.37 | 64.96	73.98 | **74.84**	57.69	62.0 [4]♡

Table 2. Dataset-wise Note Onset F1. *PS* and *noPS* represent training with and without pitch shifting augmentation, respectively. (EM) denotes evaluation using refined labels [11]. (SVS) refers to experiments using singing separated audio as input, obtained through *Spleeter* [13]. (DTP) represents using drum and percussion as input. (DTM) uses input including drum, percussion, and accompaniment. The Onset F1 score on Slakh is instrument-agnostic F1 for non-drum classes. (△) Unavailable due to training split overlaps. (♣) Single-instrument AMT. (◊) Singing voice class was not defined. (♯) Additionally collected synthetic data from 8.5K songs were used for pre-training [11].

Model	Onset F1	Offset F1	Drum F1
YMT3 base	64.8	41.7	77.8
+ *Intra-aug.*	**+4.8**	**+5.5**	+0.6
+ *Full-vocab.*	+0.6	+2.1	+2.6
+ *Data balancing*	**+ 4.0**	**+4.7**	+1.3
+ *Cross-aug.*	**+4.0**	**+7.2**	+1.6
+ *PTF-encoder*	+1.8	**+4.2**	+1.9
+ *FFN → MoE*	+1.5	+1.3	+3.7
+ *Multi decoder*	+1.8	**+4.0**	+0.6
YPTF.MoE+Multi	**84.6**	**70.7**	**90.1**
MT3 (colab)	75.2	56.8	83.9
MT3 [4]	76	57	-
PerceiverTF [5]	81.9	-	78.3

Table 3. Model component analysis and comparison on the Slakh [29] dataset. (-) Values not reported.

similar encoder structures, our decoder may be more prone to timing errors than traditional piano-roll models. Additionally, the practicality of a 100ms onset tolerance, used in past MIREX [20] singing transcription protocol, appears justified.

YMT3+ and YPTF+Single differ only in their encoders. This comparison revealed that the PTF encoder architecture performs particularly well in complex multi-instrument datasets such as MIR-ST500, ENSTdrums (DTM), and Slakh. *Cross-stem augmentation*, denoted by the (+) symbol in model names, proved essential for transcribing singing without singing voice separation (SVS). YMT3 recorded an F1 score of 3.6% without separation, while YMT3+ with augmentation reached 64%. The models with Multi decoders were beneficial when training on partially annotated datasets, such as MIR-ST500 and ENSTdrums. *Mixture of Experts (MoE)* showed consistent performance improvements across all datasets. Notably, while pitch-shifting often led to performance degradation in other models, YPTF.MoE compensated for this loss or even improved performance, as evidenced by the Slakh result.

As compared in the lower section of Table 2, YPTF.MoE+Multi significantly outperformed the baselines (MT3 [4] and PerceiverTF [5]) on multi-instrument datasets such as URMP and Slakh. The baseline Multi F1 score marked with a ♡ is from MT3 authors' report [4]. For the complete comparison table with MT3 [4] and PerceiverTF [5], see Section H of the Supplemental Document.

Ablation Study: In Table 3, the impact of each model component on performance was investigated. Both intra- and cross-stem augmentations significantly improved performance by over 4 percentage points, while all other proposed components steadily enhanced transcription performance. Additionally, the performance improvement denoted by *Data balancing* suggested that previously adopted temperature-based sampling in MT3 [4] might not be suitable for determining the sampling probability of AMT datasets. This is further discussed in Section F of the Supplemental Document.

Performance on Pop Music: As seen at the bottom of Table 3, our model demonstrated competitive performance on the synthetic dataset [29] compared to other multi-AMT models. In Figure 2, our final model achieved 50 to over 90% performance for most instruments, except for a few non-mainstream ones like chromatic percussion (c. perc) and synth pad (s.pad) in the synthetic dataset. However, *a significant limitation emerged in its performance on commercial pop music recordings*, as shown in Figure 3. Particularly for non-main instruments (excluding piano, bass, vocals, and drums), our models performed below 10%. This suggests potential biases introduced by training primarily on synthetic datasets, which may not fully cover the diverse timbres of pop music. Furthermore, except for the piano, all the pitched instruments showed a significant gap in the chroma-level metric, suggesting substantial octave errors and hinting that more varied pitch-shifting could be beneficial.

6. CONCLUSION AND FUTURE WORK

This work presented YourMT3+, a hybrid model suite that combines MT3 and PerceiverTF features. Our final model, YPTF.MoE+Multi, employed spectral cross-attention and a Mixture of Experts in its encoder for enhanced performance, and a multi-channel decoder to handle the instruments where annotation is partially available. Our models trained using the proposed online augmentation strategy demonstrated direct vocal transcription capabilities without the need for a singing separation front-end. The final model significantly outperformed MT3 and PerceiverTF on the multi-AMT benchmark with a parameter increase of less than 2.5% compared to MT3. Evaluations across ten public datasets also validated our model's competitiveness. Despite progress, challenges persist: onset timing in singing voice transcription lags behind our baseline, and low performance in pop music may stem from reliance on synthetic datasets for diverse instruments. Future research will address these issues.

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, 2018.

[2] R. Rowe, *Machine musicianship*, MIT press, 2004.

[3] G. Percival, S. Fukayama, and M. Goto, "Song2quartet: A system for generating string quartet cover songs from polyphonic audio of popular music.," in *ISMIR*, 2015, pp. 114–120.

[4] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *ICLR*, 2021.

[5] W.-T. Lu, J.-C. Wang, and Y.-N. Hung, "Multitrack music transcription with a time-frequency perceiver," in *ICASSP*, 2023.

[6] S. Chang, S. Dixon, and E. Benetos, "YourMT3: a toolkit for training multi-task and multi-track music transcription model for everyone," in *DMRN+17*, 2022.

[7] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, et al., "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[8] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases.," in *ISMIR*, 2002, vol. 2, pp. 287–288.

[9] Y.-T. Wu, B. Chen, and L. Su, "Multi-instrument automatic music transcription with self-attention-based instance segmentation," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 28, pp. 2796–2809, 2020.

[10] K. W. Cheuk, D. Herremans, and L. Su, "Reconvat: A semi-supervised automatic music transcription framework for low-resource real-world data," in *ACM Multimedia*, 2021.

[11] B. Maman and A. H. Bermano, "Unaligned supervision for automatic music transcription in the wild," in *ICML*, 2022.

[12] 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 *ISMIR*, 2018, pp. 50–57.

[13] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *JOSS*, vol. 5, no. 50, pp. 2154, 2020.

[14] T.-H. Hsieh, L. Su, et al., "A streamlined encoder/decoder architecture for melody extraction," in *ICASSP*, 2019.

[15] C. Raffel, N. Shazeer, A. Roberts, et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, vol. 21, no. 1, pp. 5485–5551, 2020.

[16] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira, "Perceiver: General perception with iterative attention," in *ICML*, 2021, pp. 4651–4664.

[17] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, pp. 127063, 2024.

[18] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," in *ICLR*, 2016.

[19] J.-Y. Wang and J.-S. R. Jang, "On the preparation and validation of a large-scale dataset of singing transcription," in *ICASSP*, 2021, pp. 276–280.

[20] "MIREX Singing Transcription from Polyphonic Music," https://www.music-ir.org/mirex/wiki/2020:Singing_Transcription_from_Polyphonic_Music, 2020, [Accessed 01-Apr-2024].

[21] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "Mir_eval: A transparent implementation of common mir metrics.," in *ISMIR*, 2014.

[22] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2009.

[23] D. Edwards, S. Dixon, E. Benetos, A. Maezawa, and Y. Kusaka, "A data-driven analysis of robust automatic piano transcription," *IEEE Signal Processing Letters*, 2024.

[24] K. Toyama, T. Akama, Y. Ikemiya, Y. Takida, W.-H. Liao, and Y. Mitsufuji, "Automatic piano transcription with hierarchical frequency-time transformer," in *ISMIR*, 2023.

[25] X. Gu, W. Zeng, J. Zhang, L. Ou, and Y. Wang, "Deep audio-visual singing voice transcription based on self-supervised learning models," *ArXiv*, vol. abs/2304.12082, 2023.

[26] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, "A review of automatic drum transcription," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, 2018.

[27] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "Guitarset: A dataset for guitar transcription," in *ISMIR*, 2018.

[28] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Trans. on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

[29] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh," in *IEEE WASPAA*, 2019.

[30] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *ICML*, 2018, pp. 4596–4604.

[31] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[32] P. Nawrot, "NanoT5: Fast & simple pre-training and fine-tuning of t5 models with limited resources," in *EMNLP Workshop for NLP-OSS*, 2023.

[33] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, et al., "Enabling factorized piano music modeling and generation with the maestro dataset," in *ICLR*, 2018.