# SCALING POLYPHONIC TRANSCRIPTION WITH MIXTURES OF MONOPHONIC TRANSCRIPTIONS

**Ian Simon** **Josh Gardner*** **Curtis Hawthorne**
**Ethan Manilow*** **Jesse Engel**
Google Research, Brain Team
{iansimon, fjord, jesseengel}@google.com
jpgard@cs.washington.edu, ethanm@u.northwestern.edu

## ABSTRACT

Automatic Music Transcription (AMT), in particular the problem of automatically extracting notes from audio, has seen much recent progress via the training of neural network models on musical audio recordings paired with aligned ground-truth note labels. However, progress is currently limited by the difficulty of obtaining such note labels for natural audio recordings at scale. In this paper, we take advantage of the fact that for monophonic music, the transcription problem is much easier and largely solved via modern pitch-tracking methods. Specifically, we show that we are able to combine recordings of real monophonic music (and their transcriptions) into artificial and musically-incoherent mixtures, greatly increasing the scale of labeled training data. By pretraining on these mixtures, we can use a larger neural network model and significantly improve upon the state of the art in multi-instrument polyphonic transcription. We demonstrate this improvement across a variety of datasets and in a "zero-shot" setting where the model has not been trained on any data from the evaluation domain.

## 1. INTRODUCTION

The variety of sounds that can appear in a musical recording is effectively infinite. Numerous instruments, articulation styles, dynamics, recording conditions, synthesizer parameters, processing effects, and more can all interact to produce the enormous space of sounds that can be heard in recorded music. This diversity of sound is a challenge for Automatic Music Transcription (AMT), the task of extracting a symbolic representation from a musical recording. In this paper we focus on the specific task of automatically extracting a symbolic representation consisting of musical *notes*. For each note we would like to infer its pitch, absolute timing of its onset and offset in seconds, and the instrument that played the note. We do not attempt to infer a musical *score* with time signatures, key signatures, etc., merely a sequence of notes with absolute timestamps that can be represented as MIDI [1].

Most recent progress in AMT has been driven by neural networks trained on musical recordings paired with their note transcriptions. However, this progress is currently bottlenecked by the limited number of existing ground truth transcriptions, which is in turn bottlenecked by the difficulty of creating note transcriptions that are precisely aligned with audio. Creating ground truth transcriptions manually is possible only for trained musicians, and is a notoriously tedious process. Furthermore, trained musicians rarely annotate the timing of note events to the precision needed for training AMT systems.

Besides being limited in number, existing transcribed recordings are also limited in *character*, as only a few methods are able to produce such aligned data efficiently:

**Capture** a musical performance using an instrument equipped with sensors, e.g. a Disklavier [2] or guitar equipped with hexaphonic pickup [3]. This yields highly accurate ground-truth transcriptions but is difficult to scale across many instruments due to the difficulty of needing to equip each instrument with specialized sensors. In addition, data created in this way is likely to be limited to a small number of recording environments, as it is easier to invite multiple performers to record in a single location than to create many sensor-equipped instruments, distribute them to performers, and ask the performers to share recordings with captured transcriptions.

**Synthesize** a sequence of notes using a software synthesizer [4, 5]. While this technique produces high-quality audio-label alignment, the space of sounds that can be generated by a synthesizer only partially covers the space of instrument sounds an AMT system is expected to transcribe. In particular, for instruments such as violin and saxophone where the sound of each note and transition is mediated by a large number of control parameters (typically "provided" by a human performer via body positioning), we do not yet have good methods for generating music that matches the realism and diversity of human performances.

**Align** an audio recording and its corresponding symbolic score using dynamic time warping [6]. This technique

* Work done as a Google Brain Student Researcher.

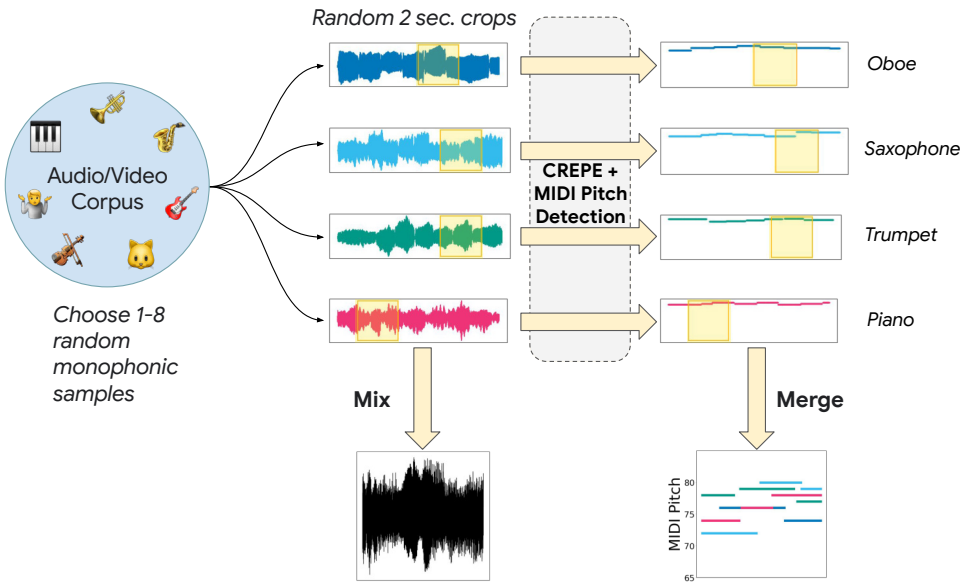


**Figure 1**. Overview of our dataset generation pipeline. Monophonic recordings are gathered from an audio/video corpus. Each training example selects 1-8 20 second clips, and then takes random ∼2 second crops from each clip for MIDI pitch detection. The resulting audio clips are mixed and MIDI clips are merged to form a polyphonic mixture for training.

has the advantage of being applicable to pre-existing audio recordings where a score is available. However, the resulting labels often end up poorly aligned due to dynamic time warping errors or when the recording does not exactly match the provided score.

In this paper, we demonstrate a new technique for creating training data for AMT models: transcribing in-the-wild monophonic (i.e. one note playing at a time) recordings and mixing the audio and transcriptions together as training data for a neural network. Monophonic recordings are much easier to transcribe due to the existence of highly accurate pitch trackers such as CREPE [7]. This technique does not suffer from any of the limitations described above; it can be applied to any monophonic audio clip, and the audio and note labels are accurately aligned.

By generating a large number of polyphonic mixes of transcribed monophonic recordings, we are able to greatly increase both the *quantity* and *diversity* of data used to train multi-instrument polyphonic transcription models. An important part of scaling our transcription data is that we mix recordings without regard to whether or not they "go together"; the mixtures are rhythmically and harmonically incoherent. Nonetheless, using these recordings allows us to train larger and better models that surpass the existing state of the art in multi-instrument AMT.

## 2. RELATED WORK

### 2.1 Automatic Music Transcription

The field of automatic music transcription (AMT) has a rich history in MIR, but has been advancing rapidly in recent years as AMT models begin to take advantage of deep learning models and large-scale datasets. For example, the Onsets & Frames model by Hawthorne et al. [2] uses a deep convolutional and recurrent neural network to jointly predict note onsets and frames (subsequently treated as an adversarial task by Kim & Bello [8]); Kelz et al. [9] use a convolutional network to model ADSR envelopes in piano transcription; Manilow et al. [10] perform simultaneous transcription and source separation with a multiheaded network; Cheuk et al. [11] use semi-supervised learning to improve transcription on low-resource datasets and later explore the effect of adding a spectrogram reconstruction loss [12], and recently Maman & Bermano [13] demonstrate the use of synthetic data and iterative alignment to enable training on weakly-aligned scores.

The unsupervised pretraining of large neural sequence architectures has been critical to scaling performance in natural language processing (NLP) [14], computer vision [15], and automatic speech recognition (ASR) [16], but the dynamics of pretraining are only beginning to be empirically well-understood [17–19]. Recently some AMT systems have adopted the same sequence architectures used in NLP (e.g. MT3 [20]), but using *pretraining* in combination with these architectures for AMT is a relatively unexplored area. While in this paper we only examine the effects of large scale pretraining on MT3, the data strategies presented are orthogonal to these improvements in model architectures and can ideally work in concert to create better overall AMT systems.

### 2.2 Dataset Mixing and Labeling Heuristics

Heuristically-labeled and randomly-mixed data have been used to improve large deep learning models in other tasks and domains. For example, data augmentation strategies that combine existing labeled examples are common in computer vision [21–23]; such strategies have been shown to reduce memorization of corrupt labels [21],

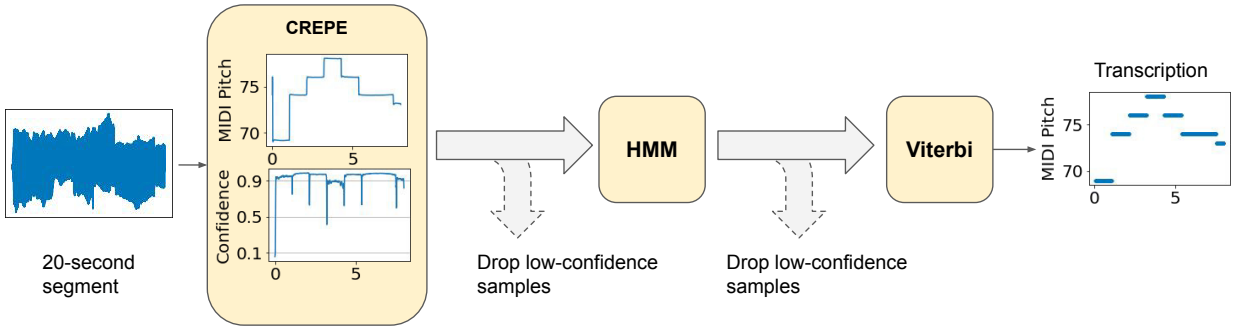


**Figure 2**. Overview of pitch tracking pipeline. HMM parameters are described in Section 3.3.

improve prediction and robustness on out-of-distribution data [21, 24], stabilize training of generative models [21], and reduce supervised models' overconfidence on samples outside the training distribution [25]. In the domain of AMT, Callender et al. [26] use data mixing to recombine random crops of drum audio samples and find that it improves a drum transcription model.

Heuristic techniques have also been used for labeling data and training MIR models. For example, Salamon et al. [27] use an analysis-synthesis framework for $f_0$ annotation where the model predictions are synthesized and mixed back into the input audio, and show that this produces results that are statistically indistinguishable from models trained on the manually-annotated mixes. Maman and Bermano [13] use dynamic time warping to generate music transcription labels from synthesized audio. In ASR, Fonseca et al. [28] show that unsupervised representation learning (via contrastive learning) can rival state-of-the-art models without expensive human-labeled datasets for audio scene separation. Furthermore, training with incoherently mixed single-instrument recordings, as we do here, is a widespread technique used for training music source separation systems [5, 29–32]. However, we are unaware of any systems that use heuristically-labeled *and* randomly-mixed data to scale AMT systems.

## 3. MODEL AND TRAINING PROCESS

### 3.1 Model Architecture

For all experiments, we use an encoder-decoder Transformer [33] architecture. Our model and training process is almost identical to that of Gardner et al. [20], with two main exceptions. First, while we use the T5.1.1 [1] [14] "Small" architecture used in [20] for some experiments, for others we use the larger T5.1.1 "Base" model.

Second, we pretrain our model on a large dataset consisting of mixes of automatically-transcribed monophonic recordings drawn from an Internet-scale pool of videos; this is the main contribution of the current work. In our experiments, we vary the number of pretraining and fine-tuning steps for the model as shown in Table 1. We share our code publicly at `https://github.com/magenta/mt3`, along with a checkpoint pretrained on our dataset of monophonic mixes.

[1] `https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#t511`

### 3.2 Data Representation

We use the exact input and output representation of Gardner et al. [20]: log-Mel spectrograms as input and a MIDI-like output vocabulary containing time, pitch, note on/off, instrument, "tie" section token, and drum tokens. This output vocabulary is capable of representing arbitrary multi-instrument polyphonic MIDI. As in Gardner et al. [20], we ignore note *velocities* as they are not present in most ground-truth transcriptions.

Again following Gardner et al., we split the input audio (and target labels at training time) into segments of 2.048 seconds (256 spectrogram frames at a hop size of 8 ms) [34]. At inference time, we transcribe each segment independently and concatenate their transcriptions. We also use the "tie" representation of Gardner et al. which requires the model to declare all notes that are active at the beginning of each segment. At inference time, if the model fails to declare a note that was active in the previous segment, we turn off the note.

### 3.3 Monophonic Detection and Transcription

We process a dataset of music recordings obtained in the wild to (a) detect clips that are likely to be monophonic, and (b) transcribe those clips into sequences of notes. While our heuristic process is far from the ideal approach to monophonic transcription, its main benefit is that it is simple and easy to apply to an Internet-scale data corpus.

Our process of creating a set of transcription labels from unlabeled audio data is illustrated in Figure 2, and proceeds as follows. First, we split each recording into non-overlapping 20-second segments. Then, we use the CREPE [7] pitch tracker to extract $f_0$ and confidence values from each segment at a frame rate of 100 Hz. If any of the four 5-second sub-segments has fewer than 20% of its confidence values above 0.95, we discard the entire segment. We do not run Viterbi smoothing on the $f_0$ values.

We then model the sequence of $f_0$ and confidence values with a hidden Markov model, where the hidden state

| Model Size | # Pretrain Steps | # Finetune Steps | MAESTRO | Cerberus4 | GuitarSet | MusicNet | Slakh | URMP |
|---|---|---|---|---|---|---|---|---|
| small (MT3) | 0 | 1M | .82 | .76 | .78 | .34 | .55 | .50 |
| base (MT3) | 0 | 1M | .85 | .78 | .78 | .26 | .61 | .51 |
| small (ours) | 1M | 100k | .84 | .81 | **.82** | .35 | .59 | .73 |
| base (ours) | 500k | 100k | **.87** | **.83** | **.82** | **.38** | **.66** | **.78** |
| base (ours) vs. small (MT3) | | ($\Delta\%$ Rel.) | +6.0% | +9.2% | +5.1% | +12% | +20% | +56% |

**Table 1**. Onset+Offset+Program F1 scores on different datasets, taking into account pitch, instrument, onset time, and offset time. Numbers in the first row are taken directly from Gardner et al. [20] and represent the previous state of the art. Numbers in the second row are from using the existing MT3 training procedure and data with a larger capacity model. The next two lines show the effect of our pretraining procedure. Pretraining on monophonic mixes yields some benefit even for the "Small" model, but does even better when model size is increased to "Base". Using a "Base" model without pretraining does considerably worse than with pretraining.

| **Model** | MAESTRO | Cerberus4 | GuitarSet | MusicNet | Slakh2100 | URMP |
|---|---|---|---|---|---|---|
| | | | *Frame F1* | | | |
| small (MT3) | .86 | .87 | .89 | .68 | .79 | .83 |
| base (ours) | **.90** | **.91** | **.91** | **.73** | **.85** | **.92** |
| | | | *Onset F1* | | | |
| small (MT3) | .95 | .92 | .90 | .50 | .76 | .77 |
| base (ours) | **.97** | **.95** | **.91** | **.56** | **.83** | **.90** |
| | | | *Onset+Offset+Program F1* | | | |
| small (MT3) | .82 | .76 | .78 | .34 | .55 | .50 |
| base (ours) | **.87** | **.83** | **.82** | **.38** | **.66** | **.78** |

**Table 2**. Transcription improvement over MT3 for Frame, Onset, and Onset+Offset+Program F1.

is a MIDI pitch value (0-127) or rest. (Note that we do not model onsets separately.) The transition distribution models the probability of a state change, set to expect two state changes per second or probability 0.04 of changing at any frame. The $f_0$ observations are modeled as a Gaussian distribution centered at the corresponding MIDI note frequency, with a standard deviation of $0.2$ semitones. We use a mixture of 3 Gaussians to model octave errors, with probability $0.025$ of an octave error occurring in either direction, respectively. We model $P(\mathrm{r}|\mathrm{c})$ independently of $f_0$, where r is the rest state and c is the CREPE confidence value. We treat $\mathrm{c}^v$ as the probability a frame is not a rest, where we empirically determine $v = 7.5$.

We use the forward algorithm to compute $P(f_0, \mathrm{c})$ and discard the audio segment if the log-likelihood per frame is less than $0.3$. This is useful for filtering out vocal recordings or other monophonic audio that is not well modeled by a sequence of discrete notes using the equal-tempered Western scale. Then, we use the Viterbi algorithm to infer the maximum-likelihood sequence of notes.

The above process and parameters were determined empirically on a small amount of unlabeled audio from our in-the-wild dataset. We believe that this process could be replaced by a more rigorous monophonic transcription model such as the one in Wu et al. [35]; however, we find that our process provides sufficiently strong results in practice to benefit AMT training. We plan to release our source code publicly, including our monophonic detection and transcription code. While we are unable to share our dataset, we believe that our results can likely be replicated on any large corpus of in-the-wild musical audio.

## 4. EXPERIMENTS

Our experiments measure how our pretraining method affects the accuracy of a Transformer-based transcription model. This section describes our experimental process, from data gathering to model training and evaluation.

### 4.1 Gathering Monophonic Recordings

We downloaded 10M audio recordings from an online audio/video sharing site, filtering based on metadata in an attempt to restrict ourselves to solo non-vocal musical performances on pitched instruments (i.e. not drums). We then split each recording into non-overlapping 20-second chunks and apply the monophonic detection and transcription process described in Section 3.3. Only about 1% of the chunks are detected as monophonic; we believe this is because most of the solo recordings are of performances on polyphonic instruments such as piano or guitar. Still, we choose not to exclude these instruments entirely as they are occasionally performed monophonically.

We also attempt to use the associated metadata to identify the instrument in each recording, which we use as instrument labels in the training examples. Table 3 shows the number of training clips identified as each instrument; we use the instrument vocabulary of Gardner et al. [20], originating in Manilow et al. [5]. In early experiments,

| Instrument | # Clips | Instrument | # Clips |
|---|---|---|---|
| Violin | 327,914 | Oboe | 3,234 |
| Flute/Piccolo | 96,507 | Bass Guitar | 2,264 |
| Tenor Sax | 71,737 | Electric Piano | 2,059 |
| Acoustic Guitar | 59,231 | Tuba | 1,526 |
| Electric Guitar | 58,037 | Harp | 1,233 |
| Clarinet | 53,513 | Bassoon | 1,114 |
| Trumpet | 52,695 | Xylophone | 897 |
| Cello | 23,856 | Double Bass | 751 |
| Viola | 22,027 | Baritone Sax | 317 |
| Sopr./Alto Sax | 18,784 | Synth | 288 |
| Piano | 18,626 | Voice | 123 |
| Trombone | 14,969 | Organ | 99 |
| French Horn | 7,135 | | |

**Table 3**. Number of labeled segments for each instrument in our dataset after filtering as shown in Figure 2.

we found that these metadata-based instrument labels only led to a minor improvement in the model's performance, so even in the case where metadata is unavailable we still expect our overall approach to work.

After the entire data gathering process, we are left with ~5,000 hours of automatically-transcribed monophonic music recordings. This is 3 times more than all of our fine-tuning and evaluation datasets combined, 20 times more if we exclude synthetic audio, and over 100 times more if we exclude synthetic audio and piano. Furthermore, by mixing together random segments of monophonic audio at training time, we increase the effective dataset size so substantially that the model is unlikely to ever see the same combination of source recordings twice during training.

### 4.2 Pretraining

We pretrain an encoder-decoder Transformer [33] model as described in Section 3.1 on mixes of up to 8 monophonic recordings and their corresponding note labels. The process that generates each training example is as follows:

1. Choose the number of tracks $k$ for the mix uniformly at random from 1-8.
2. Take the next $k$ 20-second clips.
3. From each clip, choose a 2.048-second segment uniformly at random.
4. Mix the audio from the $k$ clips together by summation, then peak normalize.
5. Combine the note labels from the $k$ transcriptions into a single stream. If each transcription has already been serialized, we can perform a merge to ensure the note labels end up in the correct temporal order.

The much larger training datasets generated by our heuristic labeling enables us to scale the model size that we use, allowing us to potentially leverage the scaling benefits of large Transformer models [17]. To that end, we test two different sizes of model from T5 [14]: the T5.1.1 "Small" model used in MT3, and the relatively larger "Base" model. When using the smaller "Small" T5.1.1 model, we train for 1M steps. While pretraining the "Small" model continues to improve up to 1M steps, we found that 500k pretraining steps works best for the "Base" model; this is in line with prior research which has suggested that larger pretrained models are more susceptible to diverging or overfitting [17].

Our first experiment is modeled after the one proposed in Gardner et al. [20], where we train on a combination of multiple datasets and then evaluate on a held-out portion of each dataset. Our goal with this experiment is to measure the effect of pretraining on a collection of standard transcription datasets, using standard transcription metrics. Following Gardner et al., we evaluate on 6 datasets:

**MAESTRO [2]:** a dataset of 1276 classical piano performances with MIDI captured via Disklavier.

**Slakh [5]:** a dataset of 2100 synthesized songs from the Lakh MIDI Dataset [36]. We train on 10 random subsets of tracks from each song, and evaluate on full mixes.

**Cerberus4 [10]:** a subset of the instruments in Slakh, where mixes consist of exactly 4 tracks: guitar, piano, bass, and drums.

**GuitarSet [3]:** a dataset of 360 guitar recordings, transcribed using a hexaphonic pickup.

**MusicNet [6]:** a dataset of 330 classical music recordings, with MIDI files aligned by dynamic time warping.

**URMP [37]:** a dataset of 44 classical music recordings with instruments independently recorded and transcribed.

We measure transcription performance using the F1 metric over notes (Onset+Offset+Program), where in order for two notes to "match" they must have the same pitch and instrument and be close enough in onset time and offset time. We use the default tolerances in the `mir_eval` [38] library: onset times must be within 50 ms to match, and offset times must be within the larger of 50 ms or 20% of the note's true duration.

Our results are shown in Table 1. When using a "Small" model, pretraining provides a small increase in F1 score across all datasets. A further advantage of pretraining is that it lets us scale up to a "Base" model, which provides a more significant boost. Note that using a "Base" model without pretraining does not provide the same performance improvement, likely because the existing datasets are simply too small for the model to take advantage of its increased capacity. Table 2 compares our best model to MT3 across all F1 metrics and shows that pretraining on monophonic mixtures leads to substantial improvements in the state-of-the-art. It's worth noting that MusicNet is now known to have many misalignements, that are likely the cause of the lower scores [13].

| Model Size | # Pretrain Steps | # Finetune Steps | MAESTRO | Cerberus4 | GuitarSet | MusicNet | Slakh | URMP |
|---|---|---|---|---|---|---|---|---|
| small (ours) | 1M | 0 | .04 | .00 | .13 | .03 | .00 | .19 |
| base (ours) | 500k | 0 | .03 | .00 | .12 | .03 | .00 | .22 |
| small (MT3) | 0 | 525k | .28 | .07 | .19 | .14 | .02 | .17 |
| base (ours) | 500k | 1000 | **.39** | **.11** | **.29** | **.17** | **.06** | **.33** |

**Table 4**. Onset+Offset+Program F1 scores (considering pitch, instrument, and onset/offset times) using the LODO methodology, where each dataset in turn is held out from training and used only for evaluation. The first two rows use no finetuning, the third row is taken directly from Gardner et al. [20], and the fourth row uses a model pretrained on monophonic mixes and finetuned for only 1000 steps. LODO scores are lower for all datasets, but pretraining provides a considerable boost. This evaluation is somewhat unfair to Slakh as it contains instruments that do not appear in the other datasets.

| **Model** | MAESTRO | Cerberus4 | GuitarSet | MusicNet | Slakh2100 | URMP |
|---|---|---|---|---|---|---|
| | | | *Frame F1* | | | |
| small (MT3) | .60 | .55 | .58 | .53 | .55 | .76 |
| base (ours) | **.65** | **.57** | **.69** | **.62** | **.64** | **.82** |
| | | | *Onset F1* | | | |
| small (MT3) | .28 | .21 | **.78** | .18 | .14 | .23 |
| base (ours) | **.83** | **.54** | .71 | **.48** | **.45** | **.54** |
| | | | *Onset+Offset+Program F1* | | | |
| small (MT3) | .28 | .07 | .19 | .14 | .02 | .17 |
| base (ours) | **.39** | **.11** | **.29** | **.17** | **.06** | **.33** |

**Table 5**. Zero-shot transcription improvement over MT3 for Frame, Onset, and Onset+Offset+Program F1.

### 4.3 Zero-Shot Experiments

While standard in AMT, the methodology of the previous experiment is unsatisfactory in that for most real applications, we want to be able to automatically transcribe recordings from musical domains that were not present in our training data; splitting a homogeneously-constructed dataset into "training" and "test" partitions is not sufficient. This "out-of-domain" or "zero-shot" transfer is considered an extremely challenging task in AMT [13, 20]. We simulate the zero-shot condition with a leave-one-dataset-out (LODO) methodology, similar to the experiment described in the appendix of Gardner et al. [20].

For each of the five evaluation datasets or "folds" (since Slakh and Cerberus overlap we combine them into a single fold), we train a model on four of the folds and test on the other fold. This ensures that the model has not seen any data from the test domain during training, a much more difficult task as the model must be robust to a wider range of instrument timbres and recording conditions. Under LODO evaluation, Gardner et al. report an Onset+Offset+Program F1 score of less than 0.3 for all datasets, and less than 0.2 for all non-MAESTRO datasets.

Our results for the LODO evaluation are shown in Tables 4 and 5. Although our LODO F1 scores are lower than in the supervised case due to the difficulty of the task, pretraining on monophonic mixes increases the F1 score for all datasets. This is unsurprising, as our pretraining data exposes the model to a much wider range of instrument sounds and recording conditions. Possibly more surprising is that the pretrained model does very poorly without further finetuning on existing datasets (first two rows of Table 4); we have no satisfying explanation for this phenomenon and leave it to future work.

## 5. CONCLUSION

We have shown that we can take advantage of the relative ease of monophonic music transcription to improve upon the state of the art in multi-instrumental polyphonic music transcription by obtaining a large number of monophonic recordings, heuristically transcribing them, and mixing several of them together at a time as pretraining examples for a neural network model. This yields improvements across many datasets, in both standard and zero-shot multi-task settings. Notably, this pretraining boosts performance of downstream transcription models despite the fact that the pretraining audio is musically "incoherent", consisting of randomly-mixed monophonic audio tracks without regard to key, tempo, style, composition, or instrumentation.

Our work provides the first evidence that Internet-scale pretraining can be used to improve Transformer-based AMT models, and we do so with a simple set of heuristics that is straightforward to implement and fast to execute. While the benefits of large-scale pretraining for large Transformer models are well-known, the benefits of pretraining have yet to arrive to the AMT community, despite the wide availability of unlabeled audio data relative to the amount of audio with transcription-quality labels. We hope that the methods presented here, along with the accompanying open-source checkpoints and code, enable further advances in using large-scale audio data for scaling AMT models beyond the scope of existing supervised datasets.

## 6. REFERENCES


[1] MIDI Manufacturers Association and others, "The complete MIDI 1.0 detailed specification," *Los Angeles, CA, The MIDI Manufacturers Association*, 1996.

[2] 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 *ICLR*, 2019.

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

[4] M. Cartwright and J. P. Bello, "Increasing drum transcription vocabulary using data synthesis," in *DAFX*, 2018.

[5] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *WASPAA*, 2019.

[6] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *ICLR*, 2017.

[7] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *ICASSP*, 2018.

[8] J. W. Kim and J. P. Bello, "Adversarial learning for improved onsets and frames music transcription," *arXiv preprint arXiv:1906.08512*, 2019.

[9] R. Kelz, S. Böck, and G. Widmer, "Deep polyphonic ADSR piano note transcription," in *ICASSP*, 2019.

[10] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *ICASSP*, 2020.

[11] 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.

[12] 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 *ICPR*, 2021.

[13] B. Maman and A. H. Bermano, "Unaligned supervision for automatic music transcription in the wild," *arXiv preprint arXiv:2204.13668*, 2022.

[14] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text Transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[15] J. Beal, H.-Y. Wu, D. H. Park, A. Zhai, and D. Kislyuk, "Billion-scale pretraining with vision transformers for multi-task visual representations," in *WACV*, 2022, pp. 564–573.

[16] O. Hrinchuk, M. Popova, and B. Ginsburg, "Correction of automatic speech recognition with Transformer sequence-to-sequence model," in *ICASSP*, 2020.

[17] D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish, "Scaling laws for transfer," *arXiv preprint arXiv:2102.01293*, 2021.

[18] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, and D. Metzler, "Scale efficiently: Insights from pre-training and fine-tuning Transformers," *arXiv preprint arXiv:2109.10686*, 2021.

[19] K. Lu, A. Grover, P. Abbeel, and I. Mordatch, "Pretrained Transformers as universal computation engines," *arXiv preprint arXiv:2103.05247*, 2021.

[20] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *ICLR*, 2022.

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[22] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *ICCV*, 2019.

[23] J.-H. Kim, W. Choo, and H. O. Song, "Puzzle mix: Exploiting saliency and local statistics for optimal mixup," in *ICML*, 2020.

[24] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou, "How does mixup help with robustness and generalization?" in *ICLR*, 2020.

[25] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *NeurIPS*, 2019.

[26] L. Callender, C. Hawthorne, and J. Engel, "Improving perceptual quality of drum transcription with the Expanded Groove MIDI Dataset," *arXiv preprint arXiv:2004.00188*, 2020.

[27] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez Gutiérrez, and J. P. Bello, "An analysis/synthesis framework for automatic $f_0$ annotation of multitrack datasets," in *ISMIR*, 2017.

[28] E. Fonseca, A. Jansen, D. P. W. Ellis, S. Wisdom, M. Tagliasacchi, J. R. Hershey, M. Plakal, S. Hershey, R. C. Moore, and X. Serra, "Self-supervised learning from automatically separated sound scenes," in *WASPAA*, 2021.

[29] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *ICASSP*, 2017.

[30] L. Prétet, R. Hennequin, J. Royo-Letelier, and A. Vaglio, "Singing voice separation: A study on training data," in *ICASSP*, 2019.

[31] X. Song, Q. Kong, X. Du, and Y. Wang, "CatNet: Music source separation system with mix-audio augmentation," *arXiv preprint arXiv:2102.09966*, 2021.

[32] Q. Kong, Y. Cao, H. Liu, K. Choi, and Y. Wang, "Decoupling magnitude and phase estimation with deep ResUNet for music source separation," in *ISMIR*, 2021.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[34] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with Transformers," in *ISMIR*, 2021.

[35] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "MIDI-DDSP: Detailed control of musical performance via hierarchical modeling," in *ICLR*, 2022.

[36] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching*. Columbia University, 2016.

[37] 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 Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

[38] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "`mir_eval`: A transparent implementation of common MIR metrics," in *ISMIR*, 2014.