Detecting Music Performance Errors with Transformers

Benjamin Shiue-Hal Chou, Purvish Jajal, Nicholas John Eliopoulos, Tim Nadolsky, Cheng-Yun Yang, Nikita Ravi, James C. Davis, Kristen Yeon-Ji Yun, Yung-Hsiang Lu

School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, 47907
{chou150, pjajal, neliopou, tnadolsk, yang2316, ravi30, davisjam, yun98, yunglu}@purdue.edu

Abstract

Beginner musicians often struggle to identify specific errors in their performances, such as playing incorrect notes or rhythms. There are two limitations in existing tools for music error detection: (1) Existing approaches rely on automatic alignment; therefore, they are prone to errors caused by small deviations between alignment targets.; (2) There is insufficient data to train music error detection models, resulting in over-reliance on heuristics. To address (1), we propose a novel transformer model, *Polytune*, that takes audio inputs and outputs annotated music scores. This model can be trained end-to-end to implicitly align and compare performance audio with music scores through latent space representations. To address (2), we present a novel data generation technique capable of creating large-scale synthetic music error datasets. Our approach achieves a 64.1% average Error Detection F1 score, improving upon prior work by 40 percentage points across 14 instruments. Additionally, our model can handle multiple instruments compared with existing transcription methods repurposed for music error detection.

Code — https://github.com/ben2002chou/Polytune

1 Introduction

Beginner musicians often need help identifying errors in their performance. For example, novice musicians may struggle with sight reading or miss notes due to a lack of muscle memory. Access to music education programs which could help address these issues is limited; in the USA alone, approximately 4 million K-12 students do not have access to music education (Morrison et al. 2022).

To bridge this gap, commercial music tutoring tools have become essential resources. Beginner musicians can practice more effectively, and teachers are provided with insights into students' progress (Nart 2016; Apaydınlı 2019). The significant demand for such automated solutions is evident, with apps like Yousician (Yousician Ltd 2024) and Simply Piano (JoyTunes 2024) each having over 10 million downloads globally.

However, Simply Piano and Yousician only identify notes as correct or incorrect, without offering detailed feedback such as missed or extra notes. They are also incapable of automatically aligning the user's performance with a reference score. Instead, Simply Piano and Yousician rely on the user to match their performance with the reference audio. Furthermore, their models cannot handle multiple instruments.

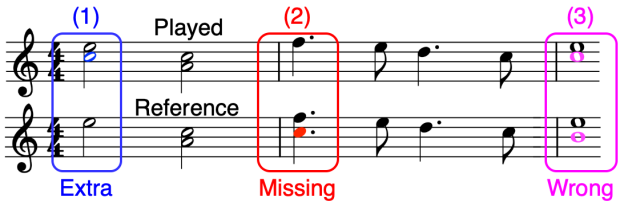

Figure 1: The score on top is the performance transcription and the score below is the reference. This paper can detect three types of errors. (1) is an extra note. "*C*" is played, but it is not expected by the score. (2) is a missed note. In this figure, "*C*" is not played. (3) is a wrong note, which is actually just a missed note and an extra note happening simultaneously. We expect the player to play a "*C*" but instead a "*B*" is played. The model inputs the score and the music student's recorded audio and labels the notes detected from the recording as "Missed", "Extra", or "Correct".

The research community has also attempted to provide fine-grained music performance feedback but has had limited success (Benetos, Klapuri, and Dixon 2012; Wang, Ewert, and Dixon 2017). A major paradigm of prior work is to temporally align a student's performance with a reference score, and then identify differences.

These alignment-based approaches often fail when there are deviations in the played notes from the score, even if they are minor. The resulting misalignment of notes leads to inaccurate error detection and ineffective feedback for students.

In this paper, we introduce a transformer based-model (Vaswani et al. 2017) called *Polytune* to detect musician errors without relying on automatic alignment. Our model utilizes an end-to-end trainable approach for error detection. Specifically, *Polytune* takes audio spectrogram pairs as inputs, and then outputs an annotated musical score without requiring any manual intervention. We train *Polytune* to annotate notes with a variety of labels beyond "correct" or

"incorrect", as shown in Fig. 1. For training, we require a larger set of training samples than what is provided by existing datasets. Thus, we introduce an algorithm for synthetically generating errors in existing music datasets, CocoChorales (Wu et al. 2022a) and MAESTRO (Hawthorne et al. 2018). We name the resulting augmented datasets as *CocoChorales-E* and *MAESTRO-E*, respectively.

We evaluate *Polytune* and previous works on *CocoChorales-E* and *MAESTRO-E*, which encompass 14 different instruments and a variety of performance errors. To evaluate error detection performance, we adapt the transcription F1 score commonly used in music transcription tasks (Raffel et al. 2014).

We make the following contributions:

1. *Polytune* achieves **state-of-the-art** performance in music error detection, with F1 scores of **95.0%** for *Correct*, **49.2%** for *Missed*, and **48.0%** for *Extra*. These results represent improvements of **+58.0%**, **+41.6%**, and **+22.1%**, respectively. On average, our method outperforms previous approaches by **40** percentage points across 14 instruments.
2. We develop an end-to-end training method for audio comparison, eliminating the need for manual alignment.
3. We create a synthetic error generation pipeline for injecting errors into music transcription datasets and create two error detection datasets: *CocoChorales-E* and *MAESTRO-E*.

2 Background and Related Work

This section explores existing methods for score-informed error detection, advances in token-based music transcription, and the deficiencies of current datasets.

2.1 Application Context: Music Terminology

We define key musical terms used in this paper: **Music Score:** A written guide showing notes, rhythms, and instructions for performance. **Music Note:** A symbol indicating the pitch and duration of a sound. **Chord:** Multiple notes played together. **Chordal:** Music with multiple notes at once. **MIDI:** A standard that uses symbols to represent musical events like pitch and duration, enabling electronic instruments and computers to communicate. **Track (MIDI Track):** A sequence of MIDI events representing one instrument.

2.2 Score-Informed Music Performance Assessment

Music performance assessment requires comparing a player's rendition against a musical score. In **score-informed** music error detection systems, the score serves as a reference for detecting errors like missed or extra notes. As shown in Fig. 2a, systems by Benetos et al. (Benetos, Klapuri, and Dixon 2012) and Wang et al. (Wang, Ewert, and Dixon 2017) use Automatic Music Transcription (AMT) to convert audio into musical notation and align it with the score to identify errors like missed or extra notes. Dynamic Time Warping (DTW) (Sakoe and Chiba 1978) is a widely

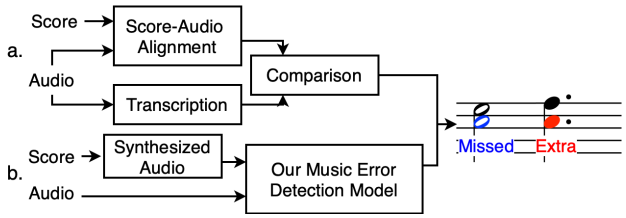

Figure 2: Illustration of differences between (a) previous work and (b) our *Polytune* . Our approach simplifies music error detection by using an end-to-end trainable architecture. This eliminates the need to explicitly align and compare audio, which is error-prone.

used technique in music tutoring applications for aligning musical performances with reference scores (Benetos, Klapuri, and Dixon 2012; Wang, Ewert, and Dixon 2017; Ewert, Wang, and Sandler 2016). DTW employs dynamic programming to find the optimal alignment by minimizing temporal differences between two time series. However, DTW faces challenges when dealing with music that contains overlapping notes, as shown in Fig. 3. This forces DTW to compromise, resulting in imperfect alignment or distortion. Instead, our approach shown in Fig. 2b replaces DTW with a learnable alignment mechanism that does not encounter the same problems as DTW. We find this leads to more accurate predictions.

2.3 Token-Based Automatic Music Transcription (AMT) Methods

Recent AMT advancements are influenced by Natural Language Processing (NLP) techniques, particularly in how outputs are represented. In **token-based** AMT models, the output is represented as sequences of tokens, where each token corresponds to an element of musical information. MT3 (Gardner et al. 2022) is a token-based AMT system using a T5 (Raffel et al. 2020) transformer to output MIDI-like tokens (Oore et al. 2020). These tokens directly represent music and are inspired by the MIDI format. AMT is treated as a language modeling problem, predicting a sequence of event tokens like time tokens, program tokens, note-on/off tokens, and note tokens.

MT3 uses spectrogram frames as the encoder input, providing context for the autoregressive decoder, which predicts future tokens based on past outputs. This allows the model to capture temporal patterns in music, similar to how NLP models handle text sequences. By using token-based representations, MT3 accurately transcribes music. Building on this capability, we adapt MT3 in this paper specifically for error detection.

2.4 Datasets for Music Error Detection

Deep learning models require extensive training data. Unfortunately, existing datasets for music error detection are limited, with only seven tracks, 15 minutes of audio, and just 40 errors in total (Benetos, Klapuri, and Dixon 2012). This scarcity restricts the training of end-to-end models. Training on such a small dataset would cause the model to overfit

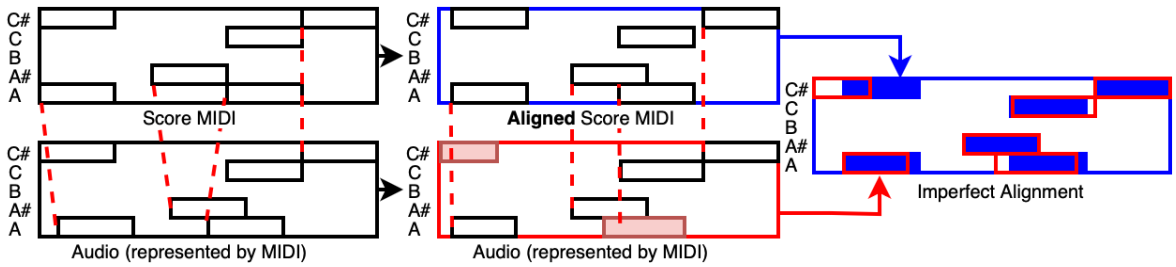

Figure 3: Deficiency of Dynamic Time Warping (DTW): DTW encounters challenges when aligning complex sequences of overlapping musical notes. Specifically, when DTW aligns one note within a group, it often compromises the alignment of other notes. This issue arises because DTW attempts to minimize timing differences across the entire sequence. Once the MIDI score is aligned with the audio, the algorithm may align one note correctly but misalign others, leading to potential classification errors. For example, as seen in the overlay of score and audio, aligning an "*A*" note might result in the misalignment of the adjacent "*C#*" note, causing the algorithm to mistakenly classify a correctly played note as a missed "*C#*" and then an extra "*C#*".

(Ying 2019). This highlights the need for more comprehensive datasets to advance research and development in this field.

For AMT, larger datasets like CocoChorales (Wu et al. 2022a) and MAESTRO (Hawthorne et al. 2018) offer over 1,000 tracks and more than 200 hours of audio. Although they are primarily used for transcription tasks, these datasets offer a rich source of musical content. In this paper we adapt them to train music error detection models.

3 End-to-end Music Error Detection

In this section, we outline key designs of our architecture and training regime. First, we provide the intuition of our model and describe how it is related to prior work in Sec. 3.1. Second, we describe and justify design decisions of our model architecture in Sec. 3.2. Third, we investigate the end-to-end training strategy, and our approach to generating large-scale datasets with synthetic errors in Sec. 3.3. Last, we provide upgraded implementations of (Benetos, Klapuri, and Dixon 2012) and (Wang, Ewert, and Dixon 2017) to fairly represent older work and contextualize our contributions better in Sec. 3.4.

3.1 Intuition and Comparison to Prior Work

Our method addresses the main limitations of previous methods, as shown in Tab. 1. Benetos et al. (Benetos, Klapuri, and Dixon 2012) and Wang et al. (Wang, Ewert, and Dixon 2017) rely on time-warping algorithms to align music scores with audio recordings. These time-warping algorithms suffer from alignment inaccuracies when notes that should be simultaneous, fall out of sync.

To overcome these limitations, we introduce a learnable end-to-end model *Polytune*. It processes two unaligned audio sequences—one from the musical score and one from the performance. We design the inputs and outputs to implicitly teach the model alignment, leading to more accurate predictions. This eliminates the need for complex pre-processing and post-processing steps, such as alignment and comparison, as illustrated in Fig. 2. We developed an updated baseline using the score-informed transcription systems of Benetos et al. (Benetos, Klapuri, and Dixon 2012) and Wang et al. (Wang, Ewert, and Dixon 2017), incorporating the MT3 model and Dynamic Time Warping (DTW) to reflect current best practices (Sec. 3.4).

Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Simply Piano	✓	✗	✗	✗	✗	✗	✗
Yousician	✓	✗	✗	✗	✗	✗	✗
Benetos*	✓	✓	✓	✗	✗	✗	✗
Wang*	✓	✓	✓	✗	✗	✗	✗
Combined*	✓	✓	✓	✓	✗	✗	✗
***Polytune** *	✓	✓	✓	✓	✓	✓	✓

Table 1: Qualitative comparison of models based on key features. We evaluate the models using the following criteria: **1**: Correct note detection, **2**: Missed note detection, **3**: Extra note detection, **4**: Multi-instrument, **5**: End-to-end training, **6**: Automatic alignment, **7**: No heuristics (Rules or methods based on experience). **Combined** refers to our baseline model, which reimplements and integrates the methods from Benetos et al. and Wang et al. to enhance performance. Models marked with a star (*) indicate those directly evaluated against in our study.

3.2 Music Error Detection Model

We show how our model architecture can be designed to implicitly learn better sequence alignment in latent representation, which is crucial for accurately detecting music performance errors. Since transcription plays a key role in existing methods for music error detection, we base our approach on MT3. Specifically, we frame music error detection as a sequence-to-sequence task, where the input consists of audio spectrogram frames, and the output is represented by customized tokens for error detection.

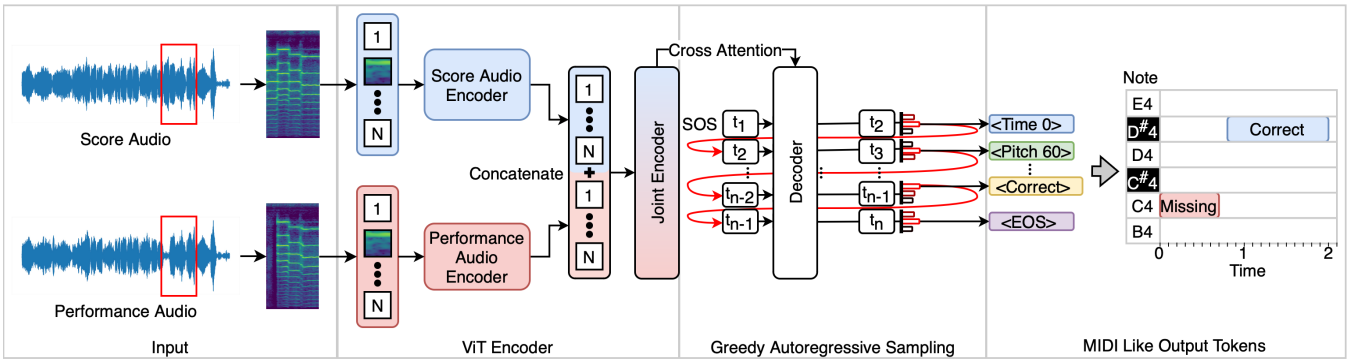

Figure 4: Architecture of *Polytune*. The diagram illustrates the process flow starting with the Score and Performance Audio inputs, each processed through dedicated AST encoders. These encoded features are concatenated and passed through a joint encoder and a decoder with cross-attention for temporal sequencing. The output is generated through greedy autoregressive sampling, providing MIDI-like tokens that classify notes as correct or missing.

Hawthorne et al. treats music transcription as a sequence-to-sequence modeling task, using audio spectrogram frames as inputs and MIDI-like tokens as outputs (Hawthorne et al. 2021). Building on this foundation, MT3 (Gardner et al. 2022) advanced the piano transcription model by generalizing it to transcribe different instruments. Therefore, we base our model's decoder on MT3, a well-studied state-of-the-art Automatic Music Transcription (AMT) method.

The architecture of *Polytune* is detailed in Fig. 4. The design fuses two inputs: the audio from the musical score and the player's corresponding performance. These are encoded into a joint encoding space. We treat our model as multimodal, utilizing modality-specific encoders similar to those in (Gong et al. 2023; Akbari et al. 2021). We reason that although both inputs are audio spectrograms, the score and performance audio serve distinct roles in music error detection.

We fuse two Audio Spectrogram Transformer (Gong, Chung, and Glass 2021) (AST) encoders with a joint encoder. Then, we use a standard T5 decoder, with greedy autoregressive decoding to output sequences of tokens, described in Sec. 3.3. To bridge the fused AST encoders and the T5 decoder, we employ a linear projection layer to transform the embedding size from AST's 768 to T5's 512. The encoder outputs condition the decoder via a cross-attention mechanism.

3.3 End-to-End Training of the Music Error Detection Model

Input Data We use audio inputs synthesized from musical scores rather than symbolic representations like MIDI tokenizations. This approach has several advantages:

1. **Generalization:** Audio inputs generalize across instruments and capture their full sound, including characteristics like vibrato (violin), trills (flute), and pitch bending (guitar), which MIDI tokenizations fail to represent. MIDI tokenizations are designed primarily for piano. Their design prohibits them from representing non-fixed-pitch instruments or capturing instrument-specific nuances (e.g., see (Hsiao et al. 2021; Huang and Yang 2020)).
2. **Extensibility and Expressiveness:** Audio preserves absolute timing and dynamics, enabling the detection of new error types, such as timing or dynamics errors, without requiring changes to the input representation. In contrast, MIDI tokenizations reduce sequence length to address the $O(N^2)$ computational cost of transformers but sacrifice expressiveness and timing precision. For instance, the Octuple tokenizer achieves a fourfold reduction in sequence length compared to REMI (Huang and Yang 2020), but introduces errors in complex time signatures (Fradet, Briot, and Chhel 2021).
3. **Comparability:** Audio inputs allow users to compare performances against any pre-recorded audio, enabling analysis in contexts where scores are unavailable or nonexistent, such as impromptu jazz or folk music.
4. **Simplicity in Design:** Using audio for both inputs simplifies the model by reusing the same feature extractor design for both encoders. We conjecture that using a single input format for the music score and performance enables the model to learn implicit alignment more effectively. This avoids the additional capacity required to process and compare different modalities. By using the same input format, we eliminate unnecessary complexity from the task.

Fig. 4 illustrates our data flow. We divide the student's recorded audio and the synthesized score audio into non-overlapping segments. The segment length is 2.145 seconds. Each segment undergoes a short-time Fourier transform, producing spectrogram frames. Finally, we tokenize each spectrogram frame using the ViT patch embedding approach (Dosovitskiy et al. 2021) and feed the resulting 512 patches into our ViT encoder.

Output Data We output a sequence of MIDI-like tokens to represent the music scores with errors annotated. The complete token vocabulary is summarized in Tab. 2. The vocabulary is similar to (Gardner et al. 2022; Hawthorne et al. 2021) with the following differences:

Name	Description	Values
SOS	Start of sequence	1
End Tie	Defines non-active notes from previous segments	1
Time	Specifies note timing in a segment	205
Label	Note is extra, missed, or correct	3
On/Off	Note is played or not	2
Note	MIDI Pitch	128
EOS	End of sequence	1

Table 2: Vocabulary for MIDI note events, adapted from (Gardner et al. 2022). Compared with their vocabulary, we add additional note labels "extra" and "missed".

- We do not specify an instrument because we assume in a music tutoring context musicians would typically play a single instrument. Our error detection model is instrument agnostic because we use the same output formats for errors of different instruments.
- We add "Label" tokens to annotate each note with one of the error categories defined in Sec. 1.

While Compound Word (Hsiao et al. 2021) and Revamped MIDI (REMI) (Huang and Yang 2020) tokenizations offer efficient sequence lengths compared to Midi-like tokenization, they come with drawbacks. These methods trade off fine-grained timing accuracy for sequence length by using bars and beats to represent time. MIDI-like tokenization for output captures timing more precisely and leads to more accurate predictions.

Generating Datasets A large number of labeled musician errors are required for end-to-end training. However, no large-scale datasets exist for music error detection. The only existing dataset by Benetos et al. (Benetos, Klapuri, and Dixon 2012) has just 7 tracks.

In order to address this gap, we create two datasets, *MAESTRO-E* and *CocoChorales-E*, that have 1000+ samples for each instrument. *MAESTRO-E* contains over 200 hours of audio, with 1,000+ tracks focused on piano, including 200k note and timing errors. *CocoChorales-E* includes 300+ hours of audio across 10k+ tracks and 13 instruments, featuring over 25k note and timing errors. The dataset from Benetos et al. (Benetos, Klapuri, and Dixon 2012) is much smaller, with only 15 minutes of audio, 7 tracks, and 40 note errors. We use MIDI samples from CocoChorales and MAESTRO augmented with common performance errors such as wrong note, missed note, and extra note. Additionally, we synthesize audio using MIDI-DDSP (Wu et al. 2022b) for each of the augmented samples. We define training labels by splitting our dataset into three MIDI files for the classes: *Correct*, *Missed*, and *Extra* as defined in Sec. 1.

The algorithm for generating MIDI errors is detailed in Alg. 1. We introduce errors into each MIDI file by selecting notes based on a Poisson distribution with a mean rate parameter λ, where λ is sampled from a uniform distribution $U(0.1, 0.4)$ and applying an error type. Then, the randomly selected notes are assigned an error type, and their time and pitch are augmented accordingly. Offset magnitudes for time and pitch are sampled from two truncated normal distribution distributions, P and Q, with mean 0 and standard deviation of 1 and 0.02, respectively. Note that P and Q can have different parametrizations. We believe our choice of truncated normal distributions simulate reasonable performance errors in pitch and timing (Trommershäuser et al. 2005; Tibshirani, Price, and Taylor 2011).

Algorithm 1: MIDI Error Generation Algorithm. This algorithm introduces errors into MIDI files. Abbreviations: *PC* (pitch change), *TS* (timing shift), *EN* (extra note).

Require: All notes in MIDI track A, error rate λ, offset distributions P, Q

Select notes from A to augment with probability λ
for each note selected **do**
  err_type $\leftarrow$ rand({miss, PC, TS, EN})
  if err_type $=$ miss **then**
    Remove note;
  else if err_type $=$ PC **then**
    $\epsilon_p \leftarrow$ sample(P)
    Offset pitch by ϵ;
  else if err_type $=$ TS **then**
    $\epsilon_t \leftarrow$ sample(Q)
    Offset time by ϵ;
  else if err_type $=$ EN **then**
    $\epsilon_p \leftarrow$ sample(P)
    $\epsilon_t \leftarrow$ sample(Q)
    Insert note with time offset ϵ_t and pitch offset ϵ_p;
  end if
end for

3.4 Baseline

We choose Benetos et al. and Wang et al. as our baseline for score-informed error detection (Benetos, Klapuri, and Dixon 2012; Wang, Ewert, and Dixon 2017). They are the most relevant prior works to score informed error detection. *We created upgraded, open-source versions of these works to enable fairer comparisons.*

In our re-implementation of Benetos et al. and Wang et al., we retain the foundational concepts of each approach but incorporate changes to each component of the transcription pipeline to reflect advances in automatic music transcription (AMT). Specifically, we replace the original non-negative matrix factorization-based transcription with the state-of-the-art MT3 model. Additionally, we use Dynamic Time Warping (DTW) instead of the less accurate Windowed Time Warping (WTW). Our updated baseline shows comparable performance and also includes the ability to detect multiple instruments.

4 Results and Discussion

In this section, we compare *Polytune* with prior work across our datasets *MAESTRO-E* and *CocoChorales-E*. First, we

Category	Method	*MAESTRO-E*			*CocoChorales-E*		
		Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Correct	*Polytune*	**96.9%**	**84.1%**	**90.1%**	**96.2%**	**94.8%**	**95.4%**
	Baseline	46.8%	40.7%	43.5%	42.3%	33.1%	36.7%
Missed	*Polytune*	**30.7%**	**26.3%**	**26.8%**	**51.6%**	**55.1%**	**51.3%**
	Baseline	3.9%	24.1%	6.6%	5.3%	17.3%	7.7%
Extra	*Polytune*	**70.5%**	76.3%	**72.0%**	**47.1%**	**48.5%**	**46.8%**
	Baseline	26.3%	**87.9%**	39.9%	16.1%	52.6%	23.5%
Average	*Polytune*	**64.9%**	**62.2%**	**62.9%**	**65.0%**	**66.1%**	**64.5%**
	Baseline	25.6%	50.9%	30.0%	21.2%	34.6%	22.7%

Table 3: Comparison of *Polytune* and Baseline across Error Types in Two Datasets (*MAESTRO-E* and *CocoChorales-E*). Boldface indicates the best performance in each category. Our method demonstrates superior performance in most cases, except for a higher recall for extra notes in the Baseline. This is explained in Sec. 4.2.

Instrument	Correct		Missed		Extra	
	Polytune	**Baseline**	***Polytune***	**Baseline**	***Polytune***	**Baseline**
Average	**95.0%**	37.0%	**49.2%**	7.6%	**48.0%**	25.9%
Piano	**90.1%**	43.5%	**26.8%**	6.6%	**72.0%**	39.9%
Flute	**96.0%**	38.9%	**56.0%**	7.2%	**52.0%**	26.6%
Clarinet	**95.6%**	38.3%	**49.7%**	6.7%	**46.6%**	24.1%
Oboe	**96.3%**	33.4%	**58.4%**	6.7%	**48.1%**	25.9%
Bassoon	**94.4%**	34.7%	**48.9%**	6.4%	**41.7%**	17.1%
Violin	**95.5%**	36.1%	**57.1%**	7.5%	**48.8%**	27.3%
Viola	**95.1%**	36.1%	**46.9%**	5.9%	**47.7%**	26.1%
Cello	**94.9%**	37.5%	**42.7%**	6.9%	**46.8%**	21.7%
Trumpet	**96.3%**	37.8%	**58.7%**	8.8%	**53.6%**	26.6%
French Horn	**96.1%**	38.4%	**53.9%**	5.9%	**43.2%**	23.7%
Tuba	**95.2%**	37.3%	**45.4%**	8.1%	**45.6%**	17.8%
Trombone	**94.8%**	35.0%	**50.4%**	7.1%	**44.8%**	21.7%
Contrabass	**94.2%**	35.7%	**42.0%**	8.9%	**38.6%**	19.9%
Tenor Sax	**95.7%**	39.7%	**56.2%**	14.2%	**45.7%**	25.1%

Table 4: Error Detection F1 Score Comparison by Instrument and Error Category. Only piano is from *MAESTRO-E*.

describe the environment used to train and test our approach in Sec. 4.1. Second, we present the performance of our model and the baseline in Sections 4.2 and 4.3. Third, insight regarding the training of *Polytune* is discussed in Sec. 4.4.

4.1 Experiment Setup

We used Pytorch 2.3.0 and Huggingface Transformers 4.40.1 for model design and training. The mir_eval package is used for evaluating Error Detection F1 scores.

All models were trained on an NVIDIA A100-80GB GPU running a Linux operating system. The datasets introduced in this work, *MAESTRO-E* and *CocoChorales-E*, were generated using AMD EPYC 7713 3.0 GHz CPUs. Dataset generation took approximately 12 hours per dataset, with *MAESTRO-E* using 1 CPU and *CocoChorales-E* using 256 CPUs in parallel.

For model evaluation, we define the Error Detection F1 score for each error type—Extra (E), Missed (M), and Correct (C)—by isolating the corresponding notes and applying the transcription onset F1 metric from `mir_eval` (Raffel et al. 2014). All results are based on a combined test set of 4401 tracks.

4.2 Quantitative Evaluation

We present a comparison of our method against the baseline across different categories for Error F1, precision, and recall. As shown in Tab. 3, our method generally outperforms the baseline derived from (Benetos, Klapuri, and Dixon 2012; Wang, Ewert, and Dixon 2017), except for a higher recall of 87.9% in extra notes for *MAESTRO-E* by the baseline. This discrepancy arises because extra and missing notes disrupt DTW alignment, leading the baseline's heuristics to mark many unmatched notes as extra. However, this misalignment also causes correct notes to be misclassified as extra, resulting in high recall but low precision, particularly

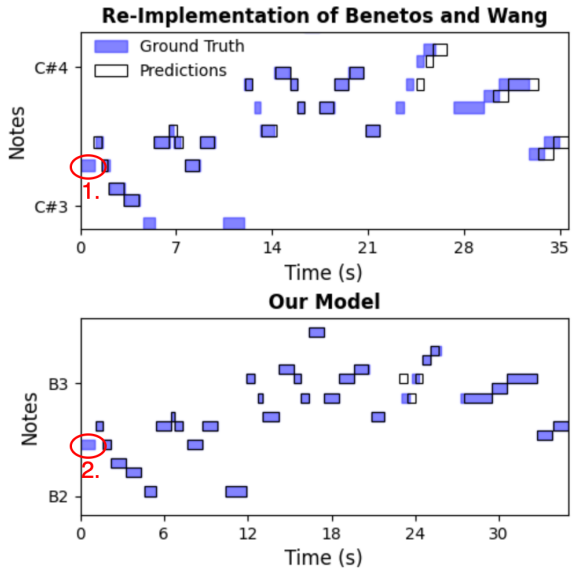

Figure 5: Qualitative Comparison of MIDI Note Events: This figure shows the "correct category" note events detected by each model for a track in the *CocoChorales-E* dataset. Ground truth notes are filled blue rectangles, and model predictions are black-outlined rectangles. Music note 1 is caused by a transcription error by MT3 and a similar error occurs with Music note 2. Overall, *Polytune* better matches the ground truth and has fewer false detections than our Benetos and Wang re-implementation.

in *MAESTRO-E*. Overall, for extra notes in *MAESTRO-E*, *Polytune* still outperforms the baseline by 32.1 F1 percentage points. A similar pattern is seen in missed notes, where the baseline shows higher recall but lower precision. Additionally, we compared Error Detection F1 scores by instrument (Tab. 4). Our findings indicate that explicit alignment is not essential for effective music error detection.

4.3 Qualitative Evaluation

Fig. 5 compares *Polytune* error detection results with the baseline. Alignment errors are visible as offsets between the outlined and colored boxes. The baseline model often fails to detect missing and extra notes due to DTW alignment issues. Meanwhile, *Polytune* does not make the same mistakes and achieves greater precision.

The baseline model's alignment errors stem from DTW struggling to align examples containing mistakes with the reference score. These alignment errors are the main reason for the baseline's false detections. However, *Polytune* also struggles with notes that transcription models find challenging. See Fig. 5 for an example where MT3 misdetects Note 1, and *Polytune* makes a similar error with Note 2.

4.4 Insights on Training Music Error Detection Models

In our work, we identify two key insights for training our music error detection models. The first insight involves addressing the class imbalance in our datasets, which we tackle using a weighted cross-entropy loss. The second insight relates to the limited benefits of self-supervised pretraining for this task, causing us to prioritize direct supervised learning instead.

First, the *MAESTRO-E* and *CocoChorales-E* datasets are class imbalanced, with *MAESTRO-E* having an average error-to-correct ratio of 1:9 and *CocoChorales-E* having a ratio of 1:4. To address this imbalance, we use a weighted cross-entropy loss, as shown in Equation 1.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} [\alpha(y_i) \cdot \text{CE}(y_i, \hat{y}_i)] \quad (1)$$

Equation 1 defines the weighted cross-entropy loss $\mathcal{L}$, averaged over N tokens. $\text{CE}(y_i, \hat{y}_i)$ is the cross-entropy between true label y_i and prediction $\hat{y}_i$, weighted by a function $\alpha(y_i)$. For our training, $\alpha(y_i)$ is 10 when y_i is an error token and 1 when it is not.

Second, we explored the effectiveness of self-supervised pretraining in improving error detection. Previous works, such as those by Huang et al. (Huang et al. 2022) and Gong et al. (Gong et al. 2023), utilized this approach for audio classification and retrieval. However, we found no benefit to the final Error Detection F1 scores in our case. Considering the significant hardware and data requirements for pretraining, we decided to train *Polytune* with direct supervised learning instead. Preliminary results are provided in the Appendix, along with potential explanations for these findings.

5 Limitations and Future Work

Polytune under-performs on missed notes in homo-phonic music (see Tab. 3). We conjecture that this is due to the challenges of representing chordal textures in spectrogram form. Furthermore, the use of only one synthesizer per instrument in our dataset may restrict *Polytune*'s ability to generalize to other datasets or real-world performances with diverse timbres. This limitation could be addressed by incorporating a wider variety of synthesized timbres, which would enhance the model's robustness and adaptability. Lastly, our model's vocabulary, designed around the 12-tone equal temperament system, may limit equitable music education for non-Western cultures; future work can expand to more inclusive vocabularies.

6 Conclusion

This paper proposes *Polytune*, an end-to-end trainable transformer model for music error detection. *Polytune* outperforms previous methods without automatic alignment methods. Additionally, we introduce synthetic data generation to improve error-detection performance. We generate synthetic errors for 14 instruments to train and evaluate *Polytune* and the baseline method. *Polytune* consistently outperforms the baseline, achieving a 64.1% Error Detection F1. This represents a 40 percentage point improvement across 14 instruments. These findings demonstrate the effectiveness of our end-to-end transformer-based approach and the significant role of synthetic data in advancing music error detection.

Acknowledgments

This work is supported in part by NSF IIS-2326198, Purdue College of Engineering, Patti & Rusty Rueff School of Design, Art, and Performance. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

Akbari, H.; Chuang, W.-H.; Yuan, L.; Chang, S.-F.; Gong, B.; Qian, R.; and Cui, Y. 2021. VATT: Transformers for Multimodal Self-Supervised Learning from Raw Video, Audio and Text. In *35th Conference on Neural Information Processing Systems*.

Apaydınlı, K. 2019. Intelligent Tutoring Systems in Music Education. In *Proceedings of The 2nd International Conference on Future of Teaching and Education*. GLOBALKS. ISBN 978-609-485-043-1.

Benetos, E.; Klapuri, A.; and Dixon, S. 2012. Score-informed transcription for automatic piano tutoring. *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pp. 2153–2157.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the International Conference on Learning Representations*.

Ewert, S.; Wang, S.; and Sandler, M. 2016. Score-informed Identification of Missing and Extra Notes in Piano Recordings. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*.

Fradet, N.; Briot, J.-P.; and Chhel, F. 2021. MidiTok: A Python package for MIDI file tokenization. *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*.

Gardner, J.; Simon, I.; Manilow, E.; Hawthorne, C.; and Engel, J. 2022. MT3: Multi-Task Multitrack Music Transcription. *International Conference on Learning Representations*.

Gong, Y.; Chung, Y.-A.; and Glass, J. 2021. AST: Audio Spectrogram Transformer. In *Interspeech 2021*, 571–575. ISCA.

Gong, Y.; Rouditchenko, A.; Liu, A. H.; Harwath, D.; Karlinsky, L.; Kuehne, H.; and Glass, J. 2023. Contrastive Audio-Visual Masked Autoencoder. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hawthorne, C.; Simon, I.; Swavely, R.; Manilow, E.; and Engel, J. 2021. Sequence-to-sequence piano transcription with transformers. In *Proceedings of the 22nd ISMIR Conference*.

Hawthorne, C.; Stasyuk, A.; Roberts, A.; Simon, I.; Huang, C.-Z. A.; Dieleman, S.; Elsen, E.; Engel, J.; and Eck, D. 2018. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In *Proceedings of the International Conference on Learning Representations*.

Hsiao, W.-Y.; Liu, J.-Y.; Yeh, Y.-C.; and Yang, Y.-H. 2021. Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1): 178–186.

Huang, P.-Y.; Xu, H.; Li, J.; Baevski, A.; Auli, M.; Galuba, W.; Metze, F.; and Feichtenhofer, C. 2022. Masked Autoencoders that Listen. *NeurIPS*.

Huang, Y.-S.; and Yang, Y.-H. 2020. Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, 1180–1188. Seattle WA USA: ACM. ISBN 978-1-4503-7988-5.

JoyTunes. 2024. Simply Piano. https://play.google.com/store/apps/details?id=com.joytunes.simplypiano&hl=en_US. Accessed: 2024-08-13.

Morrison, R. B.; McCormick, P.; Shepherd, J. L.; and Cirillo, P. 2022. National Arts Education Status Report Summary 2019. Technical report, Arts Education Data Project, Quadrant Research, State Education Agency Directors of Arts Education.

Nart, S. 2016. Music Software in the Technology Integrated Music Education. *The Turkish Online Journal of Educational Technology*, 15(2).

Oore, S.; Simon, I.; Dieleman, S.; Eck, D.; and Simonyan, K. 2020. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, 32(4): 955–967.

Raffel, C.; McFee, B.; Humphrey, E. J.; Salamon, J.; Nieto, O.; Liang, D.; and Ellis, D. P. W. 2014. A TRANSPARENT IMPLEMENTATION OF COMMON MIR METRICS. *The International Society for Music Information Retrieval (ISMIR)*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*.

Sakoe, H.; and Chiba, S. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1): 43–49.

Tibshirani, R. J.; Price, A.; and Taylor, J. 2011. A statistician Plays Darts. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 174(1): 213–226.

Trommershäuser, J.; Gepshtein, S.; Maloney, L. T.; Landy, M. S.; and Banks, M. S. 2005. Optimal Compensation for Changes in Task-Relevant Movement Variability. *The Journal of Neuroscience*, 25(31): 7169–7178.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*.

Wang, S.; Ewert, S.; and Dixon, S. 2017. Identifying Missing and Extra Notes in Piano Recordings Using Score-Informed Dictionary Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(10): 1877–1889.

Wu, Y.; Gardner, J.; Manilow, E.; Simon, I.; Hawthorne, C.; and Engel, J. 2022a. The Chamber Ensemble Generator: Limitless High-Quality MIR Data via Generative Modeling. ArXiv:2209.14458 [cs, eess].

Wu, Y.; Manilow, E.; Deng, Y.; Swavely, R.; Kastner, K.; Cooijmans, T.; Courville, A.; Huang, C.-Z. A.; and Engel, J. 2022b. MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling. In *Proceedings of the International Conference on Learning Representations*.

Ying, X. 2019. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168: 022022.

Yousician Ltd. 2024. Yousician. https://play.google.com/store/apps/details?id=com.yousician.yousician&hl=en_US. Accessed: 2024-08-13.