Midi2hands
Assigning hands to piano notes

Friday 31st May, 2024

Ivar Nelson
Blekinge Institute of Technology
Blekinge, Sweden
ivne20@student.bth.se

Oscar Andersson
Blekinge Institute of Technology
Blekinge, Sweden
osan20@student.bth.se

***Abstract*—Recent advancements in Automatic Music Transcription (AMT) have significantly enhanced our ability to extract musical notes from raw audio recordings. Despite these improvements, the manual effort required for transcribing music remains substantial. This study focuses on automating the process of assigning hand positions to piano notes using both discriminatory and generative models. We utilize a bidirectional LSTM and a Transformer model to predict which hand plays each note in a MIDI file. Our research aims to determine if a generative approach can outperform a discriminatory model in this context and how varying window sizes affect performance. We employ a dataset of 122 piano performances, applying global time encoding methods to preprocess the MIDI data. Our findings suggest that while generative models can capture more natural hand movements, they are prone to cumulative errors. In contrast, discriminatory models tend to produce less natural predictions but exhibit fewer cumulative errors. The study highlights the need for larger datasets to better assess the comparative performance of these models and further investigates the influence of window size on model accuracy.**

I. Introduction

Recent advancements in the field of Automatic Music Transcription (AMT) have significantly propelled our understanding and capabilities in extracting musical notes from raw audio recordings [3][4].

Without AMT this process inherently demands meticulous manual effort, and automation plays a crucial role in translating auditory music into tangible formats that musicians can utilize for practice and performance. This transcription can manifest in various forms, such as traditional sheet music or more modern video formats like Synthesia, where visual cues distinguish between the left and right hand parts. For sheet music, it is essential to appropriately label notes on either the upper or lower staff depending on the hand, while video formats typically use color coding to indicate the same.

A. Related work

This study builds on previous work that implemented Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and a Kalman filter in AMT systems to assign the hand that is to play a certain note [1].

Notably, this prior research also contributed a dataset that includes 122 different piano performances. The data was collected from piano students who were tasked to record the music on two separate keyboards positioned one atop the other, simultaneously, thereby having one stream for each hand.

B. Aims and Objectives

Our overarching goal is to automatically produce instructions on how to play a certain piano piece. In order to produce well written notes for piano playing it is important to know which hand is supposed to play what note and that is the objective of this work. The aim is therefore to create and train a model that takes in a midi file containing piano music and assign each note with a prediction of the hand that is supposed to produce it.

After some initial experimentation with a discriminate model, we noticed that the model was unable to overfit the training set.

This sparked the idea of treating the problem as generative. In other words, for a sequence of notes, generate assignments that are likely to come from the underlying data distribution where the assigned hand is conditioned both on the series of notes *and* what hand played them.

We are not aware of any other works that use this idea. Following this line of thought, we formulate the following research questions

1) Research questions:

RQ1: *Can the performance be improved by using a generative model instead of a discriminatory model?*
RQ2: *How does the window size affect performance?*

RQ1 will be answered by training both a bi-directional LSTM and a transformer model in both a discriminatory and generative setup. The generated hand assignments will then be measured against the known hand assignments of a selection of performances, as they were conducted. As these performances

only represent one variation of how a piece may be played we will also visualize and analyze the predictions. Accuracy alone may not be the perfect measure of performance and it is possible that the various models produce different types of errors. Furthermore, 10-fold cross validation will be used to combat the effects of the limited size of the dataset and aforementioned variation.

RQ2 will be answered by training our best performing model on a range of window sizes from 8 to 256 notes and assessing the performance. There is no need to use cross validation as we are interested in the relative performance shifts.

II. Data Representation

The MIDI format is a standard protocol used to represent musical information in a digital form. MIDI files encapsulate various elements of musical performance, including the notes played and duration, velocities, and other expressive nuances. Each MIDI file consists of a series of events that detail the actions taken during a performance, such as pressing and releasing keys, control changes, and more.

To prepare MIDI data for AMT, we parse the MIDI files to extract essential features that our model will use. This preprocessing step involves converting the raw MIDI events into a structured format suitable for machine learning algorithms.

A. Overview of Data Features

The primary features considered in this study include pitch, start time, end time, and velocity:

1) Pitch: Refers to the specific note played, among a possible 88 on a standard piano.
2) Start Time and End Time: Indicate when a note begins and ends, crucial for determining the duration of a note.
3) Velocity: Measures the intensity with which a note is played, providing dynamic nuances.

B. Encoding Time Features

The encoding of time in data representation can significantly influence the performance of AMT models. We explore two primary methods:

1) Global Time Encoding: Recent studies, such as by [2], emphasize the importance of global time positioning for sequence-to-sequence models. This method involves marking the absolute timing of events from the start of a piece, which helps the model understand the temporal context of each note within the entire composition.

2) Relative Time Encoding: Used in previous studies, this method marks the time since the last noted event. This approach focuses more on the immediate succession of notes and can be beneficial for models that need to capture rapid transitions or rhythmic patterns.

In this study, while previous work has often utilized relative time encoding, we adopt a global timing approach, calculating the time of each event as a percentage from the start of the window. This change is motivated by the need to align our methodology with recent successful applications in sequence-to-sequence AMT models.

Another way to phrase this is that our objective is to present as direct data dependencies as possible in the features. That way relationships are easier to represent for the model as it does not have to model unnecessary relationships. If we consider the absolute and relative time representation the model would, in the latter case, need to learn to make a cumulative sum of times to know where they are in relation to each other.

C. Feature Representation and Normalization

Deciding on the optimal representation of the remaining features, namely pitch and velocity, is another critical consideration. The options include:

- One-hot Encoding: Each feature is represented as a binary vector where only the index corresponding to the feature's value is marked as 1, and all other indices are 0. This method is particularly useful for categorical data like the pitch of a piano which has 88 discrete key.
- Numeric Representation: Features can also be represented as single numeric values, particularly for features like velocity, which inherently possess a quantitative nature.

Since pitch and velocity are continuous variables that are discretized, both approaches are viable options. One possible drawback of using one-hot encoding may be that it unnecessarily raises the dimensionality of the feature and we have therefore opted for the numerical representation. There are 127 different values for pitch in the midi format, however the piano only has 88 playable notes. Therefore, we subtract a constant factor from each midi note and then divide by 88 to have the pitch of the notes in the midi file map to the keys of a piano in a normalized fashion.

D. Data Slicing Strategy

Considering the temporal dynamics of piano performances, we choose to slice the data by the number of events rather than time. This method ensures that the model consistently analyzes segments with equal amounts of musical information, at least as it relates to hand movements as each event correspond to a pressing of a key. Using this method will likely result in the model learning longer dependencies.

III. Modeling

There are many ways to play any piano piece and it is problematic to "average" the prediction as there is a physical limit to how fast a pianist can move her hands. A discriminatory model predict y given x for each time step and only operates on the sequence of notes and their respective features to produce a prediction as shown in figure 1.

A generative model predicts y given x and a some y. This way we take in to account what hand played some notes when we assign the next hand to a note. By using this approach it is possible for the model to commit to one particular variation of playing instead och averaging the variations it has seen in training.

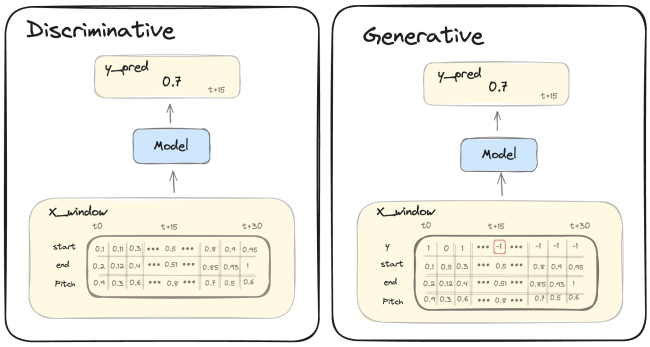

Fig. 1. Discriminatory way of modeling the problem (left). Generative way of modeling the problem (right).
At inference, the window is populated with previous predictions. Both methods use the middle time-step as target. The generative model is padded with -1 to indicate a future value or padding outside the window

IV. Windowing and Inference Strategy

Given the complexity and length of piano performances, which can consist of thousands of musical events, an effective strategy for processing and predicting hand assignments for each note involves windowing the data. This method divides the lengthy input sequences into manageable segments, allowing the model to focus on a smaller subset of notes at any given time, which is essential for both training efficiency and prediction accuracy.

A. Single Output per Window

One approach is to predict the hand assignment for only the middle note in each window. This strategy leverages the context provided by preceding and succeeding notes within the window, particularly when utilizing bi-directional models such as Bi-LSTM. Here, each window slide results in a single prediction, focusing on the note that benefits from the maximum available context on both sides. This approach is efficient during training as it simplifies the output structure and reduces computational load by minimizing the number of predictions made. One drawback of this approach is that we can not run inference for an entire piano piece but would miss a few notes in the beginning and the end unless some kind of padding is used. Regardless there will be less information available at the beginning and end of a piece.

B. Sequence to sequence prediction

A sequence-to-sequence model can be employed to generate predictions for every note within the window. It takes in a sequence of conditionals and outputs a sequence of predictions of the same length. This method is particularly advantageous for maintaining continuity and consistency in predictions across consecutive windows. By predicting every time-step within the window, the model provides a dense prediction output that captures more detailed nuances in hand assignments. This approach is well-suited for models that incorporate attention mechanisms, such as Transformers, which can effectively manage dependencies between all notes in the window.

C. Generative predictions

In order to condition the assignment of what hand is to play a certain note as well as the notes that have been, and will be, played a padding structure was put in place. In the beginning and end of a piano piece we pad the list of midi events and labels with dummy tokens. This padded piece is then windowed and the windows are fed to the model during training. The model is shown a sequence of notes and some of them have a hand assigned to them while others do not. For each of these windows the model predicts the hand of the note in the middle of the window. This configuration was selected because then we utilize knowledge of both what has been, and will be, played.

During training the model consumes these masked windows and during inference the model starts off with a completely masked label sequence. After the first label has been assigned the following predictions will be fed this assignment. Thereby we condition the prediction of the hand not only on the notes but also on prior assignment of those notes.

D. Inference Strategy

During inference, the choice of windowing technique significantly impacts the performance and utility of the model. For instance:

- Sliding Window Inference: For the single-output strategy, a sliding window can be moved note by note, ensuring that every note in the sequence receives a prediction when it reaches the central position of the window. This ensures continuous output but at the cost of increased computational effort during inference.
- Overlap and Average: In the sequence-to-sequence approach, overlapping windows can be used where the predictions for overlapping notes are averaged (or otherwise combined) to smooth out discrepancies and improve the overall prediction consistency. This method is particularly beneficial at the beginning and end of a song, where boundary effects might otherwise introduce prediction errors.

V. Models

This study employs two different model architectures; Bidirectional LSTM and a Transformer. For a detailed description of the models and hyper parameters, readers are encouraged to refer to the provided code. Both models respect the same constraints. They both input a window with $window\_size$ many events, and predict the hand assignment of the middle event.

A. LSTM

We utilize a bidirectional LSTM architecture consisting of two layers in each direction, taking in to consideration both the notes that has been, and will be, played. The LSTM outputs are passed through two fully connected layers to make the dimensionality reduction smooth and produce the final output.

B. Transformer

The model consists of an encoder-decoder structure, leveraging multi-head attention to capture relationships between time steps effectively. A simple linear layer is employed for embedding the input features, without the need for additional positional encoding, as the temporal aspects of each note are already represented in the input features. While the transformer outputs hidden states for the entire sequence only the middle embedding is used. This is then projected by a fully connected layer to predict the output hand assignment.

C. Evaluation

We assess the performance of our models using accuracy as the primary metric due to its direct reflection of the model's ability to correctly predict the hand assignments for each note. In this particular problem, there is no preferred false positive or false negative rates. In some cases, there is no clear distinction of which hand is should to play the note. Instead of using the mean accuracy of all the windows produced by the pieces in the validation set we opted to use the average of the mean accuracy for each piece. The rationale being that each piece is equally important, regardless of it´s duration.

We opt to split the dataset based on individual songs rather than windows. This approach prevents the model from being tested on data that is too similar to its training set, thus providing a more realistic gauge of its performance on unseen data. However, we noticed that unique characteristics of each song, such as tempo, complexity, and style, was likely to influence the performance of the model. We therefor employ a 10-fold cross-validation method. This method provides us with a comprehensive overview of the model's effectiveness. By measuring the average accuracy across the folds, we obtain a better estimate of the true performance. We also calculate the standard deviation to estimate the models performance.

In addition to quantitative metrics, we also conduct a visual evaluation with three examples not included in the training, using a custom-built MIDI visualizer. This tool allows us to color-code the notes based on the predicted and actual hand assignments, enabling a direct visual comparison. Visual evaluations help identify any systematic errors or biases in the model's predictions, such as consistently misclassifying certain types of notes or patterns. This method is particularly useful for qualitative feedback and helps in further refining the model.

VI. Result

In table I we show the group mean results and standard deviation of our four models for each fold. Figures 2 and 3 in the appendix illustrate the qualitative differences in the way the models work. The generative models appear to capture a more natural movement of the hands but are prone to cumulative errors whereas the discriminatory models are less natural but do not exhibit the same cumulative errors.

Table II show the group mean accuracy and standard deviation for the piano pieces in the first fold at various window sizes. Apart from where the window size is 64 the standard deviation for the generative model decreases whereas only minor performance changes are visible for the discriminatory model.

	Discriminatory		Generative	
Fold	BiLSTM	Transformer	BiLSTM	Transformer
0	0.939 ± 0.037	0.932 ± 0.039	0.938 ± 0.045	0.909 ± 0.074
1	0.941 ± 0.038	0.948 ± 0.034	0.914 ± 0.041	0.915 ± 0.047
2	0.892 ± 0.071	0.868 ± 0.065	0.870 ± 0.082	0.853 ± 0.068
3	0.919 ± 0.056	0.907 ± 0.058	0.910 ± 0.056	0.835 ± 0.070
4	0.925 ± 0.042	0.907 ± 0.039	0.922 ± 0.044	0.867 ± 0.069
5	0.943 ± 0.029	0.944 ± 0.024	0.922 ± 0.032	0.841 ± 0.146
6	0.940 ± 0.065	0.925 ± 0.081	0.917 ± 0.085	0.846 ± 0.192
7	0.894 ± 0.055	0.872 ± 0.060	0.895 ± 0.064	0.861 ± 0.081
8	0.921 ± 0.068	0.898 ± 0.063	0.888 ± 0.097	0.874 ± 0.101
9	0.901 ± 0.080	0.904 ± 0.077	0.871 ± 0.129	0.859 ± 0.106
tot	**0.922 ± 0.060**	0.910 ± 0.063	**0.905 ± 0.076**	0.866 ± 0.107

TABLE I
PERFORMANCE METRICS FOR EACH FOLD. THE VALUES ARE PRESENTED AS GROUP MEAN ± STANDARD DEVIATION OF THE ACCURACY ON THE PIANO PIECES IN THE 10 FOLD CROSS VALIDATION. NOTE THE LARGER STANDARD DEVIATION FOR THE GENERATIVE MODELS.

	BiLSTM	
Window Size	Discriminatory	Generative
8	0.922 ± 0.063	0.920 ± 0.050
16	0.937 ± 0.045	0.937 ± 0.049
32	0.938 ± 0.042	0.937 ± 0.043
64	0.942 ± 0.038	0.907 ± 0.108
128	0.939 ± 0.040	0.924 ± 0.055
256	0.940 ± 0.041	0.915 ± 0.048

TABLE II
THE INFLUENCE OF WINDOW SIZE FOR A DISCRIMINATORY AND GENERATIVE BILSTM. SHOWING GROUP MEAN ACCURACY AND STANDARD DEVIATION FOR THE PIANO PIECES IN THE VALIDATION PART.

VII. Discussion

The figures in the appendix illustrate typical errors and how they differ between discriminatory and generative models. Figure 2 show an example of an misprediction that is common for the discriminatory models. The model does not take into account the movement of the hands and injects a few notes to played by the left hand when it clearly should be the right hand.

Again, in figure 3 we see an unnatural movement with the discriminatory model injecting two left hand assignments into a sequence of notes that belong to the right hand. The generative model incorrectly switches hands but then remains committed to this style. This signals that the generative model has picked up a concept that the discriminatory model has not been able to learn, namely that a pianist has a limit as to how fast she can move her hands and that a natural style does not unnecessarily switch hands.

However, the physical distance between concurrent notes is a feature that the model seems to struggle to grasp. It is possible that augmenting the data with a representation of their physical position in addition to their placement in pitch would give the model an impulse to not assign hand positions that are overstretched. However, larger hands can comfortably manage wider spans and more complex chord

structures without needing to switch hands or adjust positions frequently.

In table I it is evident that the standard deviation of accuracy of the generative models is larger than that of the discriminatory models. However, the mispredictions are qualitatively different and while a variation chosen by the model is not the same as the one that was recorded it is not implausible whereas the discriminatory error is more severe.

Providing a straight answer to **RQ1** is difficult. The dataset is simply too small to be able to capture the many acceptable variations on how to play a certain piano piece. The discriminatory BiLSTM has the highest accuracy and lowest standard deviation of all the models but as evident from the figures in the appendix sometimes produce unnatural predictions. Another strong candidate is the generative BiLSTM but since it carries bad predictions onwards one single incorrect assignment can produce a bad overall accuracy.

A dataset with more variations could allow for the possibility to assess the best match between a prediction and ground truth. In this setting it is possible that the generative model would not be as penalized.

To assess **RQ2** the metrics in table II provide some insight although further investigations are necessary. The accuracy of the generative model stabilizes as the window size goes up, except for a window size of 64, indicating that a a longer window gives the model a better opportunity to capture a plausible hand movement. However, a larger window also allows for compounding errors to be carried further which may be why the average accuracy decreases.

One possible explanation for why the performance of the discriminatory model is less influenced by window size is that there is no need for very long sequences in order to capture what hand is to play what note. Simply put; it does not matter which hand that played what note two hundred notes ago.

Furthermore, using larger window sizes may require the expressivity of the models to be revisited as they need to hold more information.

To conclude we need larger and more complete datasets to use accuracy as a fair metric when comparing discriminatory and generative model although generative models show promising results. Additionally a more comprehensive study of the interplay of window size and model architecture needs to be conducted.

References

[1] Aristotelis Hadjakos, Simon Waloschek, and Alexander Leemhuis. “Detecting Hands from Piano MIDI Data”. In: (2019). DOI: 10.18420/muc2019-ws-578.

[2] Curtis Hawthorne et al. *Sequence-to-Sequence Piano Transcription with Transformers*. July 19, 2021. arXiv: 2107.09142[cs,eess]. URL: http://arxiv.org/abs/2107.09142 (visited on 03/31/2024).

[3] Qiuqiang Kong et al. *High-resolution Piano Transcription with Pedals by Regressing Onset and Offset Times*. July 31, 2021. arXiv: 2010.01815[cs,eess]. URL: http://arxiv.org/abs/2010.01815 (visited on 03/31/2024).

[4] Keisuke Toyama et al. *Automatic Piano Transcription with Hierarchical Frequency-Time Transformer*. July 9, 2023. arXiv: 2307.04305[cs,eess]. URL: http://arxiv.org/abs/2307.04305 (visited on 03/31/2024).

APPENDIX

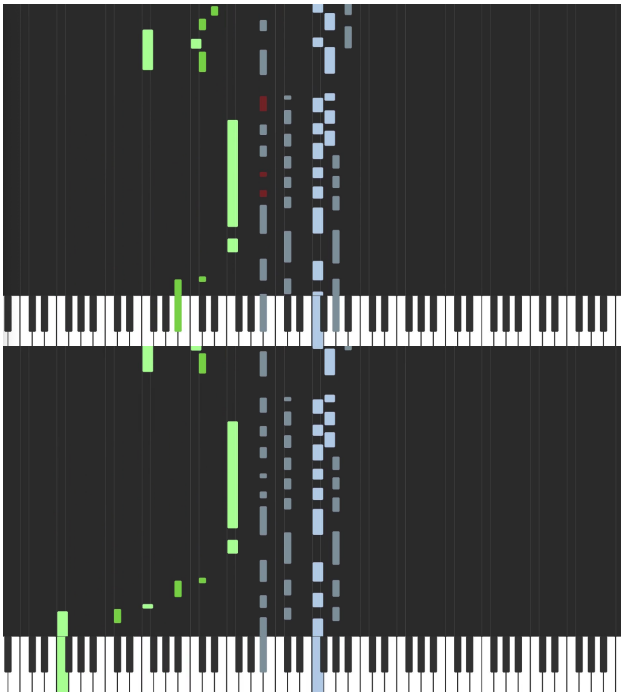

Fig. 2. Part of a piano piece. Top image is predictions made by the discriminatory model and the bottom by a generative model. Notes highlighted in red are incorrectly assigned.
The discriminatory model swaps hand for a sequence of the same note.

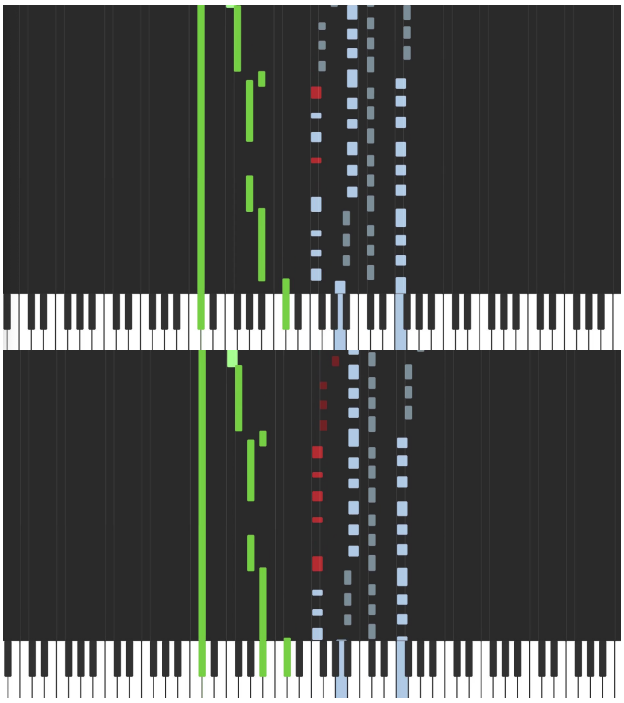

Fig. 3. Part of a piano piece. Top image is predictions made by the discriminatory model and the bottom by a generative model. Notes highlighted in red are incorrectly assigned.
Both models make incorrect predictions; however, the generative model provides more consistent hand placement, although it tends to result in a somewhat stretched hand position.