Exploring Recurrent Neural Network Architectures for Multi-Modal Score Following

Sally Ma
sally.ma@yale.edu
Yale University

Annie Gao
annie.gao@yale.edu
Yale University

William Hu
william.hu@yale.edu
Yale University

ABSTRACT

As more and more musicians store their sheet music in portable electronic devices, automatically turning pages of music scores has become an useful functionality with wide reach. Score following, which seeks to allow a computer program to track the position in the music sheet score based on audio input, is key to a robust automatic page turning application. In this project, we study the effectiveness of using recurrent neural networks (RNNs) for the task of score following. We preprocess the Multimodal Sheet Music Dataset (MSMD), which contains 794 pieces of synthetic polyphonic classical music with fine-grained alignment between noteheads in sheet scores and their corresponding audio onsets in the spectrogram. We then experiment with three RNN architectures: Simple Recurrent Network (SRN), Long Short-Term Memory network (LSTM), and Gate Recurrent Network (GRU). The networks take in a sheet music and a window of audio excerpt, and outputs an estimated position in the sheet score. We implement visualization of the model's performance tracking a score compared to the target. We find that the LSTM performs the best out of the three RNN architectures, and a penalty preventing big jumps in the model's predicted positions improves the model's prediction. We also find that running models on longer songs leads to worse performance, which suggests that additional constraints are needed to guide recurrent networks through long sequential data for score following.[1]

1 INTRODUCTION

A frequent problem that musicians face is flipping the pages of their music scores. Since almost all musical instruments actively engage the hands, it becomes inconvenient for musicians to manually flip their sheet music as they play, especially when the music is complex and fast. Although many performers use human page turners or foot clickers, these are not always accessible to everyone and can be expensive.

The problem we plan to investigate is score following, which is intimately related to automatic page turning. Score following is an open research problem that seeks to allow the computer to track a musical performance in the form of audio with a corresponding score representation. A robust score following algorithm will enable the music page turner to turn the page at the right time, as it listens to musical input.

Recurrent neural networks have shown potential in learning from time series data and are therefore well-suited to this task. Our main approach will be building and experimenting with various RNN architectures, mainly convolutional RNNs, LSTMs, and transformers, which will allow the model to remember information from previous time steps to make a more accurate prediction of its current location in the music score. In addition to the hidden states, the network input includes the music score and the audio excerpt, and the output is just the score position that corresponds to the audio. The objective is to make the model's output position as close to the target output position as possible.

The most recent work on score following leverages machine learning to learn a direct mapping between sheet music and audio. Henkel et al. uses a multimodal deep learning method of creating an audio-conditioned U-Net for position estimation in full sheet images [2]. Dorfer et al. uses a deep reinforcement learning (RL) algorithm, which includes multimodal RL agents that simultaneously learn to listen to music, read the scores from images of sheet music, and follow the audio along in the sheet, in an end-to-end fashion [4]. In another paper, Dorfer et al. uses multimodal convolutional neural networks to learn joint embedding spaces for short excerpts of audio and their respective counterparts in sheet music images [3].

There hasn't been prominent prior work on using RNNs for score following. Due to the time dependent nature of the task, RNNs are well-suited for score following, and we therefore explore their potential.

2 DATASET

The primary dataset we will be using is the Multimodal Sheet Music Dataset (MSMD), which is a synthetic dataset of 497 pieces of classical music that contains both audio and score representations of the polyphonic classical piano pieces aligned at a fine-grained level (344,742 pairs of noteheads aligned to their audio/MIDI counterpart) [1].

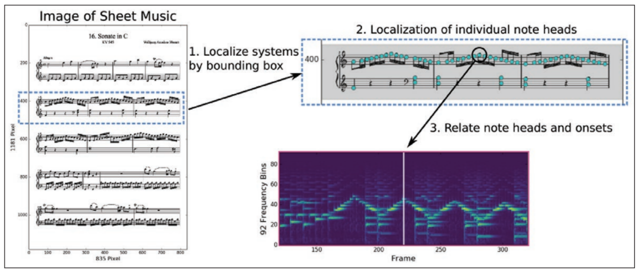
Image of Sheet Music
16. Sonate in C
1. Localize systems by bounding box
2. Localization of individual note heads
3. Relate note heads and onsets
1181 Pixel
835 Pixel
92 Frequency Bins
Frame

Figure 1: Workflow for the MSMD dataset. The note coordinates, MIDI events, and audio are aligned at a fine-grained level to allow for multimodal learning [1].

MSMD is structured into abstract musical entities encoded with a LilyPond file extracted from Mutopia [2]. The abstract musical

[1]Our code repository is online at https://github.com/xm83/AutoPageTurner/tree/rnn_folder/version_2/experiments/rl_score_following_game/score_following_game.

[2]Mutopia is an open-source collection that contains 2099 pieces by 317 composers at https://github.com/MutopiaProject/MutopiaProject/.

entities have their corresponding MIDI files and are embodied both as scores, the visual modality, and performances, the audio modality. The data are augmented with varying tempo. Fluidsynth [3], a cross-platform soundfont, generates audio files from the MIDI encoding of a performance, and the audio yields the spectrogram. Several features of a performance are also available:

(1) The MIDI matrix: a frame-wise feature with the frame rate set to 20 frames per second. The rows are pitches and the columns are frames. If a given pitch is active in a given feature at a frame, that matrix cell contains is 1;
(2) Note events list: a Numpy array derived from the performance MIDI, where the rows are the each note event, and the columns are: onset time (in seconds), pitch, duration (in seconds), track, and channel.
(3) Onsets list: an array mapping note events to onset frames, where length = number of note events
(4) Spectrogram: computed from the synthesized audio with a sample rate of 22050 Hz. The rows are frequencies (representing pitches) and the columns are the number of frames.

The scores are based on PDFs generated by LilyPond, and have 3 representations:

(1) Images: pages of the PDF exported as images with a predefined width of 385
(2) Coords: the coordinates of noteheads and staff systems are extracted per page
(3) MuNG: records the locations of the noteheads and staff systems, where notehead corresponds to which staff system, and the cross-modality alignment. MuNG, short for MUSCIMA++ Notation Graph [4], is a format that explicitly records how noteheads are grouped by staff systems, and how visual elements are aligned to their counterparts in the performances.

For our purposes, we have pre-processed the data into a format that is suitable for feeding into the recurrent neural network: the dataset is an array of arrays of tuples, where each inner array contains all the tuples for one song (i.e. music piece), and each tuple contains 3 elements: the audio excerpt for the current performance frame, the whole unrolled score, the normalized score position corresponding to the current performance frame, in a sequential manner.

3 TECHNICAL APPROACH

We propose various recurrent network architectures for score following and automatic page turning. Unlike the reinforcement learning algorithm proposed in [4], we attempt to build supervised-learning models that directly predict score location from the performance audio and score representation. In particular, we are interested in simple recurrent neural networks (SRN), gated recurrent unit (GRU) networks, and long-short term memory (LSTM) networks. We train each method on songs from the MSMD dataset and evaluate their performance for score following.

[3]http://www.fluidsynth.org
[4]https://github.com/hajicj/muscima

3.1 Data Pipelining

In our experiments, We used the same data pipelining structure as the implementation for [4], which loads and processes paired scores and performances into various structured arrays. Since we are not using a reinforcement learning approach, we modified the pipeline to better accommodate supervised learning in the following ways:

(1) We added a function to prepare a dataset from the loaded score and performance files that better fits into the different structure of our network. This function takes in the shared cache and splits each song into segments with corresponding snippets of the score and performance, as well as the interpolated true target position in the score. We played around with the size of the cache but found that small cache sizes, such as 5, led to overfitting. For our final experiments, we used a cache size of 15 songs.
(2) Since the songs have variable length, we normalized the target score positions by the length of each song to avoid scaling problems with our network loss function and to encourage the network to output values between 0 and 1.
(3) In each training epoch, instead of looping through the entire training dataset, we feed the network the paired data of a random subsample of songs. We reset the hidden sets after every song in hopes of exploiting the sequential nature of score-following, while not confusing the network between song switches, when the score position resets to zero.

3.2 Model Architecture

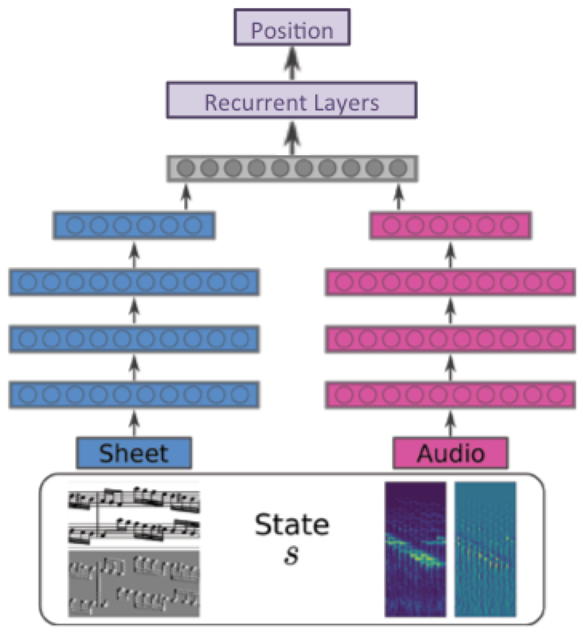

Figure 2: Multi-modal network architecture for supervised learning. Given state *s*, the network outputs the estimated position on the score.

We modify the network architecture used in [4] but preserve the preliminary structure to handle multi-modal input until the two input branches are merged into the centralized portion of the network. At this point, we remove the connections to the policy and critic networks for reinforcement learning, and add the appropriate layers to make a SRN, GRU, and LSTM. Instead of outputting two values relating to the velocity of the agent's estimation, we output a single value representing the interpolated, normalized estimated position on the score. A diagram of our general architecture is shown in Figure 2, and the specific layers in our network are displayed in Figure 3. For the architecture of the recurrent layers, we found through experimentation with different numbers of recurrent layers and different hidden layer dimensions that using 2 recurrent layers with a 1024-node hidden layer produced the best results.

The model is evaluated using a Mean Squared Error loss, coupled with a weighted penalty term for jumping to encourage smoothness:

$$\epsilon_t = (p_t - \hat{p}_t)^2 + \lambda_{jump}(\hat{p}_t - \hat{p}_{t-1})^2 \quad (1)$$

where p_t is the true, normalized target position at timestep t, $\hat{p}_t$ is the corresponding prediction by the network, $\hat{p}_{t-1}$ is the network's prediction from the previous timestep, and λ_{jump} is the weight of the jumping penalty; we found that setting $\lambda_{jump} = 0.2$ achieved the best performance. We decided to use this squared error method over other nonlinear methods because our objective function ultimately seeks to reduce the distance between the target and the prediction.

Audio (Spectrogram) 78 × 40	Sheet-Image 80 × 256
Conv(3, stride-1)-32	Conv(5, stride-(1, 2))-32
Conv(3, stride-1)-32	Conv(3, stride-1)-32
Conv(3, stride-2)-64	Conv(3, stride-2)-64
Conv(3, stride-1)-64 + DO(0.2)	Conv(3, stride-1)-64 + DO(0.2)
Conv(3, stride-2)-64	Conv(3, stride-2)-64
Conv(3, stride-2)-96	Conv(3, stride-2)-64 + DO(0.2)
Conv(3, stride-1)-96	Conv(3, stride-2)-96
Conv(1, stride-1)-96 + DO(0.2)	Conv(1, stride-1)-96 + DO(0.2)
Dense(512)	Dense(512)
Concatenation + Dense(512)	

Figure 3: The layers of the model that combine the multi-modal input. Here, DO indicates a dropout layer. The recurrent layer for each network is added after the Concatenation + Dense layer. Modified from [4].

4 RESULTS

4.1 Training

For our experiments, we trained SRN, GRU, and LSTM models using 1024 hidden dimensions, 2 recurrent layers, and a 0.2 jump penalty over 50 epochs. As shown in Figure 4, the GRU and LSTM models improve significantly during training, while the RNN model seems to saturate relatively early on. The LSTM model appears to learn faster than the GRU, likely because of the LSTM's better long-term memory, which can store longer range dependencies in the data. Overall, the LSTM achieves the lowest training loss at epoch 20, and likewise, the LSTMs perform the best in the visualizations.

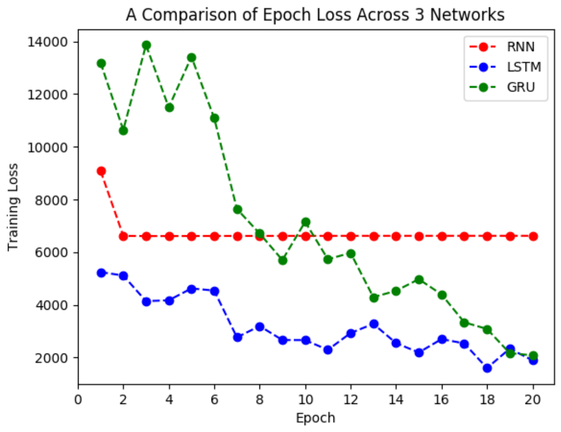

Figure 4: A comparison of learning curves for each of the three architectures. As can be seen, the RNN fails to learn and its loss stays consistently high throughout training. The LSTM finishes with the lowest loss and reaches lower losses more quickly than the GRU does.

4.2 Visualization

To get a better sense of how each model was performing, we modified the visualization approach used in [4]. Instead of receiving action selections and state-value estimates from the reinforcement learning framework, we receive the estimate of the normalized position in the score, un-normalize it to scale it back with respect to the length of each song, and render a video showing excerpts from the score, spectrogram, and the predicted and target positions on the score. Figure 5 shows an example.

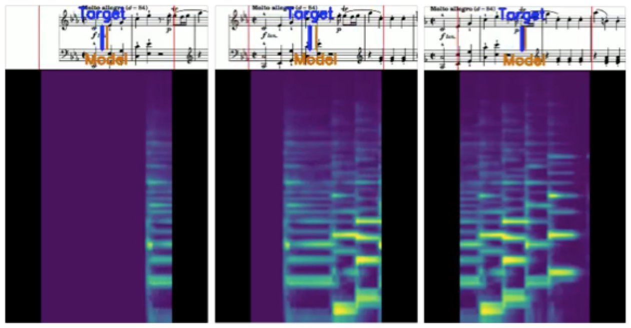

Figure 5: A visualization of our trained model (orange) and the target position (blue) on Mozart.

4.3 Limiting Song Length

We experimented with numerous combinations of hyperparameters to tune each model. For most of the models we trained, the training loss curve is similar to that shown in ??. For the convolutional LSTM network, which had the lowest training loss out of all the different architectures we tried, using the full length of each song resulted in the model converging to the middle of the song, even in training. This indicated a failure to learn to follow the score, and we hypothesized that the songs, which could be comprised of upwards of 300 data points each, were way too long for the network to digest. Therefore, we tried to limit each song to a fixed number of data points with the hope that this would allow the network to better fit the training data. We found that limiting each song to about 300 data points allowed the network to perform very well on the training set; when visualized, the predictions follow a smooth path through the song, staying within the rendering window for most of the excerpt. Anything larger than this amount of data points seemed to overwhelm the network, and the predictions would go back to converging in a single point of the song.

4.4 Model Fit

While we were able to achieve reasonable performance for short excerpts in the training data, when the models were exposed to the same length of unseen data, they were unable to generalize and again hovered around a narrow section of the score. When we increased the cache size to introduce more training examples, the behavior on unseen data was not significantly different. We conclude that the model was unable to learn to follow the score and had instead memorized a path that minimized the average cost across the different songs. This ties back to the notion of velocity proposed in [4] as a more appropriate output than position for the network, which seems to have limited performance when trained with a supervised approach.

5 CONCLUSION AND FUTURE WORK

We study the effectiveness of using recurrent neural networks for the task of score following. We preprocess the MSMD dataset into score and audio excerpts for multi-modal training of the RNNs. We also implement visualization of the models' performance and find that the LSTM performs the best out of the three RNN architectures (SRN, GRU, LSTM), and a penalty for jumps in predicted positions improves model prediction. Our results show that the model achieves reasonable score-following on shortened excerpts of the training data, but is unstable on full songs and does not generalize well to unseen data. This suggests that to achieve greater stability and generalizability, additional constraints are required to train the recurrent neural networks on long sequences of data.

For future work, we would like to add a validation step to the training process. We did not expect the models to overfit to the training data so quickly because each epoch consists of five to fifteen different songs with various tempos and rhythms. However, through our experiments, we believe that validation can simplify steps like hyperparameter tuning and model selection.

In addition, more computational resources are needed for better model training. Our access to GPUs for computation is limited to only 24 hours at a time. As a result, we were unable to use a large training dataset or train for larger number of epochs. More computational resources would either result in better hyperparameter tuning and lead to a better score following model, or allow us to state our hypotheses with stronger evidence. Additionally, once we have a reasonable score following model, it will be valuable to train and test the model on real-time performance rather than pre-recorded, synthetic music. Incorporating real-time performance is essential to the application of the score following model to an automatic page turning program.

Furthermore, a potentially better alternative to our model inputs is that instead of feeding in a score excerpt, we could feed in the entire score. The potential benefit of this alternative is that the model might learn a more global pattern rather than learning from an excerpt at a time step during training. An additional benefit is that this mode is closer to our ultimate goal of applying the score following model to automatic page turning. It is more realistic for the model to receive and follow the whole score in alignment with live audio input, rather than pre-segmented score excerpts.

The work presented in [4] involved a reinforcement learning agent who took in multimodal data in the form of spectrogram and score excerpts, and outputting a Δvelocity value that would adjust the agent's position in the score accordingly, as well as an expected reward given the current state. After observing the large jumps that our models' predictions were making in the score, even after meticulous training and tuning, we more clearly see the value of outputting adjustments to the agent's velocity instead of directly predicting the position, since the former results in smoother movement across the page. This also makes a RL-based framework more practical than a supervised approach, since paired training data in this manner is expensive to obtain.

REFERENCES

[1] Matthias Dorfer, Jan Hajič, jr, Andreas Arzt, Harald Frostel, and Gerhard Widmer. 2018. Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification. *Transactions of the International Society for Music Information Retrieval* 1 (09 2018), 22. https://doi.org/10.5334/tismir.12

[2] Gerhard Widmer Florian Henkel, Rainer Kelz. 2019. Audio-Conditioned U-Net for Position Estimation in Full Sheet Images. The Austrian Research Institute for Artificial Intelligence, Austria.

[3] Andreas Arzt Harald Frostel Gerhard Widmer Matthias Dorfer, Jan Hajič jr. 2018. Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification. In *Transactions of the International Society for Music Information Retrieval*. The Austrian Research Institute for Artificial Intelligence, Austria, Article 1(1), pp.22–33. pages.

[4] Gerhard Widmer Matthias Dorfer, Florian Henkel. 2018. Learn to Listen, Read, and Follow: Score Following As a Reinforcement Learning Game. In *Proceedings of 19th International Society for Music Information Retrieval Conference*. The Austrian Research Institute for Artificial Intelligence, Austria.