Advanced Wind Music Classification and Generation

Ben Rocklin (brocklin@stanford.edu), William Dunlop (wjdunlop@stanford.edu)

Video Link: https://www.youtube.com/watch?v=YdStnNLmSQM&feature=youtu.be

Motivation and Overview

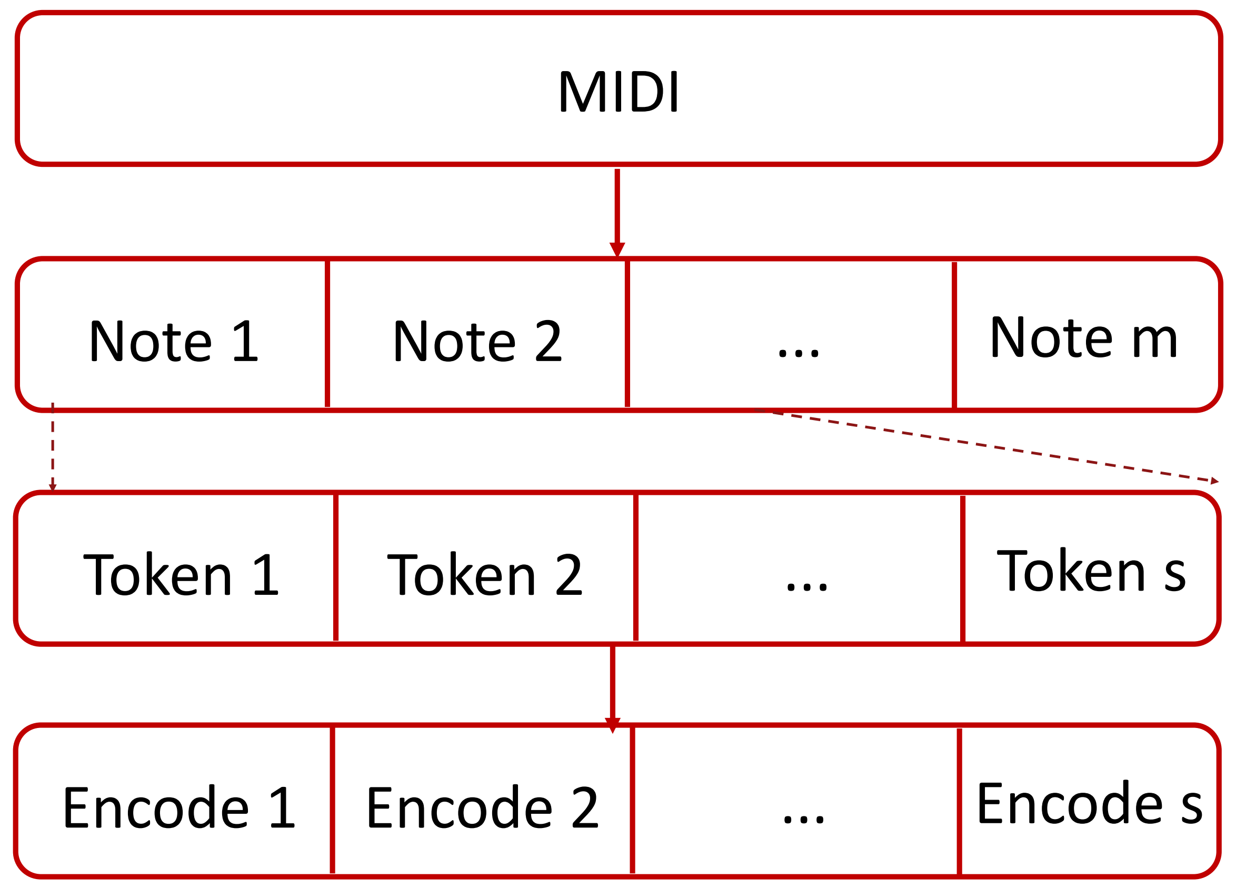
In the field of music generation, most work has focused on instruments with few physical limitations, such as piano. Fewer still have differences in note structure, generating output with identical note lengths and offsets. We applied RNN techniques to trumpet music, which has physical limitations and other features, to try and solve this. Our input data consists of trumpet note beginnings, ends, and rests. Our output data for our classifier consists of the next musical token, and our generator then creates a MIDI.

Data

We curated our own dataset of just under 2,000 trumpet melodies from www.8notes.com [1], gathering their freely available MIDI files for the trumpet and other instruments. We then convert these MIDI files into sequences of notes (more on this in the next section). Additionally, we augment this dataset by shifting the pitch of every note up by 6 MIDI pitches and down by 5 MIDI pitches, to simulate each sequence being played in every key. This yields 933,102 sequences for sequences of length 10 and 920,197 sequences for sequences of length 50.

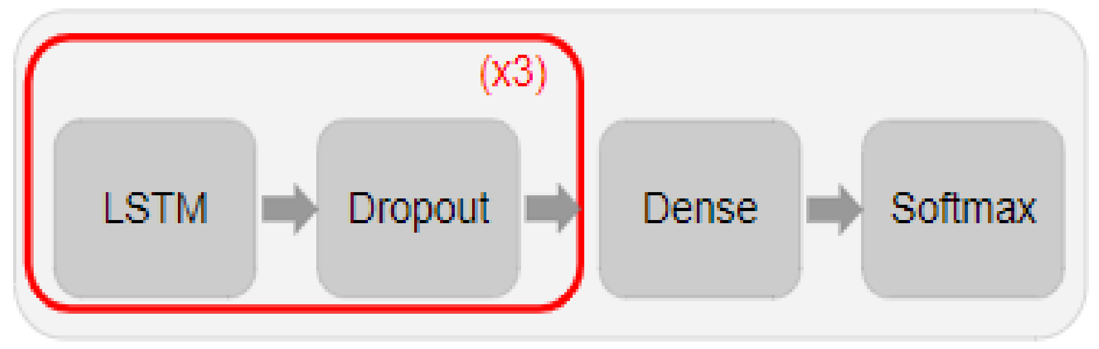
Features

Our feature vectors are composed of a series of encoded tokens of varying length, where this varying length is given by the sequence length hyperparameter. This hyperparameter was tested at length 10 and at length 50. Each of the MIDI files listed above is converted into a sequence of string tokens, such as “t88” (which represents starting to play note 88 on the trumpet), “endt88” (which represents halting a note 88 being played on the trumpet), and “wait12” (which means to wait for 12 ticks). These string tokens are then converted via a dictionary to integer values, where note beginnings are towards the start, note endings are towards the middle, and wait periods are towards the end. Outputs are converted in the same manner.



Formula and Illustrations

Model:



Categorical Cross-Entropy Loss:

$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$

Softmax:

$$g(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

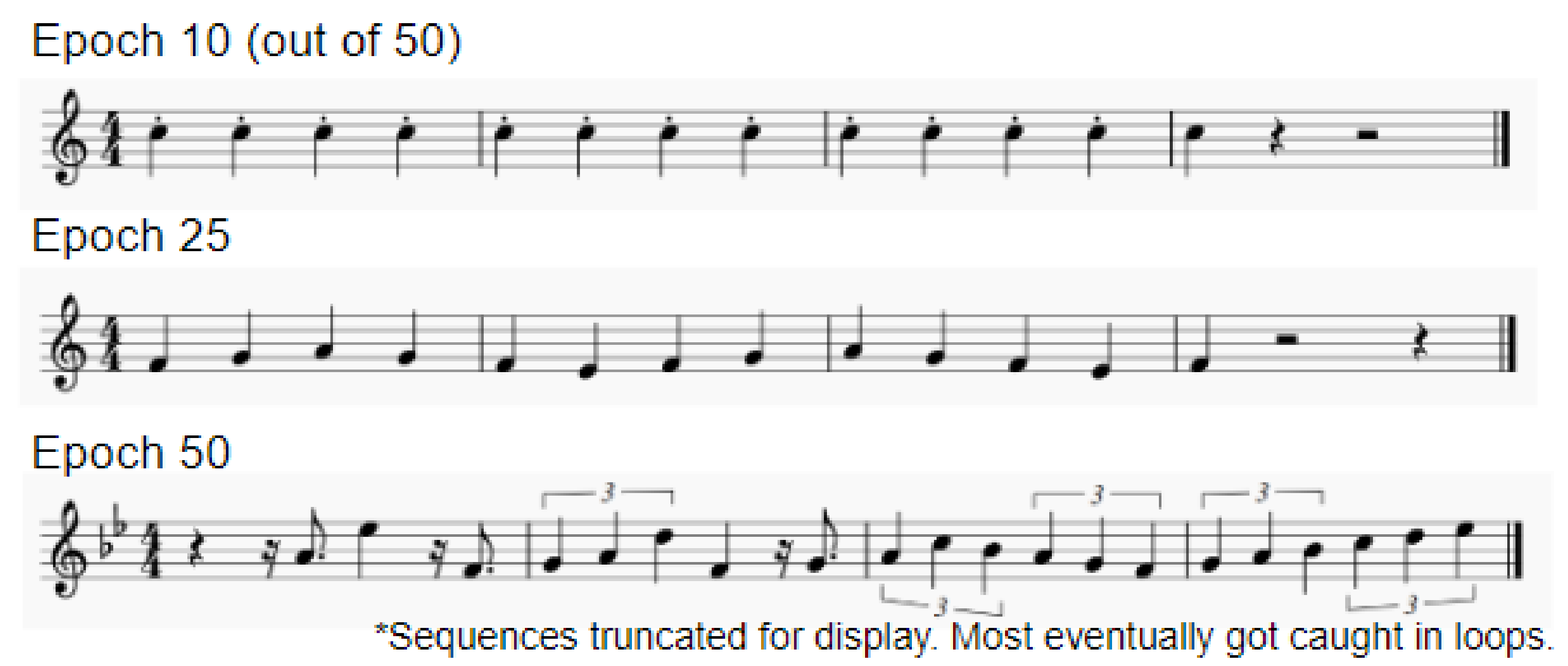
Results

As shown below, both our classification and generation algorithm reached impressive results. Our best classifier (which we subsequently used for generation) managed to achieve well over 70% accuracy for both the training and test set, although it suffered from high variance despite dropout. Subsequently, we used different saved epochs from training this model to generate music, which is shown further below and gradually increased in complexity as time trained increased.

Classification:

	Training Set Accuracy	Test Set Accuracy
Sequence size 10 w/o augment	0.6762	0.6562
Sequence size 10 w/ augment	0.6781	0.6587
Sequence size 50 w/o augment	0.7654	0.7152

Generation:



*Sequences truncated for display. Most eventually got caught in loops.

Classifier Model

Our classifier model was an LSTM-based network with dropout layers that, given an input sequence of encoded tokens, would attempt to predict the next token in the sequence for a song converted into such sequences. This was done for every sequence in every song in our dataset. The architecture can be seen to the left, and we used categorical cross-entropy as the loss function and softmax as the activation in order to pick from all possible tokens.

Generator Model

Our generator model began by selecting a random sequence from the dataset. It then generated a specified number of notes by iteratively calling our classifier model, selecting the token with the highest softmax score, and pushing the token onto a list of outputted notes and the end of the sequence. This list of outputted notes was then converted back into a MIDI file.

Discussion

Our classification results were very promising, as we did not expect to achieve over 70% accuracy on predicting the next output. Given enough time to train more powerful models, train the models we created for more epochs, and add data augmentation to the model of sequence length 50, we believe that we might have been able to surpass both the variance problem present and achieve an accuracy in the realm of 80-90%.

Additionally, our generation was a success, despite the fact that we did not have time to formally evaluate it as we initially planned. Our output music was able to create playable music for a trumpet player that respected physical limitations by avoiding large interval jumps and staying in a reasonable range. Additionally, it was able to learn rests and complex rhythms, as shown by the last generated example in the Results section, both of which many other models fail to generate.

Future

To begin, we had planned to train more complex models for longer and to try more hyperparameter combinations, but computational and time constraints limited our ability to do so. With regards to generation, we would also find a more objective metric for evaluation than simply observing results. Finally, we had hoped, given enough time, to apply this to ensemble music, perhaps using transfer learning in the process, and other genres/styles as well.

References

[1] D. Bruce, “Free Sheet Music \& Lessons,” Free Sheet Music \& Lessons, 2001. [Online]. Available: https://www.8notes.com/. [Accessed: 15-Mar-2020].