Real-time prediction of MIDI sequences for a collaborative drumming application

Darren Sholes
ATLAS Institute
University of Colorado Boulder
Boulder, CO 80309
dash7575@colorado.edu

Abstract

Network latency and jitter are major concerns for real-time remote jamming applications. Four algorithms are compared for a sliding window regression task that aims to predict MIDI drum sequences for an augmented reality, remote jamming application. While the simple algorithms perform significantly better than chance on accuracy of note prediction, the generated drum "performances" lack the variability and expressiveness of the human performed ground truth. Reducing bias in the training examples can improve expressiveness at the cost of overall note prediction accuracy.

1 Introduction

The ACME Lab at CU Boulder is developing an augmented reality application for remote "drum circle" jam sessions (AR Drum Circle). Network latency and jitter are major concerns for real-time remote jamming applications. In the AR Drum Circle application, participants use Musical Instrument Digital Interface (MIDI) drum pads to send drum beats over a network. A machine learning model that could predict a sequence or generate a "realistic" sequence of MIDI notes from past MIDI input from jam participants would greatly enhance the user experience over high-latency networks. The model could also be used offline for asynchronous jam sessions, perhaps re-trained to incorporate a particular user's data to capture individual styles, as in Dinculescu et al. [2019].

2 Related Work

Gillick et al. [2019] explored Seq2Seq and recurrent Variational Information Bottleneck (VIB) models for generating expressive drum performances from simple musical ideas. The focus is on humanization and infilling of sequences, which can turn simple MIDI input sequences into complex drum beats. Roberts et al. [2019] developed a recurrent Variational Autoencoder (VAE) to generate a compact latent space representation of drum MIDI sequences. The properties inherent to the latent space (smooth and continuous) allow for realistic interpolations between MIDI sequences. The results are impressive, but the applications are different enough to warrant a different, simpler approach for the AR Drum Circle prediction task.

3 Data

The Magenta Groove MIDI Dataset (GMD), which contains over 13 hours of "tempo-aligned expressive drumming," serves as an ideal data source for the prediction task [Gillick et al., 2019]. The GMD is available for public download, and contains over 1,150 MIDI files of various styles of drumming.

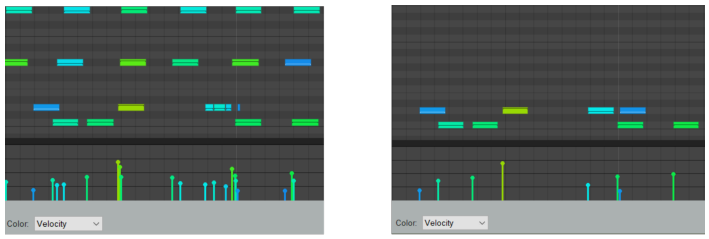
Preprint. Under review.

In the GMD, a MIDI file contains a single track, which in turn contains messages. The messages are a sequence of bytes, that represent drum "notes" or "hits". Each message contains

- a note type (`note_on` or `note_off`),
- a note number (0-127) representing a unique drum sound (e.g. snare, kick, etc.),
- a note velocity (0-127) representing how hard the drum was hit, and
- a delta time between successive messages

3.1 Preprocessing

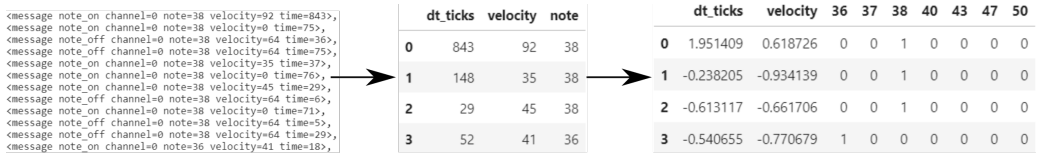
The GMD was produced using a full drum kit (Roland TD-11 electronic drum kit). Due to stylistic differences between hand drumming and drumming with a full kit, all cymbals were removed from the GMD, as were certain drum rudiments (e.g. drag) that sound unnatural in a hand drumming context (see Figure 1). Each drum hit thus has a fixed duration, and only the note initiation (i.e. message type = `note_on`) is of interest.



(A) (B)

Figure 1: (A) Raw example from GMD, (B) Cymbals removed and rudiments adapted to align with hand drumming style.

The `note_on` MIDI messages were converted from bytes into a 2-dimensional array with each row corresponding to a `note_on` message. The columns contain info for delta time (elapsed time since the previous `note_on` message), velocity and note number. The unit for delta time is MIDI "ticks", with 120 ticks per quarter note (480 ticks per beat). "Ghost" notes with velocity of zero were removed. The delta time and velocity were treated as continuous variables, and scaled by removing the mean and scaling to unit variance. The note number is a categorical variable, and was one hot encoded (see Figure 2).



(A) (B) (C)

Figure 2: (A) MIDI messages in byte form, (B) Converted to 2D array,(C) Scaled and one hot encoded

The data set is split into a training, validation, and test set.

4 Methodology

A common method for time series prediction tasks is the sliding window technique, where N preceding data points [x(t-N) … x(t)] are used as input to predict target x(t+1) [Frank et al., 2001]. In the case of the MIDI data set, N=10 preceding notes were used to predict a single note at t+1:

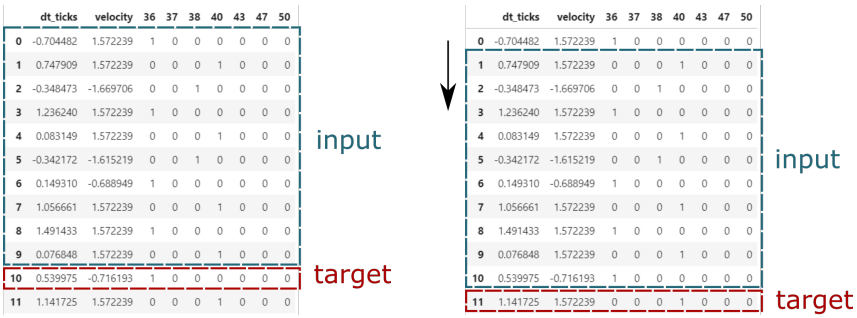


Figure 3: Window slides along data, creating input-target pairs for a standard regression task.

The benefit is that the time series problem is transformed into a standard multivariate regression problem, for which a number of algorithms exist. Four common algorithms were compared, using implementations in scikit-learn and Keras (high level Tensorflow API):

- Linear Regression (LR)
- Random Forest (RF)
- Convolutional Neural Network (CNN)
- Long Short-Term Memory Network (LSTM)

The models were compared against a Baseline that generated drum beats randomly using probability distributions of the data from the training set. It is difficult to quantitatively judge and compare models with artistic, generative tasks [Gillick et al., 2019]. For simplicity, accuracy of the predicted note number was used as a quantitative metric. The accuracy was calculated on the test data set and a few random listening examples were output for a qualitative comparison of the models.

4.1 Linear Regression (LR)

Linear regression is a simple, interpretable model fit with coefficients w = (w1, . . . , wp) to minimize the residual sum of squares between the observed targets in the data set, and the targets predicted by the linear approximation. The sklearn implementation of Ordinary least squares linear regression with default parameters was used (`sklearn.linear_model.LinearRegression`) [Pedregosa et al., 2011].

4.2 Random Forest (RF)

Random Forest is an ensemble method that fits a number of randomly split decision trees on different sub-samples of the training data. It uses averaging of many trees to improve prediction accuracy when compared to just one decision tree. The sklearn implementation of the Random Forest Regressor was used (`sklearn.ensemble.RandomForestRegressor`) [Pedregosa et al., 2011]. The model was manually tuned on the validation set using trial and error. A final model was selected that uses one hundred trees, each with max depth of 20.

4.3 Convolutional Neural Network (CNN)

A CNN was developed using Keras. It used a single 1-dimensional convolution layer connected to a single "dense" layer (i.e. densely connected NN layer). The mean absolute error (MAE) on the validation set was monitored during the training process and, training was stopped if there was no improvement on the validation MAE for proceeding epochs (with 5 epochs of "patience"). The model weights that gave the minimum mean absolute error on the validation set were restored at the end of the training process.

4.4 Long Short-Term Memory Network (LSTM)

As with the CNN, an LSTM was developed with Keras. It used a single LSTM layer connected to a single dense layer. The mean absolute error (MAE) on the validation set was monitored during the

training process and, training was stopped if there was no improvement on the validation MAE for proceeding epochs (with 5 epochs of "patience"). The model weights that give the minimum mean absolute error on the validation set are restored at the end of the training process.

5 Results

Initially, the full training set was used to train all models. The note number accuracy of the different models is compared in Table 1.

Table 1: Note accuracy (out of 1) for each model trained on full data set.

Model (Full Train)	Note Accuracy
Baseline (random)	0.301
LR	0.546
RF	0.527
CNN	0.565
LSTM	0.550

All models outperformed the Baseline in regards to note accuracy. However, the distribution of predicted notes show a bias towards note 38 for all models (see Figure 4A), with the implication that predicted beats lack the variability, and thus expressiveness, of human drumming. Upon inspection of the full training set, the models were being trained on significantly more samples of note 38 and 36, due to the structure of the data windows (i.e. ten inputs to predict a single output).

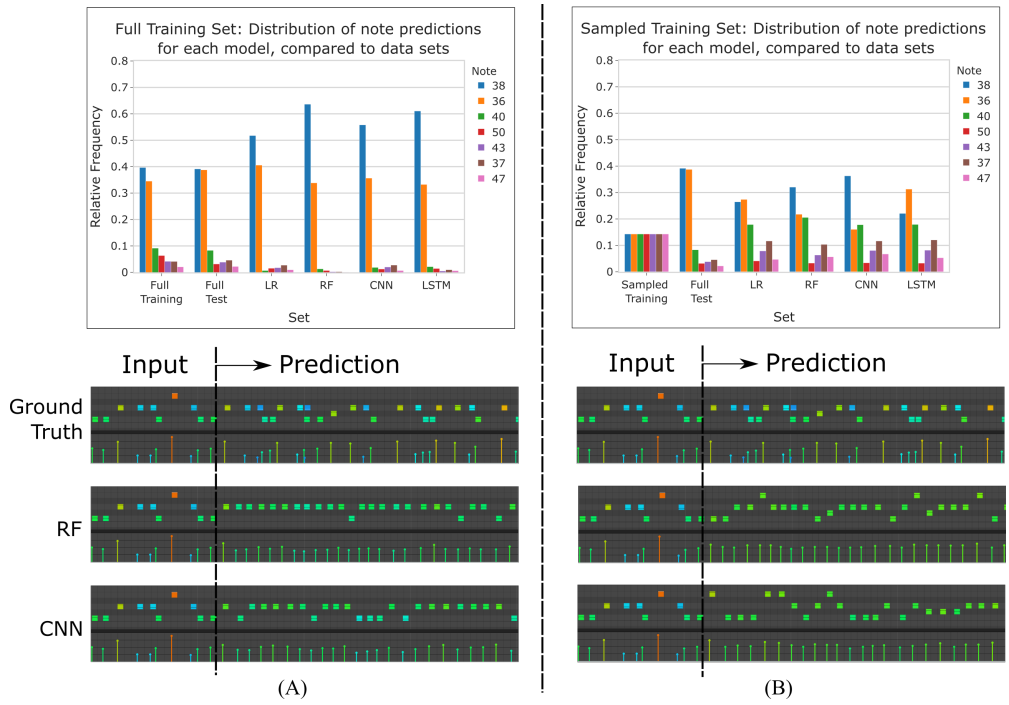


Figure 4: (A) Models trained on full data set, (B) Models trained on sampled data set, with equal distribution of notes (and 6X fewer samples)

Alternatively, the training data can be sampled, so that the models are trained on an equal number of each note type. A follow up analysis on a sampled training set, that required an equal number of samples for each note, resulted in greater note variability (see Figure 4), but lower note accuracy for each model (see Table 2). It should be noted that forcing equal training sample distribution reduced the size of the training set by 6X, which may explain lower note accuracy. This example highlights the need for a better metric than simple note accuracy for this kind of a prediction/generation task.

The models trained on a much smaller sample of data, with lower note accuracy, arguably produced more realistic predictions (see Figure 4B).

Table 2: Note accuracy (out of 1) for each model trained on **sampled** data set.

Model (Full Train)	Note Accuracy
LR	0.424
RF	0.381
CNN	0.403
LSTM	0.422

6 Future Work

Due to time constraints, I did not implement a Seq2Seq model or the recurrent VAE from Roberts et al. [2019], where a bar (or measure) of notes is used to predict the next bar of notes. Attempting to predict a single note from a fixed number of preceding notes ignores the musical structure of bars, a loss of valuable information for the models. Future work will attempt to predict the next sequence based on a series of sequences in the latent space so that predictions have more realistic transitions.This will require developing a recurrent VAE similar to Roberts et al. [2019]. Computational performance will need to be tested to determine if predictions are fast enough to address latency issues in the AR Drum Circle application.

References

Monica Dinculescu, Jesse Engel, and Adam Roberts, editors. *MidiMe: Personalizing a MusicVAE model with user data*. 2019. Publication Title: Workshop on Machine Learning for Creativity and Design, NeurIPS.

R. J. Frank, N. Davey, and S. P. Hunt. Time Series Prediction and Neural Networks. *Journal of Intelligent and Robotic Systems*, 31(1):91–103, May 2001. ISSN 1573-0409. doi: 10.1023/A:1012074215150. URL `https://doi.org/10.1023/A:1012074215150`.

Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning (ICML)*, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL `http://jmlr.org/papers/v12/pedregosa11a.html`.

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. *arXiv:1803.05428 [cs, eess, stat]*, November 2019. URL `http://arxiv.org/abs/1803.05428`. arXiv: 1803.05428.