Multi instrument music generation using VAE

Saravana Rathinam (saravanr@stanford.edu)

SCPD Student, Stanford University

Summary

MIDI format gives a compressed representation of a Music which may give us a tractable way to generate long melodies. In this project we generate MIDI audio using Variation Auto Encoders (VAE) for multiple instruments. We also generate MIDI sequences given a short begining sequence and let the model fill in the rest using a Conditional-VAE.

Motivation

Recent advances in Generative models for Music generation have shown impressive results. Learning the distribution of music creation may be an intractable problem at the moment. However one approach we can take is to build tools that serve as an aid in the creative process. If a musician already has a few ideas in mind on how a song or melody should start, can the problem be modelled as a conditional generative process where given the start can a model generate multiple possibilities of how the song can proceed?

Background Information

A MIDI sequence can be thought to consist of a list of notes and attributes. Each note has:

1. **Pitch**: Frequencey of the note
2. **Velocity**: Intensity of the note
3. **Instrument**: The instrument where the note should be played (or synthesized)
4. **Program**: A control message that specifies which instrument should be selected to play the note
5. **Start time**: The start time of the note (seconds)
6. **End time**: The end time of the note

The data set consisted of 140944 midi files obtained from The dataset for the project a combination of the Lakh MIDI Dataset v0.1 [1] and the MIDI dataset posted at [2]. All the MIDI files were encoded into Note Sequences using the Google Magenta library .

Technical Methods

The following architectures were tried for controllable generation:

- Variation Auto Encoder (VAE) with a single Encoder Decoder
- Beta VAE with $\beta \in [0, 100]$
- VAE with single Encoder and multiple Decoder sharing Z space

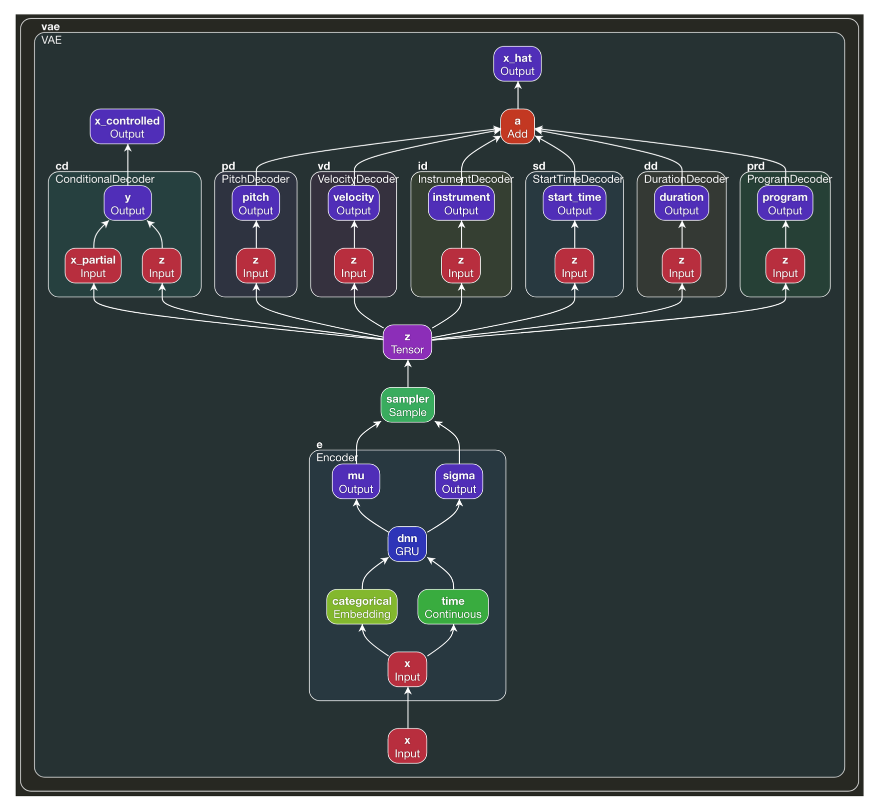

Figure: Conditional VAE Architecture

VAE Architecture

Considerations:

- Variables Pitch, Velocity, Instrument and Program are categorical variables.
- Variables Start time and End time are continuous.
- Hence we need different decoders for Categorical and Continuous variables.
- A separate decoder for controlled music generation.
- One-hot encoding cannot be used for categorical variables - Memory usage.

We convert the categorical variables to embedding space and use this representation in training. To decode back to categorical values, we check closes $L1$ distance of the **logits** to each of the embedding weights. The embeddings are learned during training and allow a much smaller memory foot print.

Loss

The total loss in our Architecture is a sum of KL loss, categorical cross entropy losses, reconstruction losses and controlled reconstruction loss:

$$\begin{aligned}
&\mathbb{E}_{q_\phi(z|x_{time})}[\log(p_\theta(x_{time}|z))] - \mathbb{D}(q_\phi(z|x)||p(z)) \\
&\quad + \mathbb{E}_{q_\phi(z|x_{duration})}[\log(p_\theta(x_{time}|z))] \\
&+ \mathbb{E}_{q_\phi(z|x,x_{partial})}[\log(p_\theta(x|z, x_{partial}))] \\
&- \mathbb{D}(q_\phi(z|x, x_{partial})||p(z)) \\
&+ \mathbb{E}_{q_\phi(z|x,x_{partial})}[\log(p_\theta(x|z, x_{partial}))] \\
&- \mathbb{D}(q_\phi(z|x, x_{partial})||p(z)) \\
&- \sum_{i=1}^{128} x_{pitch} \cdot \log \hat{x}_{pitch} \\
&- \sum_{i=1}^{128} x_{velocity} \cdot \log \hat{x}_{velocity} \\
&- \sum_{i=1}^{128} x_{instruments} \cdot \log \hat{x}_{instruments} \\
&- \sum_{i=1}^{128} x_{program} \cdot \log \hat{x}_{program}
\end{aligned}$$

Generated Sample

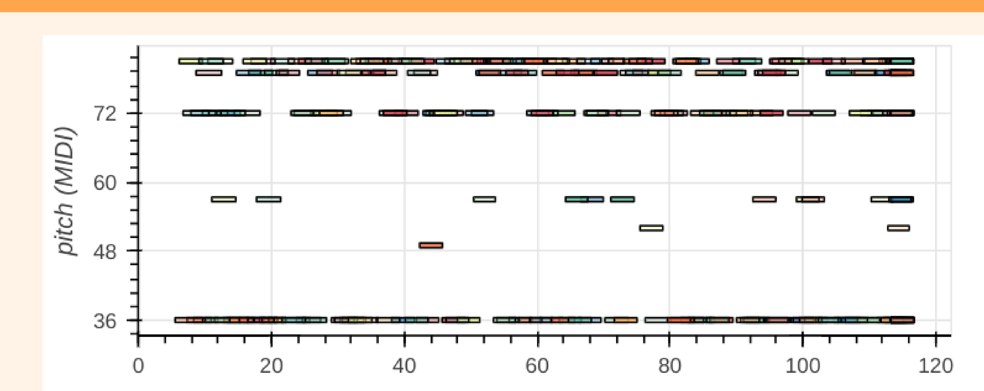

Figure: Pitch and Velocity Plot of Generated Sample

The generated samples follow patterns found in the MIDI files. In this case one can see repeated patterns, a small number of instruments and varying velocity levels in the plot. This sample can be listened at *SoundCloud Link*

Results

The losses in the simple VAE case did not converge. The beta VAE trained models did not produce good quality audio output. The model learned to produce related musical instruments together and have repeated variations in background audio. The pitch and velocity of many samples did not feel like noise. However the results seem to be affected by class imbalance of the input, which has a lot of Piano music and less of other types of instruments.

(a) Test Loss (b) Train Loss

Figure: Test and Train Loss

Acknowledgements

My TA mentor, **Kelly He** is awesome! Thank you Kelly. This project would not be possible without your prompt guidance and support. And thank you **Jiaming Song**, **Chenlin Meng** and **Kristy Choi** for the very thoughtful discussions, ideation and support on building this project.

References

[1] Colin Raffel.
Lakhs midi database.

[2] midi man.
The largest midi collection on the internet.

[3] Gautam Mittal, Jesse H. Engel, Curtis H., and Ian S.
Symbolic music generation with diffusion models.
CoRR, abs/2103.16091, 2021.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel.
Denoising diffusion probabilistic models, 2020.

[5] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon.
Ilvr: Conditioning method for denoising diffusion probabilistic models, 2021.

[6] Jiaming Song, Chenlin Meng, and Stefano Ermon.
Denoising diffusion implicit models, 2020.

[7] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon.
Sdedit: Image synthesis and editing with stochastic differential equations, 2021.