Final Project Report

Team members: Jung Eun Hong, Jimin Heo

Problem Description

In this project, we explore the collaborations of machine learning techniques with music processing to generate songs into different genres. We propose a methodology that leverages Variational Autoencoders (VAE) and Gated Recurrent Units (GRU) to capture the essence of music compositions, including pitch and instruments. With this approach, we aim to generate music with different styles while maintaining the essence of the original compositions.

Our project not only demonstrates the capabilities of AI in music transformation but also offers insights into the intersection of creativity, technology, and musical expression. Moreover, since this method analyzes key features of music like notes, and pitches, it enables listeners to generate their own music with different styles using these features. Additionally, we anticipate that this technology can inspire new genres, and provide tools for people to experiment with unique sounds and arrangements.

Related Work and References

We focused on the methods we will use; VAE and GRU. There are some related works that transform music styles or generate new music.

To establish a foundational understanding of VAEs, we refer to "An Introduction to Variational Autoencoders" by Kingma and Welling. This work offers a comprehensive overview of VAEs and their applications in generative modeling. Building upon this knowledge, we investigate music style transfer, guided by "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer" by Roberts et al. This study presents an approach for modeling latent space including compact encoding of the music style.

1. An introduction to Variational Autoencoders
   Paper: https://arxiv.org/pdf/1906.02691

2. MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer
   Github: https://github.com/brunnergino/MIDI-VAE
   Paper: https://arxiv.org/abs/1809.07600

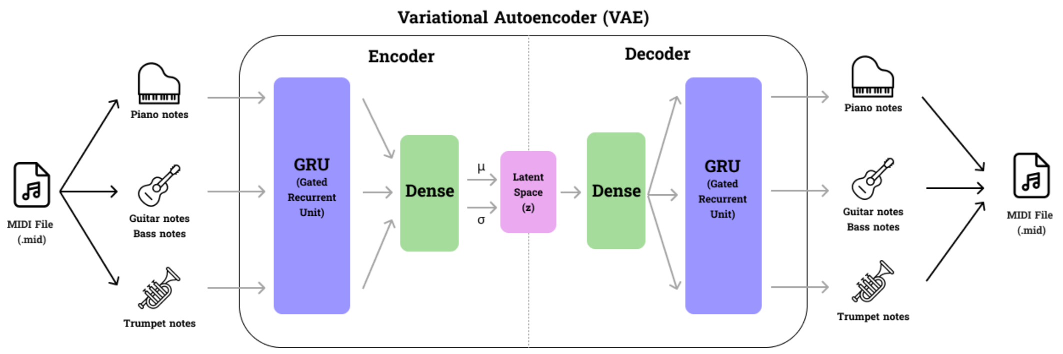
3. Self-Supervised VQ-VAE for One-Shot Music Style Transfer
   Github: https://github.com/cifkao/ss-vq-vae
   Paper: https://arxiv.org/abs/2102.05749

Model Description and Illustration

The model we used in the project utilizes a Variational Autoencoder (VAE) combined with Gated Recurrent Unit (GRU) layers to capture musical sequences from various instruments. The architecture features an encoder that reduces the input sequences to a latent representation and a decoder that reconstructs the sequences from this latent space. Given that our input data—music—is inherently sequential, we initially considered using LSTM due to its proficiency in handling sequential data. However, we opted for GRU due to its simpler architecture and quicker training times, which proved more efficient and suitable for our project's requirements.

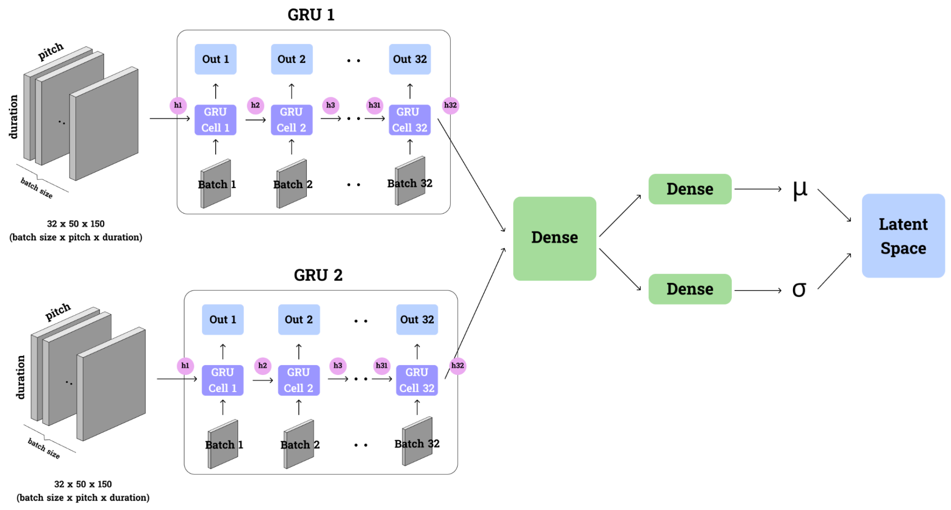
- Overall structure of VAE (Variational Autoencoder)

In the figure below, the encoder processes the input sequences from each instrument(but for now we only used right and left-hand piano notes each) through GRU layers. The outputs are concatenated and passed through a dense layer to obtain the latent representation. The decoder then reconstructs the sequences using GRU cells for each instrument, starting from the transformed latent vector.

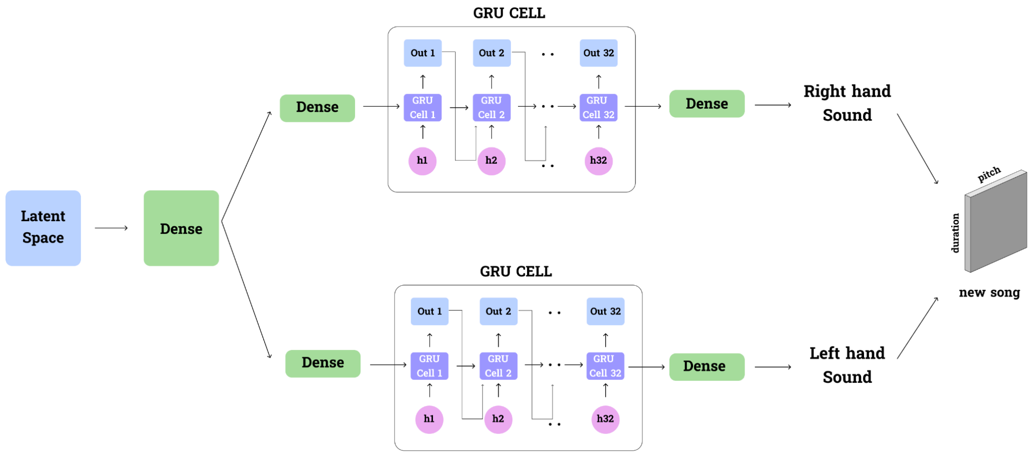


- Structure of GRU (Gated Recurrent Unit)

1. GRU in Encoder



2. **GRU in Decoder**



$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})) \\
h_t &= (1 - z_t) \odot n_t + z_t \odot h_{(t-1)}
\end{aligned}
$$

[3] pytorch

h_t is the hidden state at time t, x_t is the input at the time t, and n_t, z_t, r_t are the new gates, update, and reset each.

Learning/Inference Description and Validation

The inference algorithm of this model involves using the trained VAE model to generate new musical sequences or to reconstruct existing sequences.

1. **Preprocess input data**

   Our workflow begins with preprocessing [5]MIDI file inputs to align with the requirements of the GRU-VAE model. Musical sequences are standardized into binary data, focusing on the pitch range of 34 to 84 and separating them into right-hand and left-hand piano parts. When only one piano part is available, it is duplicated for both. These sequences are reshaped into (batch size, sequence size, input size) and normalized for consistency across the dataset. This preprocessing ensures that the model receives structured input conducive to learning and generating diverse musical compositions.

2. **Encode**

   The encoder component transforms our preprocessed music sequences into a latent space representation. To achieve this, we employ dedicated GRU layers for processing both left-hand and right-hand piano sequences. Each GRU layer returns an output sequence along with its hidden state. We utilize the last hidden state from each GRU layer for left-hand and right-hand sequences, concatenate them, and pass the concatenated output through a dense layer. This dense layer effectively reduces dimensionality and enhances feature extraction using the ReLU activation function. The output sequence from each GRU layers serves as the input data for the decoder component.

3. **Compute latent space (Used as the Style Changer)**

   Following the dense layer, the output is further processed to derive the mean (mu) and log-variance (logvar) parameters of the latent space distribution. Reparameterization ensures that the latent space adheres to a normal distribution, facilitating effective sampling and learning within the VAE framework.

   When transitioning classical music to Jazz music during reconstruction, we utilize the latent space originally created for Jazz music. Within this latent space, we incorporate additional information from the style classifier. The style classifier condenses data from the latent space, enabling the VAE to integrate stylistic attributes essential for generating Jazz music. This process enhances the model's capability to capture and reproduce the distinctive musical characteristics inherent to Jazz.

4. **Decode**

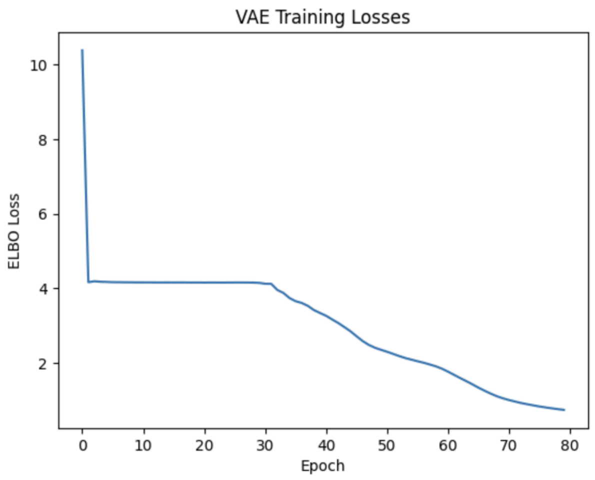
   For music sequence reconstruction, the decoder component begins by transforming the latent variable z through a linear layer followed by ReLu activation. This processed data initializes the GRU cell, which then iteratively generates the output sequence. The input data for the GRU cell is the output of the GRU from the encoder. Each step in this process updates the hidden states based on the previous state and the current latent representation. Finally, the output sequence undergoes a sigmoid function to ensure values are constrained between 0 and 1, aligning with the binary nature of the data.

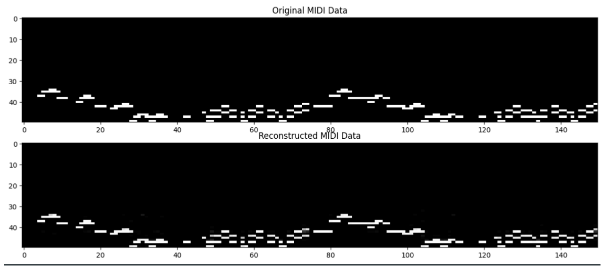
5. **Loss Function**

   Our dataset is characterized by binary data with a significant number of zeros. To address the imbalance between classes and measure reconstruction accuracy, we employ the binary cross entropy loss function with pos_weight in our Variational Autoencoder (VAE). This function evaluates the fidelity of the VAE's output compared to the original input data.

   Additionally, the Kullback-Leibler (KL) divergence term is utilized to regularize the VAE's latent space. It penalizes deviations from a standard Gaussian distribution defined by mu and logvar, thereby encouraging the model to learn meaningful representations of the data. By minimizing this combined loss during training, the VAE not only achieves effective representation learning but also ensures accurate reconstruction of the input data.

6. **Validation**
   According to the loss function our model is converging effectively. Additionally, visual inspection of the original music compared to its reconstructed music shows similarity in their tendencies. This observation indicates that our current model successfully captures the underlying patterns and features present in the music sequence. However, it was hard to validate the change of the style with this plot.



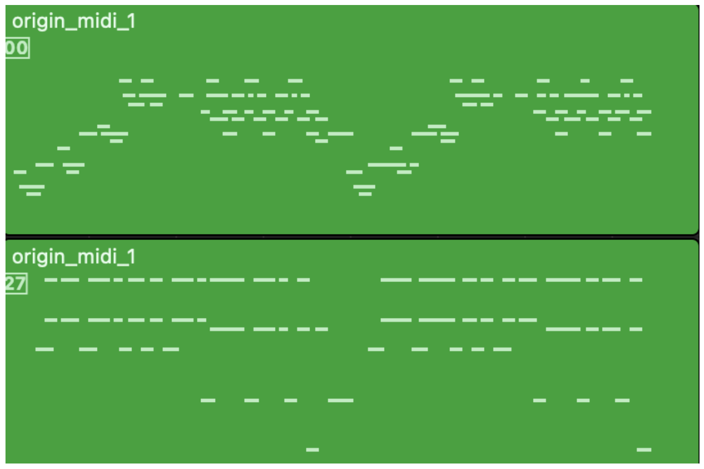


The upper plot is original music
The below plot is regenerated music

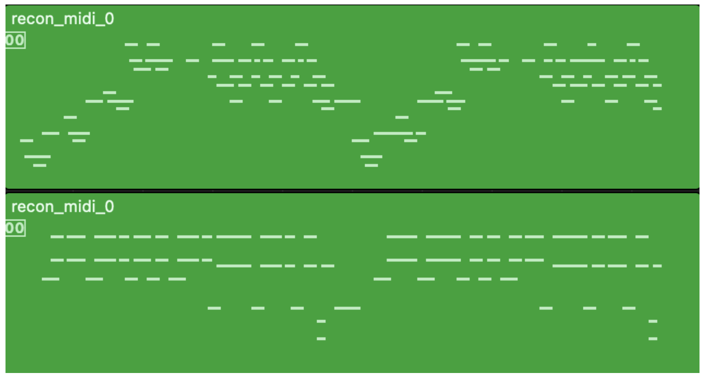
Visualization and Analysis of Results

- **Visualization of reconstructed songs**

<u>Note of original music(right-hand, left-had)</u>



<u>Note of reconstructed music(right-hand, left-had)</u>



- **Analysis of Result**

  As shown in the visualization, we succeeded in generating regenerated songs based on original songs. It has the same musical trend as the original songs and the generated music is high-quality. We attempted to change the music's style, but evaluating the style transfer is challenging due to the difficulty in defining the style of the generated song. Through various changes in the model, we realized that we could have made a better style changer with these improvements.

  1. Preprocess input data

     We extracted only the pitches from the piano roll as input data, but there are other features, such as volume and note duration, that can also be used as inputs. Incorporating these features could make the generated music sound more like a real song.

  2. Optimize hyperparameters

     There are a lot of hyperparameters like latent dimensions, learning rate, and batch size. We could optimize these parameters and make the song sound better.

Implementation Details

Our project is implemented in Python, utilizing PyTorch for building and training the Variational Autoencoder (VAE) model. The model leverages GRU layers to handle sequential musical data from multiple instruments. We followed the overall structure from the paper "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer". While the original code from the paper is implemented with the TensorFlow library, our model is implemented using the PyTorch library.

In summary, in our VAE model, there are 4 main functionalities.

- Encoder: Compress musical sequence data to into a latent space
- Decoder: Reconstruct the musical sequences from the latent space
- Train: Use loss function(binary cross entropy) and optimizer(Adam) to train the model
- Inference: Generate a new song by decoding vectors

Here is the code of our project.

Github: https://github.com/hjimjim/Jukebox_VAE

Individual Contribution

- Jungeun Hong: Preprocess input data, Design&Implement VAE model, Generate music files
- Jimin Heo: Design&Implement VAE model, Optimize the model, Loss Function, Generate music files

References

[1] Cramer, J., Dhariwal, P., Hesse, C., Kreso, A., Lample, G., & Talbot, D. (2019). *Musikalo: Few-shot Music Generation with Long Short-Term Memory Networks*. arXiv preprint arXiv:1906.02691.

[2] Brunner, G., Konrad, A., Wang, Y., & Wattenhofer, R. (2018). *MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer*. 747–754. https://doi.org/10.3929/ethz-b-000292318

[3] Lim, Y.-Q., Chan, C. S., & Loo, F. Y. (2021). ClaviNet: Generate Music With Different Musical Styles. *IEEE MultiMedia*, *28*(1), 83–93. https://doi.org/10.1109/mmul.2020.3046491

[3] GRU — PyTorch 2.1 documentation. (n.d.). Pytorch.org. https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

[4] GRUCell — PyTorch 2.0 documentation. (n.d.). Pytorch.org. https://pytorch.org/docs/stable/generated/torch.nn.GRUCell.html

[5] Classical Music MIDI. (n.d.). www.kaggle.com.

https://www.kaggle.com/datasets/soumikrakshit/classical-music-midi