74 Autocomplete for Music

Shou Miyamoto[1] and Rachel Nataatmadja[1]
Supervisors: Nathan Allen[1] and Partha Roop[1]
[1] Department of Electrical, Computer, and Software Engineering, The University of Auckland

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
ENGINEERING
DEPARTMENT OF ELECTRICAL, COMPUTER, AND SOFTWARE ENGINEERING

Objectives

To design an auto-complete music generation model that can take in an MIDI input and complete the rest of the phrasing with the attributes:

- Lightweight model that can be used on edge devices.
- Music from the classical genre.
- Lower complexity and relatively fast generation time.
- Performs to a similar degree as state-of-the-art music generation models.

Background

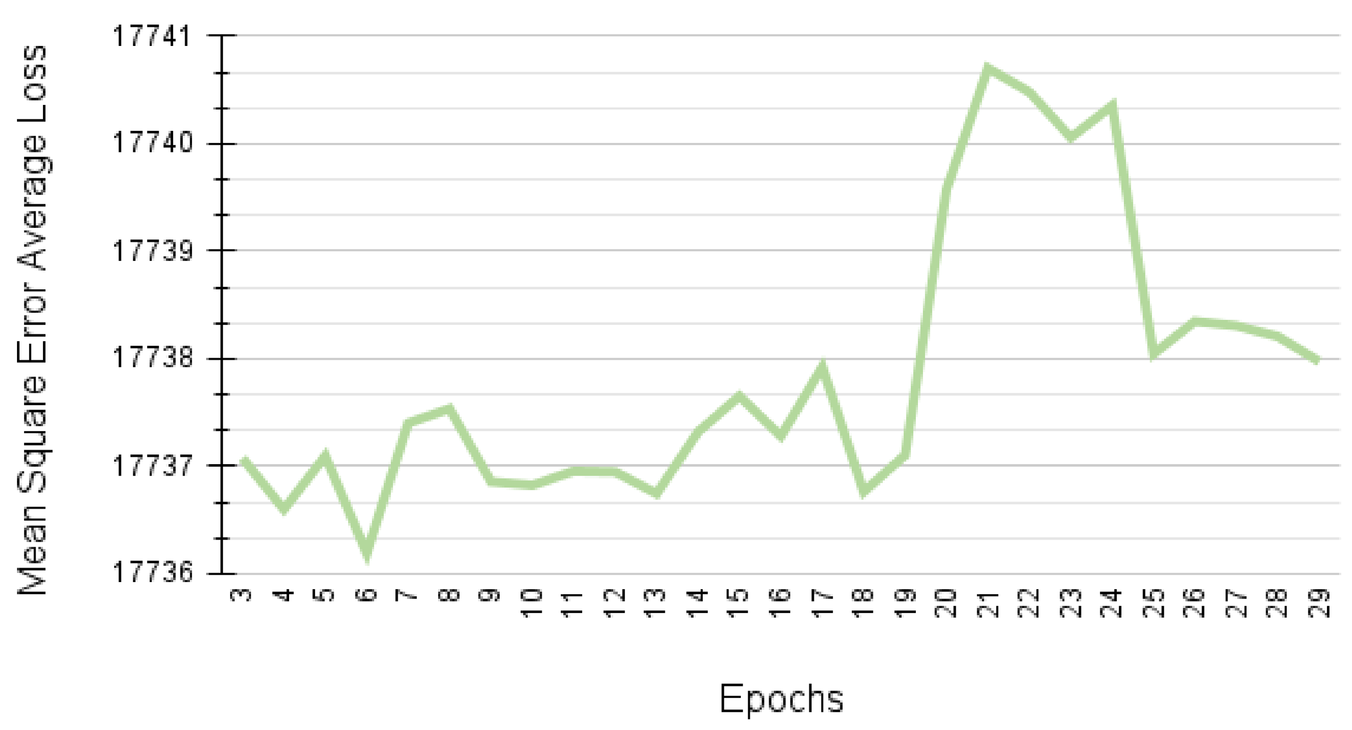
Music generation using AI can potentially expose more people to music composition. However, some limitations can prevent people from accessing this tool, such as:

- Current trends favor high-complexity models.
- These models demand powerful hardware to run.
- Not everyone has the computational resources to run those models.

Our approach used the Musical Instrument Digital Interface (MIDI) files, which is:

- A file format used to train and infer musical data in models.
- Consists of data on what notes to play, when, and how they should sound.

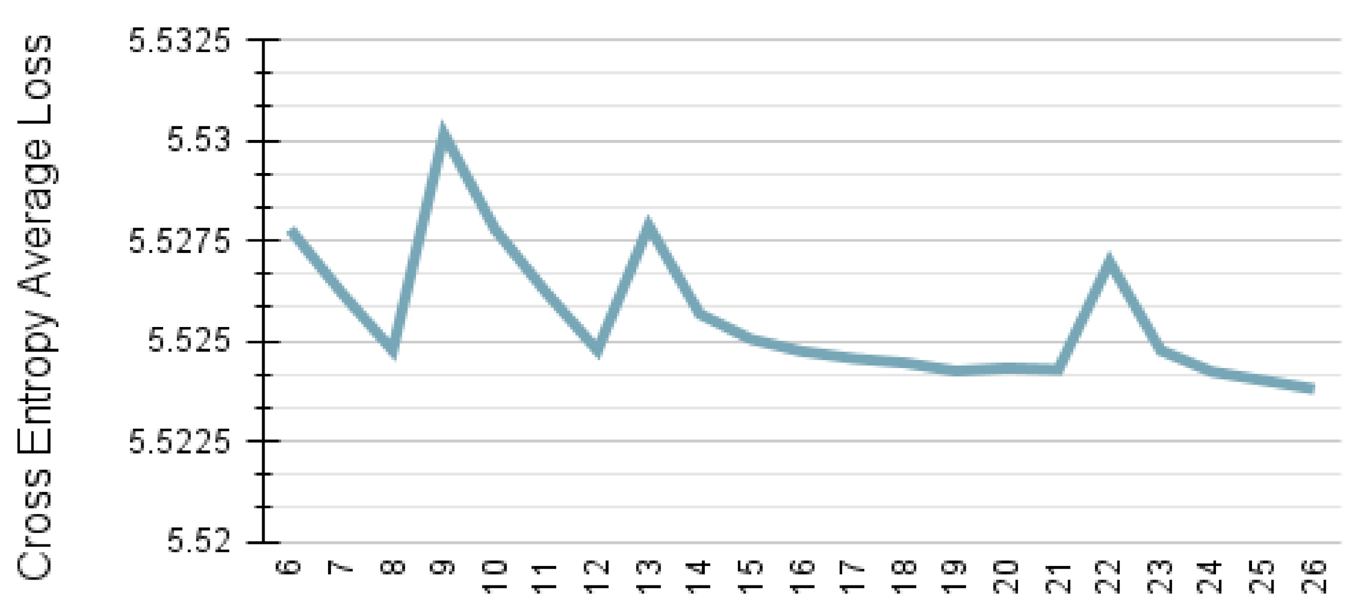
v1.1.1 Training Loss Progression Over Epochs

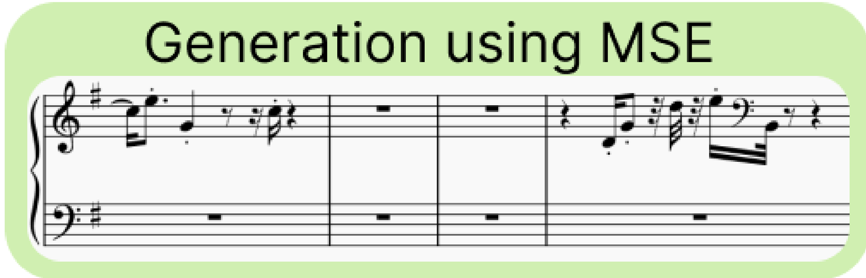


Lightweight Transformer Model

- Used MAESTRO version 3 dataset consisting of MIDI files of Classical Piano.
- Tokenised MIDI files using MidiTok [1] for enhanced model training.
- Created a lightweight encoder-only transformer to ensure low complexity and fast inference time.
- Fine-tuned hyper-parameters for the final model version (3.1.2).
- Experimented with different loss functions, Mean Square Error and Cross Entropy to minimise silence gaps in the music.

v3.1.2 Training Loss Progression Over Epochs





Less music notes.
More silences.

More music notes.
Less silences.

Evaluation Metrics

Metrics are a critical section of our research project in order to compare how well our model performs in generating auto completions against other generative music models. In this section we compare against models such as:

- Music Transformer (Google)
- Sentiment Transformer-GAN (University of Campinas)
- Theme transformer (Researchers)
- Musenet (Open AI, SparseTransformer)
- MusicGen (Meta)
- Coconet (Google , CNN)

The two metric systems we used are Frechet Audio Distance (FAD) and Musepyn:

- FAD measures the distortion in audio. We use the lightweight library with PANN mode, where a lower FAD score depicts a high-quality audio generation.
- MusePyn, is a library for symbloic music generation and evaluation metrics. Pitch range, pitch class and Polyphony are the metrics measured against in [2].

Model Name	FAD Score $\times 10^{-4}$	Pitch Range	Pitch-class	Polyphony	Parameters
Project 74's Transformer	26.9	43	10	1.75	19m
Coconet	3.23	31	8	3.6875	3.7m
Musenet	2.17	17	8	1.07	110m
MusicGen	21.2	–	–	–	300m-3.3B
Music transformer	5.28	36	11	4.05	4m-44m
Theme transformer	20.4	48	10	1.75	3m
Sentiment Transformer-GAN	3.86	64	12	3.13	40m

Conclusions

Overall, while comparing the data generated between our model and the others, our model is almost comparable to existing models, using significantly fewer resources and parameters. Future avenues to explore are the optimisation of layers and further training.

References

[1] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah Seghrouchni, and N. Gutowski, "MidiTok: A python package for MIDI file tokenization," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[2] P. Neves, J. Fornari, and J. Florindo, "Generating music with sentiment using transformer-gans," 2022.