Self-supervised melody generation using Large Language Model

Phan Anh Khoi, Tran Phuong Anh, Pham Xuan Loc, Vo Thi Yen Nhi, Hoang Thi Tu Anh

Introduction

Music, a universal form of art, is not merely a combination of sounds but a powerful medium for conveying emotions and connecting people across cultural and linguistic boundaries.

It is interesting to note that we can identify concepts in the musical structure that are similar to concepts in the linguistic structure, such as "phrases", "stresses", "punctuations", and "sentences" [8]. As presented by Adorno and Gillespie [1], music also resembles a language in that both are a temporal succession of articulated sounds that are more than just a single sound and follow a certain coherence. Furthermore, from a neuroscience perspective, there is a growing body of evidence linking the way we perceive music and the way we perceive natural language [10].

Building on this similarity, along with the remarkable advancements in Natural Language Processing (NLP) and Deep Learning, new promising directions have emerged in the field of automated music generation. For instance, the Transformer-based neural architectures of BERT [3], GPT-2 [13], and Transformer XL [2] use encoders-decoders and various attention mechanisms to achieve great performance at language learning and generation. For example, the study by MuseNet [11] utilized a Transformer model to generate complex MIDI music compositions spanning multiple instruments and genres, ranging from classical to jazz, by learning sequential relationships between instruments and chords. Similarly, LakhNES [5] leveraged pre-trained Transformer models to produce chip-tune music from the Lakh MIDI dataset, demonstrating the ability to learn and recreate specific musical styles. More recently, Jukebox [4] advanced the field further by generating raw audio, including vocals and lyrics, using large language models to produce highly realistic and stylistically diverse music.

This study builds upon these advancements with a novel approach, focusing on implementing the GPT architecture from scratch while constructing a dedicated vocabulary and tokenizer tailored for the task of melody generation. Unlike previous approaches, this research specifically aims to generate melodies using data from the Essen Folksong Database, a collection of approximately 8,400 folk songs from various European countries. The dataset was converted from MIDI files into text format (.txt) using the MidiTok library [7].

However, due to the relatively small size of the dataset, mapping between pitch and duration is not feasible because the frequency of each pitch-duration pair is extremely low. To address this limitation, the study focuses solely on pitch values, disregarding duration and rhythm. This simplification provides a clearer direction for the melody generation task rather than attempting to tackle generalized music generation.

After encoding, the data is structured into a vocabulary consisting of unique pitch values, which are then mapped into numerical tokens through a specifically designed tokenizer. This process ensures that the GPT model's input accurately reflects the sequential structure of melodies. Furthermore, implementing the model from scratch allows for flexible customization of key parameters such as vocabulary size, embedding dimensions, and output structure, optimizing the model's performance for this specific task.

This approach not only enables the generation of coherent and creative melodies but also opens up possibilities for developing automatic music generation tools with high customization potential, especially suitable for small datasets like the Essen Folksong Database.

Previous works

Music generation has long been a field of interdisciplinary exploration, blending music theory, computational methods, and artificial intelligence. The evolution of this domain has seen significant shifts—from rule-based systems to modern deep learning approaches, driven by advancements in machine learning, particularly in natural language processing (NLP). In recent years, the adaptation of large language models like GPT-2 for music generation has further expanded the creative potential of AI systems. This section discusses the historical progression of computational music generation, the role of NLP-inspired models in advancing the field, and the specific application of GPT-2 in generating symbolic music.

The journey of music generation began with algorithmic and rule-based approaches in the mid-20th century. These early methods relied on predefined rules and probabilistic models, such as Hidden Markov Models (HMMs) [12], to generate symbolic musical sequences. HMMs were adept at capturing short-term dependencies through state-based transitions, enabling the creation of simple melodies. However, they struggled with long-term coherence, as they lacked the ability to model hierarchical relationships and broader structures in music.

The development of Recurrent Neural Networks (RNNs) marked a major milestone in the field. RNNs were inherently suited for sequence modeling due to their ability to process temporal data. Hochreiter and Schmidhuber's introduction of Long Short-Term Memory [9] addressed key limitations of traditional RNNs, such as the vanishing gradient problem. By effectively learning long-range dependencies, LSTMs enabled more sophisticated modeling of musical compositions. Skuki [16] demonstrated the utility of LSTMs in generating single-instrument compositions, highlighting their ability to capture melody and rhythm. However, despite their advantages, LSTMs were often limited in maintaining global structure over extended sequences, which is a critical aspect of music composition.

Generative Adversarial Networks (GANs) brought further innovation to music generation by introducing adversarial training paradigms. These models enabled conditional generation and the creation of complex multi-instrument arrangements. For example, MidiNet [17] focused on chord-conditioned music generation, while MuseGAN [6] extended this framework to model multi-instrument interactions, producing richer and more cohesive musical outputs. Despite their advancements, GANs required significant domain-specific adaptation to ensure stylistic consistency. The introduction of Transformer architectures represented a transformative moment in music generation. The Transformer's self-attention mechanism allowed for more effective modeling of long-range dependencies, overcoming many of the limitations faced by RNN-based models. MusicVAE [15], a hierarchical variational autoencoder, demonstrated the utility of latent space representations for symbolic music, enabling style interpolation and rhythmic manipulation. Subsequently, models like MuseNet [4] and LakhNES [13] further leveraged Transformer-based architectures to generate multi-instrument compositions and genre-specific music. OpenAI Jukebox [3] expanded the scope of AI-driven music generation by shifting from symbolic representations to raw audio synthesis, achieving high fidelity through hierarchical VQ-VAE models.

Building on the successes of Transformer architectures, GPT-2 has been adapted for symbolic music generation due to its autoregressive nature and capacity for modeling long-range sequential data. GPT-2's design, which focuses on predicting the next token in a sequence, aligns closely with the requirements of music generation, where temporal dependencies play a crucial role. Researchers have fine-tuned GPT-2 on symbolic datasets, including ABC notation and MIDI representations, to generate diverse musical compositions. Payne [4] demonstrated the potential of GPT-2 in generating stylistically coherent music using ABC notation, a text-based format for encoding melodies. Similarly, Donahue et al. [13] employed GPT-2 to produce MIDI sequences, showcasing its adaptability across genres and its ability to maintain structural coherence in complex compositions.

The evaluation of GPT-2 for music generation combines both qualitative and quantitative approaches. Human listeners play a central role in assessing the musicality, coherence, and stylistic fidelity of generated pieces. Qualitative evaluations often reveal the model's strengths in generating contextually appropriate and aesthetically pleasing compositions. For instance, Payne [4] conducted qualitative studies where participants assessed GPT-2-generated symbolic compositions encoded in ABC notation, highlighting the model's ability

to produce stylistically consistent and contextually appropriate pieces. Similarly, Donahue et al. [13] used human listeners to evaluate GPT-2's outputs on MIDI datasets, noting high fluency in melody generation and convincing harmonic structures. These evaluations consistently demonstrate that GPT-2 excels in generating contextually relevant compositions that align well with the styles present in the training data. Quantitative metrics provide objective measures of GPT-2's performance in music generation. Perplexity, a key metric in sequence modeling, measures how well the model predicts the next token in a sequence, with lower perplexity indicating better predictive accuracy. Payne [4] observed that fine-tuning GPT-2 on symbolic music datasets resulted in significant reductions in perplexity, indicating improved modeling of musical sequences. BLEU and ROUGE scores also complement qualitative evaluations by quantifying the model's ability to replicate characteristic patterns, motifs, and rhythmic structures present in traditional music. These metrics highlight GPT-2's ability to generate musically coherent and stylistically accurate pieces.

Comparative studies have shown GPT-2's superior performance over earlier models such as LSTMs and other Transformer-based architectures. Payne [4] reported that GPT-2 outperformed LSTMs in maintaining long-range dependencies, which is essential for generating cohesive musical compositions. Additionally, Dhariwal et al. [3] found GPT-2 excelled at capturing nuanced patterns and generating compositions with greater structural progression, a key characteristic of compelling music.

Statistical analyses further validate GPT-2's effectiveness by examining pitch distributions, rhythmic patterns, and harmonic structures in its generated compositions. Payne [4] demonstrated that GPT-2-generated pieces exhibited pitch distributions closely resembling those of human-composed works, while Donahue et al. [13] confirmed the model's ability to effectively learn and replicate rhythmic structures from its training datasets. Roberts et al. [15] also noted that GPT-2's outputs often captured harmonic relationships consistent with the style and genre of the training data, reinforcing its capacity for stylistic adaptation.

Despite its success, GPT-2 faces limitations in generating multi-instrument compositions and real-time music synthesis. Future research could address these challenges by integrating symbolic and audio representations, improving polyphonic modeling, and combining GPT-2 with other frameworks, such as GANs or hierarchical VAEs, to enhance its creative capabilities.

In summary, GPT-2 has established itself as a powerful tool for symbolic music generation, leveraging its NLP-inspired architecture to produce compositions that are both coherent and stylistically rich. Its application has advanced the field of computational creativity, laying the groundwork for future innovations in AI-driven music composition.

Methodology

Dataset Preparation and Tokenization

Converting a collection of MIDI files into a text-based representation formed the basis of our dataset. This process involved extracting individual note pitches and durations, which were subsequently encoded as a sequence of distinct tokens. Thankfully, the Essen Dataset on Kaggle has already extracted pitches from MIDI files to 8472 text files using the OctupleMono tokenizer featured by MidiTok. We created a custom tokenizer, MidiTokenizerV1, to map these musical elements into numerical values, enabling the model to effectively process the sequential folk music data.

Create a vocabulary for pitches

Creating a new vocabulary specific to our folk music dataset was a key step in adapting the GPT-2 model. The vocabulary stored in the vocab variable, serves as a dictionary that maps each unique token to a corresponding integer. This allows the model to interpret and generate sequences of folk music elements. We also create 2 special tokens, UNK and EOS, to handle unknown tokens from the input and the end of each sequence, respectively.

Model adaptation

The GPT-2 Medium model served as our foundation, which we then adapted for folk music generation. We implemented GPT-2 model from scratch as instructed in the book "Build a Large Language Model from Scratch" by Sebastian Raschka [14]. GPT-2 Medium is the 355M parameter version of GPT-2, which has 16-head attention, 24 transformer blocks, the embedding dimension of 1024 and 355 million parameters. To ensure compatibility with our custom vocabulary, we replaced the original embedding layers and output layers of the model. Specifically, the token embedding layer (tok_emb) and the output head (out_head) were replaced with new layers that match the size of our custom vocabulary. This adaptation enables the model to effectively learn and generate folk music sequences using our vocabulary.

Fine tuning

It's not strictly necessary to fine-tuning the model with pretrained weights. Pretraining an LLM on a large text corpus ourselves is time and resource-intensive, by experience, we found that using openly available weights as an alternative is less time and resource-consuming and less overfitting, although the pretrained weights was used for a completely different vocabulary and tokenizer. We employed a cross-entropy loss function, a common choice for classification tasks, to measure the difference between the predicted and target token distributions. We used the AdamW optimizer, an extension of the Adam optimizer that incorporates weight decay, with a learning rate of 5e-5 and weight decay of 0.1 to update the model's parameters and minimize the loss function.

Generation

The fine-tuned model was then used to generate new folk music sequences. We employed techniques like top-k sampling and temperature scaling to control the output, to increase creativity and variety of melody generated compare to greedy algorithms.

Temperature scaling adjusts the probability distribution by dividing the logits (raw output scores of the model before passing them to the softmax function) by a temperature parameter T. Higher temperatures (T>1) increase diversity, while lower temperatures (T<1) favor more predictable outputs.

Top-k sampling, when combined with probabilistic sampling and temperature scaling, can improve the text generation results. In top-k sampling, we can restrict the sampled tokens to the top-k most likely tokens and exclude all other tokens from the selection process by masking their probability scores. The top-k approach replaces all nonselected logits with negative infinity value (-inf), such that when computing the softmax values, the probability scores of the non-top-k tokens are 0, and the remaining probabilities sum up to 1.

Evaluation: Perplexity

We also monitored perplexity during training, which measures how well the model predicts a sample:

$$PPL = exp(L)$$

where L is the average cross-entropy loss. Lower perplexity suggests better generalization. Perplexity is often considered more interpretable than the raw loss value because it signifies the effective vocabulary size about which the model is uncertain at each step. The generated sequences were converted back to MIDI format for playback.

Experiment and results

Quantitative Evaluation

After 5 epochs, the model alternated between an average cross-entropy train loss of 1.75 and validation loss of 1.80 over several hours, meaning the model had a hard time optimizing further from this point on. The

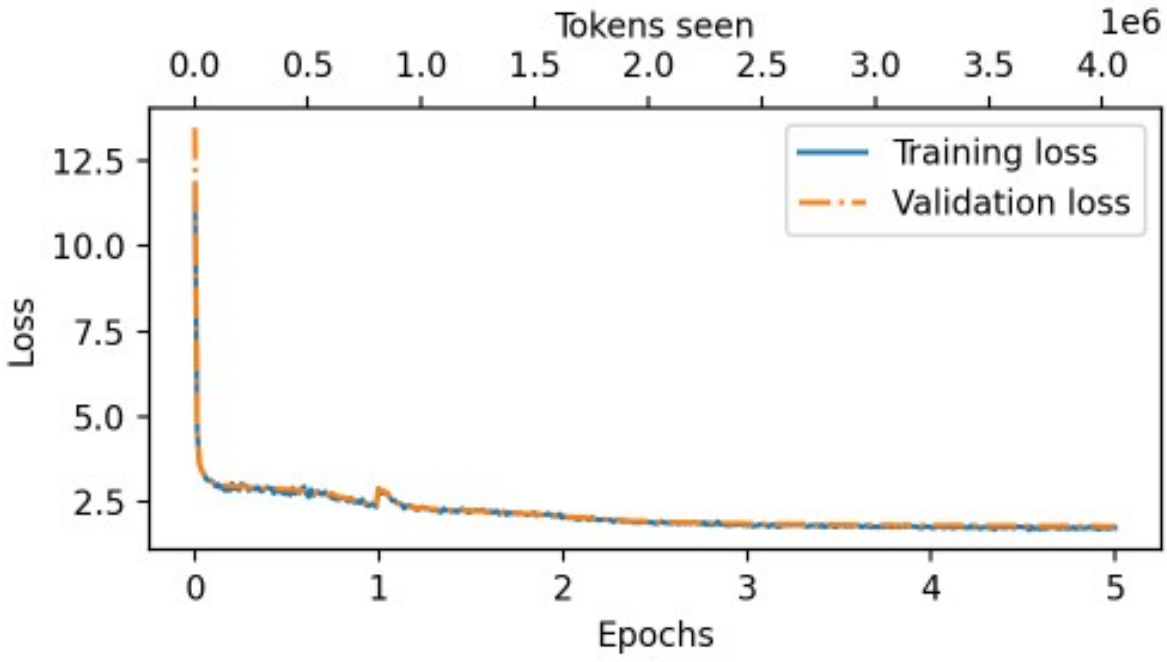

Figure 1: The model's training and validation loss over the five training epochs. Both the training loss, represented by the solid line, and the validation loss, represented by the dashed line, sharply decline in the first epoch and gradually stabilize toward the fifth epoch. This pattern indicates good learning progress and suggests that the model learned from the training data while generalizing well to the unseen validation data.

average perplexity calculated on validation set is approximately 6, which would translate to "the model being unsure about 6 among 54 tokens in the vocabulary to generate as the next token".

Qualitative Evaluation

Experimental Setup

The input sequences, comprising a series of musical notes represented in scientific pitch notation (e.g., C4, G4), were provided to the model. The model, trained on a comprehensive dataset of musical pieces, then generated a continuation of the sequence, producing a series of MIDI note numbers corresponding to the predicted pitches.

We used ChatGPT-4o for scoring and give qualitative feedback about generated melody. The prompt is like this: "Given the input input_pitches, score the model response output_pitches on a scale from 0 to 100, where 100 is the best score." Qualitative feedback was also provided to justify the scores and highlight specific strengths and weaknesses.

Results and analysis

The results demonstrate that our model is capable of generating melodic sequences that exhibit a degree of musicality. The generated melodies generally demonstrate coherent melodic development, utilizing motifs and repetition effectively. However, some instances of excessive repetition and predictable harmonic progressions were observed, suggesting areas for further refinement.

The highest-scoring melodies (e.g., Input: F4, D4, G4, B4, C5, F4, G4) exhibit a balance of structure and creativity, showcasing the model's potential for generating engaging and musically interesting sequences. Future work will focus on addressing the limitations identified in this experiment, particularly by exploring

5. Input: E4, F#4, G#4, A4, B4, F#4

[64, 66, 68, 69, 71, 66]

Output:

```
[64, 66, 68, 69, 71, 66, 66, 64, 69, 71, 64, 64, 66, 69, 66, 69, 69, 71,
73, 71, 73, 71, 71, 76, 73, 76, 76, 71, 73, 74, 71, 71, 69, 68, 71, 73,
73, 69, 66, 69, 66, 71, 69, 69, 69, 64, 61, 64, 61, 64, 66, 68, 66, 68,
71, 68, 69, 71, 73, 71, 73, 71, 71, 71, 69, 71, 71, 73, 76, 76, 69, 64,
64, 64, 64, 64, 66, 69, 71, 73, 76, 71, 69, 64, 66, 64, 61, 64, 64, 66,
69, 71, 69, 64, 69, 64, 66, 69, 66, 71, 69, 71, 71, 69, 71, 73]
```

Score: **85/100**
Reason:

- Rich tonal palette with strong ascending/descending phrases.
- Repetitive sections (64) weaken the overall composition but still retain thematic strength.

Figure 2: Input, generated melody, and a feedback from GPT-4

mechanisms to enhance the model's capacity for originality and harmonic diversity.

Conclusion

This study presents a new method for generating melodies, focusing on implementing the GPT architecture from scratch and building a vocabulary and encoder specifically designed for the melody generation task. Unlike previous methods, this study specifically targets melody generation using data from the Essen Folk Song Database. The dataset was converted from MIDI files to a text (.txt) format using the MidiTok library.

This study focuses solely on pitch values, disregarding duration and rhythm to simplify the melody generation task. After encoding, the data is structured into a vocabulary consisting of unique pitch values, which are then mapped to numerical tokens through a specially designed encoder. The implementation of the model from scratch allows for flexible customization of key parameters such as vocabulary size, embedding size, and output structure, optimizing the model's performance for this specific task.

The results show that the model is capable of generating melody sequences that demonstrate a certain level of musicality. The generated melodies often exhibit coherent melodic development, using motifs and repetition effectively. However, some cases of excessive repetition and predictable harmonic progression were observed, indicating areas for further improvement. Future research will focus on addressing the limitations identified in this experiment, particularly by exploring mechanisms to enhance the model's creativity and harmonic diversity, and apply the model to duration and rhythm to improve the performance of generated music.

Source code

Project code available at https://github.com/anhkhoiphan/Fine-tuning-GPT-2-for-melody-generation

Acknowledgement

We would like to express our sincere gratitude to our lecturer, Mr. Thanh Tuan Nguyen, for his invaluable guidance, support, and encouragement throughout this course and the last 4 years. His passion for the subject and dedication to his students' learning were truly inspiring. We would also like to thank our teaching assistant, Mr. Thanh Dat Nguyen, for his patient assistance and helpful feedback. His willingness to go the extra mile to ensure our understanding was greatly appreciated.

References

[1] T. W. Adorno and S. Gillespie. Music, language, and composition. *Music Quarterly*, 77:401–414, 1993.

[2] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint*, January 9 2019.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, October 11 2018.

[4] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever. Jukebox: A generative model for music. *arXiv preprint*, April 30 2020.

[5] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley. Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training. *arXiv preprint*, July 10 2019.

[6] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. 2017.

[7] Sebastian Eck. Essen folksong database (conversion & tokenization), March 12 2024. Available at https://www.kaggle.com/datasets/sebastianeck/essen-folksong-database-conversion-and-tokenization.

[8] P. Ferreira, R. Limongi, and L. P. Fávero. Generating music with data: Application of deep learning models for symbolic music composition. *Applied Sciences*, 13(7):4543, 2023.

[9] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, pages 1735–1780, 1997.

[10] A. D. Patel. *Music, Language, and the Brain*. Oxford University Press, Oxford, UK, 2010.

[11] Christine Payne. Musenet, 2019. Available at https://openai.com/blog/musenet/.

[12] Lawrence Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, 1986.

[13] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. OpenAI Technical Report.

[14] Sebastian Raschka. *Buiding a Large Language Model (From Scratch)*. Manning Publisher, 2024.

[15] Adam Roberts, Jesse Engel, Colin Raffel, and Curtis. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[16] S. Skúli. Music generation using a lstm neural network in keras, 2017. [Online]. Available: https://medium.com/@fernandorojas$_5$1691/$generating - music - with - neural - networks - a - journey - through - lstm - transformer - and - gpt - 2 - models - 0fdf39f20453$.

[17] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. 2017.