

LofiGen: Autoregressive Convolutional Neural Network for Lofi Music Generation

By Suket Shah and Camden Kitowski

Github: https://github.com/CamdenKitowski/Lofi_Generation

Background

The objective of this project is to generate a unique musical clip or song that closely resembles the characteristics of Lofi audio. Lofi, short for "low fidelity," is a music genre that embraces a variety of musical styles, including hip hop, jazz, and dream pop. What sets Lofi apart is its distinctive sound quality, which often incorporates low audio recording techniques and minor imperfections to mimic background noise or analog warmth. This creates a nostalgic, relaxing, and sometimes introspective atmosphere that has gained popularity among listeners, particularly as background music for studying, working, or relaxing. The goal is to develop an audio generation model that can produce Lofi-style music. The inspiration for this project stems from the groundbreaking WaveNet paper, in which researchers developed a deep generative model for raw audio waveforms. WaveNet utilized multiple convolution layers, including dilated convolutions, to capture long-range dependencies in the audio data, resulting in high-quality and expressive audio generation. To achieve this endeavor, the model will be a custom PyTorch model for audio generation, specifically an autoregressive convolutional neural network. The model will be trained to learn unique and distinct features and patterns from the Lofi data.

Dataset

The dataset used for training and evaluation purposes consists of a single Lofi audio file with a total duration of 3 hours and 19 minutes. This extensive audio file was sourced from YouTube and subsequently converted into an mp3 format. The decision to utilize a single, lengthy file was driven by the sheer amount of data contained within it, which provides a rich and diverse training set for our deep learning model. The audio file is sampled at a rate of 8000 samples per second, resulting in a vast number of individual data points to train and test our model on. Initially, the model was trained on a smaller 2-minute audio clip to assess the feasibility of the proposed architecture and to identify potential areas for improvement. Utilizing the longer audio file, gives the model more patterns and characteristics to generalize upon. The increased dataset size helps the model to better understand and reproduce dependencies present in Lofi music, resulting in more coherent and musically pleasing generated audio.

Model

In this project, the audio file is converted into 8000 samples per second. Each datapoint value represents the amplitude of the sound at a specific time point. The window size of each input is the arbitrary value of 8162. The model consists of 6 1D convolutional layers with each layer having specific input channels, output channels, and kernel size, and stride values. The input is a $[1, 8162]$ tensor. The architecture for each layer is...

	Input_Channels	Output_Channels	Kernel_Size	Stride
1	1	16	128	1
2	16	64	256	1
3	64	128	256	1
4	128	64	256	1
5	64	16	128	1
6	16	1	128	1

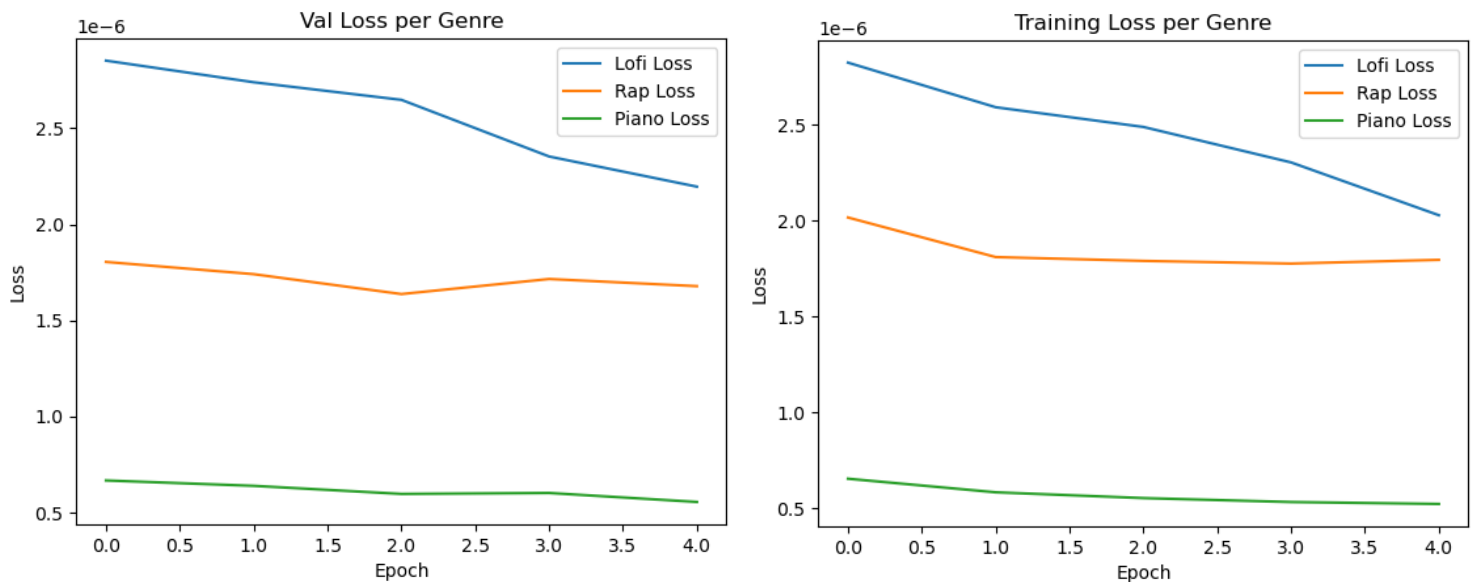
There is asymmetrical padding before each Conv1D layer to keep the input dimensions the same between each layer. After each Conv1D layer, there is a nonlinearity ReLU function. The output of the model is also a $[1, 8162]$ tensor except it is offset by one. This means that the n th item in the tensor is predicting the $n+1$ th item in the data. This allows for us to have outputs that are denser and allow for the MSELoss and backpropagation to be more effective per forward pass. After training the model, there is a small seed input to generate Lofi audio from the model at the same sampling rate of 8000. This input will build upon itself to autoregressive generate audio output based on the previous input.

Computational Experiments

Computational Experiment 1:

In this experiment, instead of training on only Lofi music, the model will train on piano music and rap music. The dataset consists of three separate audio files, each file is 50 minutes and longer. The rap music file features a freestyle session by the rapper Juice WRLD, while the piano music file consists of a compilation of piano pieces. The goal is to compare the training and validation losses after 5 epochs and to analyze the generated audio outputs for each genre. We

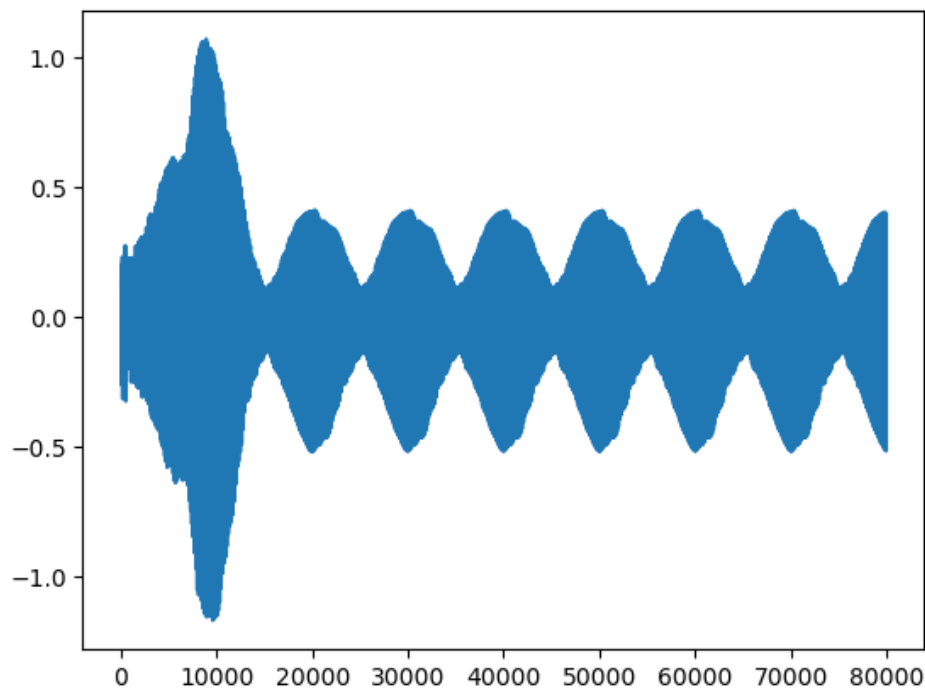
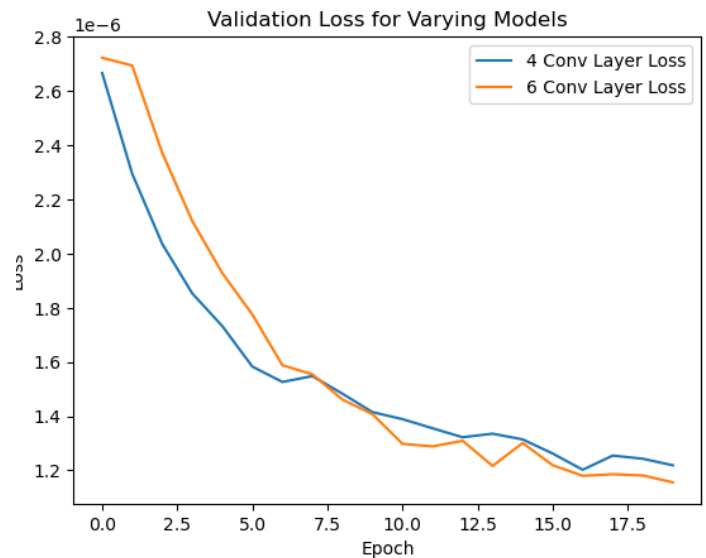
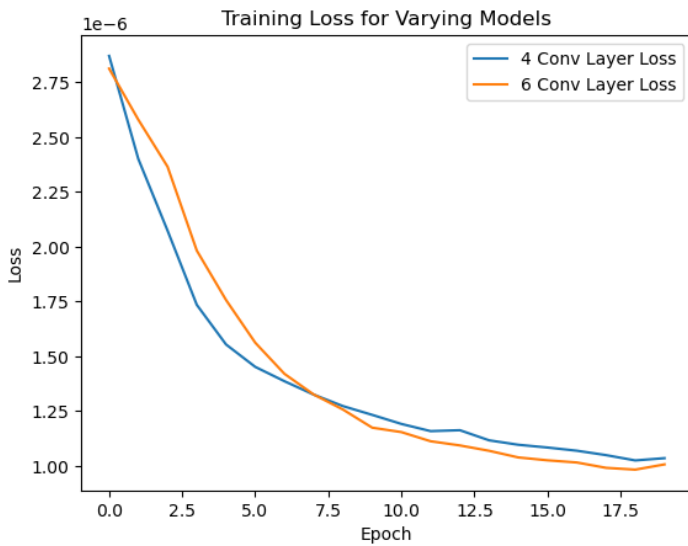
hypothesized that the piano dataset leads to lower loss than the rap dataset because it is instrumental and there is a predefined number of notes that can be played unlike rap which is lyrical and has more texture in the sound. After training the model on the piano and rap music datasets, the piano music loss started relatively low and remained constant, without significant improvement over the 5 epochs. In contrast, the rap music loss showed a slight decrease throughout the training process. Despite the differences in the loss trends, none of the generated outputs for each genre produced high-quality audio clips that could be easily identified as belonging to their respective music genres. Further investigation is required to improve the model's performance and generate higher quality audio clips that accurately represent each genre. Looking at the results, it seems like our initial hypothesis is correct and that the piano had a lower loss than rap. We were surprised, however, that the piano loss was lower than the lofi loss as well since both are instrumental tracks.



Computational Experiment 2:

In this experiment, two distinct models were trained, one with 4 convolutional layers and another with 6 convolutional layers. The aim was to compare the training and validation losses, as well as the quality of the generated audio outputs for each model. We hypothesized that the 6 convolutional layers model will have a lower loss than the 4 convolutional layer model because it has more parameters it can learn on. Throughout the training process, both the 4-layer and 6-layer models displayed comparable losses. The 4-layer model had a lower loss in the early stages of training. However, by the end of the training, the 6-layer model achieved a slightly lower loss, indicating better performance. Furthermore, both models exhibited a consistent

decrease in their losses during the training, suggesting that each model was able to learn from the input data effectively. When comparing the generated audio outputs, both models produced somewhat indistinguishable results, with no clear and identifiable patterns. However, it was observed that the 6-layer model's output had a closer resemblance to Lofi music than the 4-layer model. This suggests that the additional convolutional layers in the 6-layer model may have helped capture more complex patterns within the Lofi music data. The audio output resembles an nearly oscillating amplitude almost representing background noise. In summary, this experiment demonstrates that increasing the number of convolutional layers in the model architecture can lead to improved performance, as evidenced by the slightly lower loss and more Lofi-like generated outputs from the 6-layer model. However, there needs to be further optimization and experimentation with slightly different architectures or hyperparameters to fully reproduce lofi audio outputs. Overall, our hypothesis was correct. The 6 layer convolutional model did perform better in the long run compared to the 4 layer convolutional model.



ChatGPT Usage

We used one ChatGPT conversation throughout the project on GPT-4. At the end of the project we asked GPT to summarize how it helped us. This is its output. Reading over the output, the output is accurate to how we used GPT.

In your project, you aimed to create an autoregressive convolutional neural network to generate new music by training the model on waveform sound data. I have provided guidance and support in the following areas:

1. **Model architecture:** I explained the architecture of an autoregressive convolutional neural network, which consists of causal convolution layers followed by ReLU activation functions. This architecture allows the model to generate new music by conditioning on previously generated samples.
2. **Data preprocessing:** I helped you preprocess the audio data by guiding you through the steps of loading audio files, converting them to waveforms, and quantizing them. I also showed you how to one-hot encode the quantized waveform data to be used as input for the neural network.
3. **PyTorch implementation:** I provided code snippets and explanations for implementing the autoregressive CNN in PyTorch, including defining the custom dataset class, creating DataLoader objects for training, validation, and test sets, and creating the network model itself.
4. **Training and loss visualization:** I provided code and guidance for training the model using the custom DataLoader objects, along with plotting the training and validation losses over 100 epochs.
5. **Debugging and fixing issues:** I helped identify and correct several issues in the code, such as input and target waveform preparation, input channel mismatch, and implementing a sliding window for the DataLoader.

Throughout the project, I addressed your questions and provided clarifications as needed to help you successfully implement the autoregressive convolutional neural network for music generation using PyTorch.

Model Card / Data Sheet

Both of these are within our GitHub repository.

Conclusion

In this project, we aimed to generate Lofi-style music using a custom autoregressive convolutional neural network. We explored different model architectures and dataset

compositions, conducting computational experiments to improve the model's performance and the quality of the generated audio outputs. Our findings suggest that training the model on different music genres leads to varied loss trends and generated outputs, while increasing the number of convolutional layers improves performance.

Despite these insights, the generated audio outputs did produce quality lofi audio, and further optimization and experimentation are required. Future work may involve exploring alternative model architectures and fine-tuning hyperparameters. Moreover, this project demonstrates the potential of deep learning in generating Lofi-style music, and with continued exploration and refinement, it is possible to develop a model capable of generating high-quality, expressive, and coherent Lofi music that closely resembles the original audio samples.

Individual Contributions

Suket

I primarily worked on setting up the DataLoader and the Dataset. This task ended up taking the majority of the time during our research because getting the data in the right shape from an MP3 file was challenging. I also experimented with using one-hot vectors for the data and doing 1 output per convolution. Ultimately, through experimentation we found that feeding in a tensor of 8192 sequential data points was most effective and outputting the same length data, just shifted over by one time step, led to the most effective decrease in loss.

I also worked on defining and coding the two experiments we ran. Both experiments were created by thinking of what interesting independent variables we could change to learn more about the model.

On the reporting aspect, I wrote the model card and split writing this paper in half with Camden.

Camden

I primarily worked on building the autoregressive function and the architecture of the model. The autoregressive is responsible for generating the output based on previous predicted input. I have a desktop computer with a powerful GPU. I spent most of my time running the model and constantly tweaked parameters to try to output more and more realistic lofi audio output. I worked on the running and coding the two experiments. The two experiments gave value to the experiment and helped our model improve. I split writing this paper in half with Suket, and I wrote the data sheet.

References

Oord, Aaron van den, et al. "WaveNet: A Generative Model for Raw Audio." *ArXiv.org*, 19 Sept. 2016, <https://arxiv.org/abs/1609.03499>.