CSE 676:Group 12: Music Generation using Mamba

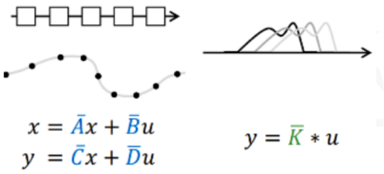
Mohammad Nafis Alam

Professor : Dr. Kaiyi Ji
malam22@buffalo.edu

Abstract

To utilize recurrent dynamics of RNN and parallelism of CNN, Mamba introduce an novel approach that utilize State Space Model for quicker training and efficient model.

We have designed a model using Mamba Block to test its claim of better context learning. In this project , we have gathered audio inputs in form of tokens and trained the model to generate new tokens. The audio signal is transformed to two set of tokens (Acoustic and Semantic). We also, trained the same input in model which uses transformer in order to do comparison in terms of memory utilization.



State Space Representation

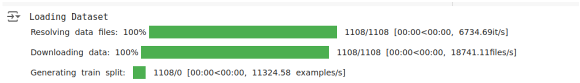
$$y_k = \overline{\boldsymbol{C}\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}\boldsymbol{B}}u_k$$
$$y = \overline{\boldsymbol{K}} * u$$

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L = (\overline{\boldsymbol{C}\boldsymbol{B}}, \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}})$$

Kernel Fusion

1 DataSet

Using pytube library, we downloaded all songs from a playlist from youtube and saved in google drive. Then , we splitted the mp4 files to have only 10 sec length. Then , with help of torchaudio library we resampled the audio files to have custom sample rate(16000). After getting the normalized waveform, we used pretrained model from SpeechTokensizer from HuggingFace to get two set of tokens that provided semantic and acoustic information respectively . Dataset library is used to enumerate the music directory more conveniently.



```
Loading Dataset
  Resolving data files: 100%  1108/1108 [00:00<00:00, 6734.69it/s]
  Downloading data: 100%  1108/1108 [00:00<00:00, 18741.11files/s]
  Generating train split:  1108/0 [00:00<00:00, 11324.58 examples/s]
```

2 Model Description

MambaAudio model is composed of a Feed Forward Class and Block Class. Following are its working.

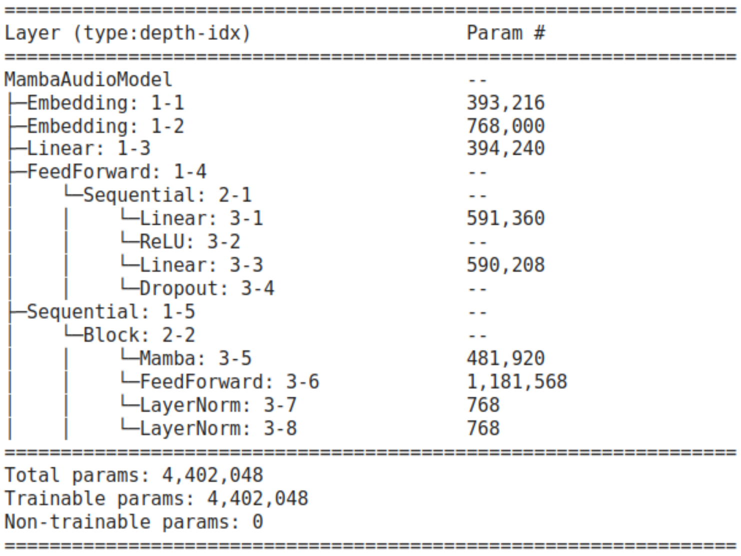
FeedForward: This class defines a simple feedforward neural network with two linear layers and a ReLU activation function. It takes the input dimension n_embed and creates a feedforward network with the specified architecture.

Block: This class represents a Mamba block, consisting of a (Mamba class), a feedforward neural network (FeedForward), and layer normalization layers (nn.LayerNorm). The result from Mamba block is followed by layer normalization, and then the feedforward network is applied followed by another layer normalization.

MambaAudioModel: This class represents the main model, which includes token embeddings, positional embeddings, linear layer (lm_head), and a stack of Mamba blocks (blocks). It also defines the forward method, which takes input indices (idx) and optionally target indices (targets). It first retrieves token embeddings and positional embeddings, adds them together, and passes the result through the stack of Mamba blocks. Finally, it applies a linear layer to get logits, calculates the loss if targets are provided, and returns logits and loss.

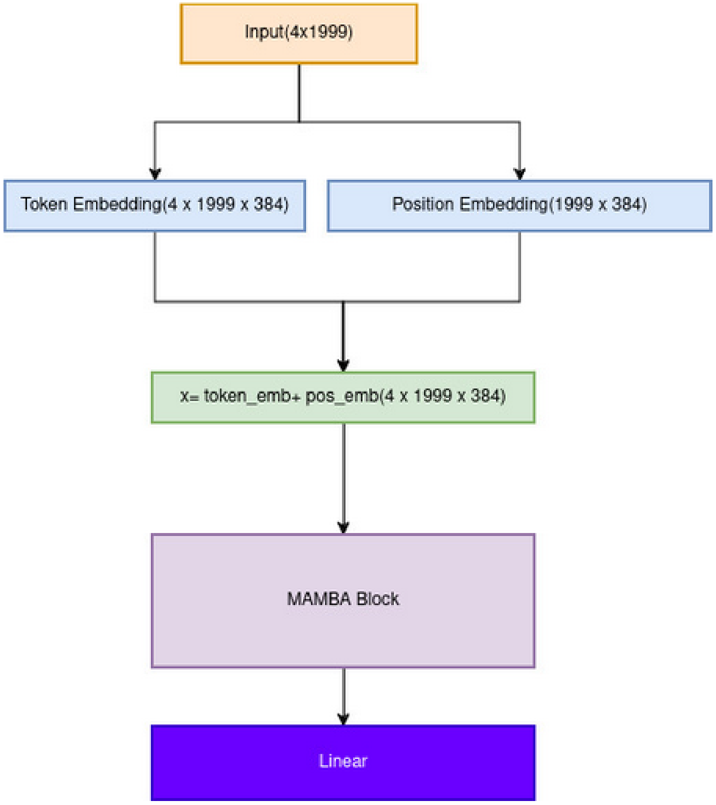
We took vocab size of 1024 and embedding to bebe 384. For Mamba block, we transformed our tensors to shape [4,1999,384]. After the computation , we get output of [4,1999,384]. We used first 1999 from series of 2000 to generate last 1999 of the series.

A function produce_wav use multinomial fucntion to choose the right predictions from vocabulary using the probability. We then decode the tokens using the earlier used SpeechTokenizer pretrained model to produce the wav file.



```
=================================================================
Layer (type:depth-idx)                   Param #
=================================================================
MambaAudioModel                          --
├─Embedding: 1-1                         393,216
├─Embedding: 1-2                         768,000
├─Linear: 1-3                            394,240
├─FeedForward: 1-4                       --
│    └─Sequential: 2-1                   --
│    │    └─Linear: 3-1                  591,360
│    │    └─ReLU: 3-2                    --
│    │    └─Linear: 3-3                  590,208
│    │    └─Dropout: 3-4                 --
├─Sequential: 1-5                        --
│    └─Block: 2-2                        --
│    │    └─Mamba: 3-5                   481,920
│    │    └─FeedForward: 3-6             1,181,568
│    │    └─LayerNorm: 3-7               768
│    │    └─LayerNorm: 3-8               768
=================================================================
Total params: 4,402,048
Trainable params: 4,402,048
Non-trainable params: 0
=================================================================
```

Pytorch Model


Input(4x1999)
Token Embedding(4 x 1999 x 384)
Position Embedding(1999 x 384)
x= token_emb+ pos_emb(4 x 1999 x 384)
MAMBA Block
Linear

Model Overview

3 Loss Function

Since, token generator which comprises token from set vocabulary, it is similar to multiclass prediction. In the context of token prediction with a token generator, choosing the appropriate loss function depends on the specific characteristics of the task and the desired behavior of the model during training. Experimentation with different loss functions can help determine which one performs best for the given task and dataset.

cross_entropy

mse_loss

l1_loss

As cross-entropy measures the difference between the predicted probability distribution and the true distribution of class labels, it is more effective here. We chose it for our model with AdaW optimizer

4 Optimizer Algorithm

We experimented with multiple loss functions and got different trends in result. In our training we got lot of noise which is due to resampling of data. The data quality after transformation is not proper. We had to resample because of resource constraint. Mamba has a specific hardware requirement.

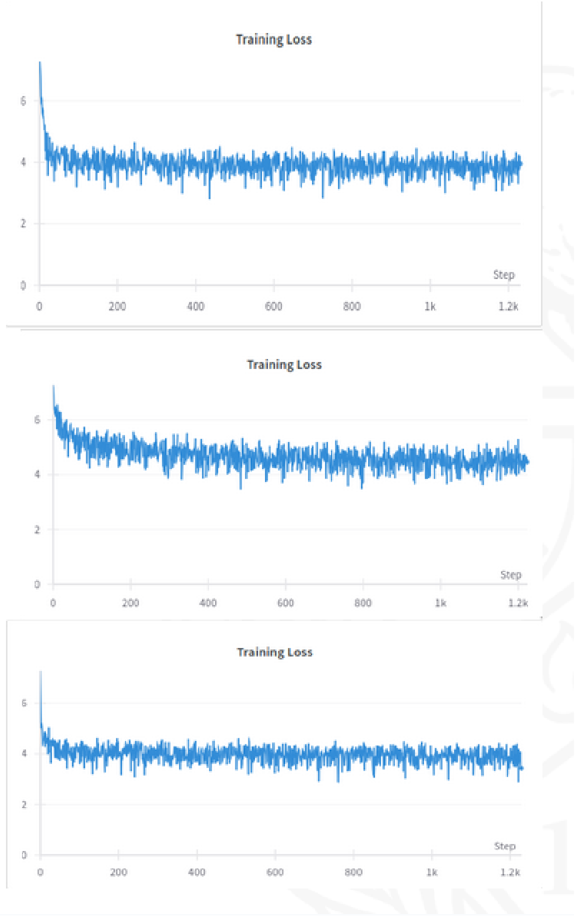
Adam :Adaptive learning rates, allowing the model to adjust the learning rate for each parameter individually based on historical gradients.

AdamW : Decouples Weight Decay,separating the weight decay regularization from the optimization steps, improving performance and stability.

SGD : Updates parameters in the opposite direction,an optimization algorithm that updates parameters in the opposite direction of the gradient, aiming to minimize the loss function iteratively.

Adagrad: Adapts the learning rate based on the frequency of parameters, allowing larger updates for infrequent parameters and smaller updates for frequent ones, potentially improving convergence for sparse data.

There was not any marginal difference , but comparatively AdamW converged quickly, although noise remains same.

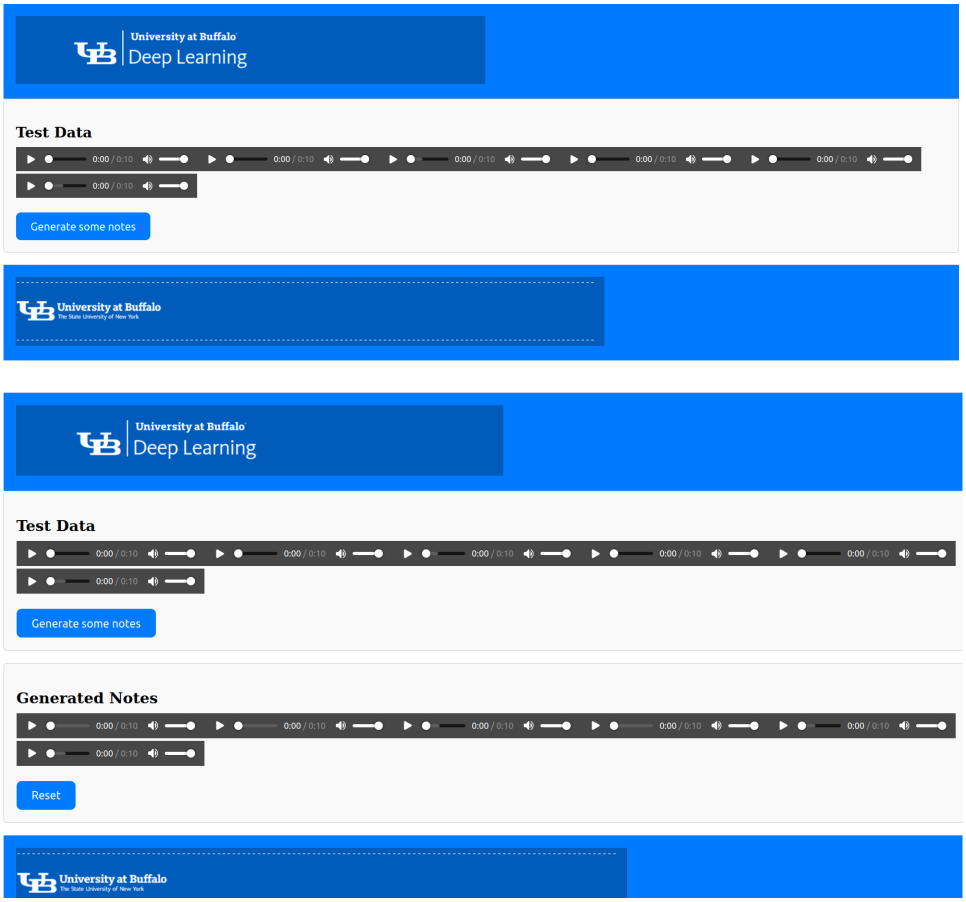


AdamW, SGD, AdaGrad (top to bottom order)

5 Metrics and Experimental Results

5.1 Demo

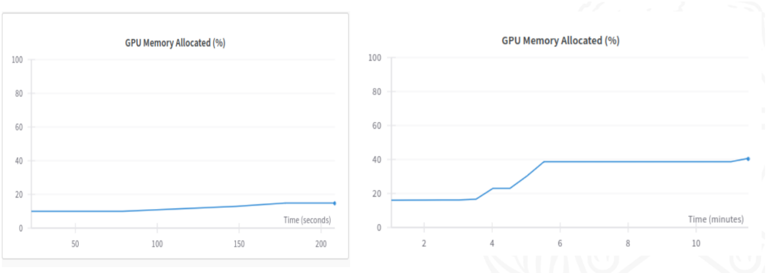
The UI shows the input 10 sec mp4 files and the result generated wav file.



UI interface to generate new music filed based on input files

5.1.1 Comparision

Since, Mamba uses Hardware-Aware State Expansion, it is less depedent on GPU.



The GPU utilisation comaprision between Mamba and transformer

6 Contributions and GitHub

`https://github.com/nafisdev/CSE676/`

This group has only one member so, everything is handled by the author.

References

The technicality of Mamba are provide in (1,2) and to run mamba on google colab, i used a workaround provided in (3).

[1] https://arxiv.org/abs/2312.00752

[2] https://github.com/state-spaces/mamba

[3] https://www.kaggle.com/code/wolfy73/run-mamba-on-p100

[4] https://srush.github.io/annotated-s4/

[5] https://www.youtube.com/watch?v=wjZofJX0v4Mpp=ygUcdHJhbnNmb3JtZXJzIDMgYmx1ZSAxIGJyb3duIA

[6] https://www.youtube.com/watch?v=vrF3MtGwD0Yt=774spp=ygUFbWFtYmE