Modeling Dynamic Range Compressor Using S4

Submitted in partial fulfillment of the requirements for the degree of

Master of Science in Music and Technology

School of Music

Carnegie Mellon University, Pittsburgh, PA

Hanzhi Yin

Thesis Committee:
Prof. Roger B. Dannenberg, Chair
Prof. Richard M. Stern

August 2023

Acknowledgements

I would like to show my most profound gratitude to Prof. Roger Dannenberg for his tutelage and supervision. Throughout the thesis, he listened to my ideas, discussed methodologies, followed my work, and provided feedback on my writing. His fundamental support has been indispensable for the completion of the thesis.

I would also like to thank Prof. Richard Stern for his reading and comments, Christian Steinmetz for his suggestions on the evaluation workflow and his previous work, Dr. Albert Gu for his remarks on S4, Dr. Chris Donahue for his initial thoughts on my thesis proposal, and Ruibin Yuan for generously making his personal computing resources available.

Finally, I would like to express my heartfelt appreciation to my family, whose altruistic financial and mental support allowed me to be who I am today. Never have I been so resolute in devoting myself to computer music, which I had never considered before adulthood. Your love and provision have set up the environment where I could grow and thrive.

Abstract

Analyzing analog dynamic range compressors and building faithful digital models is hard due to non-linear behaviors operating on long-time scales, and there is less attention being paid to modeling dynamic range compressors compared to other non-linear audio effects. The structured state space sequence model (S4) has proven to be efficient at learning long-range dependencies, which could be promising for modeling dynamic range compressors. For this work, I constructed a model mainly comprising S4 layers to model an analog dynamic range compressor, namely the Teletronix Universal Audio LA-2A compressor, a dynamic range compressor widely used in neural network-based virtual analog modeling research. This thesis presents model design, model hyper-parameter search, loss value analysis, comparison to the previous state-of-the-art, and a discussion on real-time inference. Surprisingly, a gray-box model, where the existence of the final multiplication operation allows S4 blocks to act as gain computers, does not perform as well in objective measures as a black-box model without the final multiplication operation, although the gray-box model seems to be more stable.

Contents

List of Tables

List of Figures

Chapter 1

The Introduction

The definition of the term "virtual analog" can be traced back to 1994, when in that year's National Association of Music Merchants (NAMM) show, the Swedish company Clavia introduced the NordLead synthesizer that introduces subtractive synthesis into digital synthesizer offerings [2, 3]. Subtractive synthesis, once a prominent analog synthesis method in the analog domain in the 1960s and 70s, was overshadowed by other digital synthesis techniques in the 1980s. However, it regained its attention in the 1990s as musicians showed interest in its "warm" sound quality. The Clavia company took the trend and introduced their NordLead synthesizer, along with the term "virtual analog", which concerns the digital simulations of analog audio devices, such as synthesizers and audio effect units. Introducing analog audio devices to the digital domain can bring lots of advantages. Musicians can free their labor by refraining from carrying bulky cargo, and digital audio units are not susceptible to environmental change, such as humidity and temperature [3].

Traditionally, virtual analog modeling (VA modeling) can be divided into three categories: the black-box approach, the white-box approach, and the grey-box approach. The categorization is based on how much the virtual analog model knows about the underlying physical structure [4]. While black-box models rely on general and abstract mathematical frameworks, such as the Wiener-Hammerstein system [5], without explicitly modeling underlying physical structures, white-box models utilize existing physical structures to derive digital models by obtaining the discrete-time transfer function from the continuous-time one [6] or by directly modeling the circuit structure [7]. Grey-box models, on the other hand, are intermediate between black-box models and white-box models.

Recently, there has been a trend in utilizing deep learning (DL) techniques in VA modeling. Using DL techniques like back-propagation and automatic differentiation, VA modeling research can be transformed into data-driven tasks by utilizing input-output waveform pairs produced by the target analog signal

system. The VA modeling research community has conducted several attempts at modeling vacuum-tube amplifiers [8, 9, 10, 11, 12] and distortion pedals [13, 14, 15, 16]. Some work also attempts to model multiple audio effects using one model [17]. Introducing VA modeling to deep learning may bring some research benefits. Audio synthesis and encoding strategies based on differentiable digital signal processing (DDSP) [18] proposed by Engel et al. could be freed from algorithmic signal processing components, and understandings learned from the research process can benefit DL audio synthesis tasks like automatic mixing [19, 20].

Compared to most other audio effects, dynamic range compressors (DRCs) are difficult to emulate due to their long temporal dependencies and non-linearity [17]. Yet, DRC is essential in audio production as it can minimize recording artifacts, reduce masking, and unify loudness among multiple audio channels [21]. DRCs can analyze the audio loudness and apply variable gain to reduce the loudness of the louder part and leave the quieter part untouched. Existing attempts include using an autoencoder architecture with multi-layer perceptron (MLP) [22], using temporal convolutional network (TCN) [1], and factoring several DRC working theories outlined by Giannoulis et al. [21] into the proposed model [23]. While these attempts investigated various aspects of the topic, there is still room to improve the objective numerical performance. The output generated by current models still exhibits artifacts. Some best-performing models either rely on hard-coded components or are non-causal and require an excessively large parameter space. Furthermore, compared to other non-linear audio effects, such as vacuum-tube amplifiers or distortion pedals, DRCs receive less attention. The call for a model with higher accuracy and perceptual quality, causal formulation, and parameter efficiency remains.

In recent DL studies, Gu et al. [24] proposed the structured state space sequence model (S4) for long-sequence modeling. Essentially a linear time-invariant (LTI) system implemented as an infinite impulse response (IIR) system in the state-space form, S4 is capable of producing a convolutional kernel that is arbitrarily long, the same as the input sequence, which has a more flexible receptive range compared to traditional CNN. S4 was once the state-of-the-art on Long Range Arena, a neural network model benchmark assessing "model quality under long-context scenarios" [25]. Thus far, no existing work discusses the effectiveness of introducing a state-space model (SSM) in VA modeling using DL techniques. Although some previous work applied SSM in audio synthesis [26], they did not apply SSMs in non-linear audio effects. They also did not discuss the effectiveness of applying SSM and other non-linear functions, which are prevalent in DL. Discussing and exploring the effectiveness of SSM in VA modeling, especially in non-linear audio effect modeling, seems promising for the evolution of audio effects modeling, computational audio synthesis, and, in general, utilizing DL techniques in pure-audio applications.

In my thesis, I propose to use S4, along with linear layers and non-linear activation functions, to model

DRCs. I have created a model that has S4 as a major component to fit the Teletronix Universal Audio LA-2A compressor using the SignalTrain dataset [27] that is widely adopted by major DRC modeling work. Numerous objective evaluations have been conducted to analyze the effectiveness of having S4 in a DL model, and several future directions have been outlined to promote the evolution of VA modeling using DL techniques. The thesis outline is as follows: Chapter 2 will present some in-depth analysis of some key related work and prerequisite theories, Chapter 3 will elucidate my experimental procedure, Chapter 4 will present all objective evaluation metrics along with some in-depth analyses of some of those metrics, and Chapter 5, the last chapter, will conclude the thesis and suggest possible future work.

Chapter 2

Related Work and Theory

2.1 Related Work in DRC Modeling

There are three significant contributions to modeling analog DRCs using DL techniques, all of which are essential to my thesis. I present a detailed view of previous work and show its connection to my thesis.

2.1.1 Hawley et al., 2019

The first attempt to investigate modeling DRCs using DL techniques is from Hawley et al. [22] in 2019. Subsequently, Mitchell and Hawley further explored the original model's possible improvements and optimization [28]. Their work proposed an auto-encoder model containing fully-connected layers as the encoder and the decoder. The model takes audio waveform, transforms it to magnitude and phase spectrograms, and passes them through encoders and decoders with the intermediate "bottleneck" as the auto-encoder latent space. Fully-connected layers apply an affine transformation on the spectrogram's frequency domain (the other domain is time). The output would be transformed back from the frequency domain to the time domain, meaning that both the input and output of the model are audio waveforms.

Audio effects usually have external controls to tune behavior. Hawley et al.'s model includes control parameters to change the DRC behavior, such as compression ratio. In their model, control parameters are concatenated to the latent space. The number of control parameters is flexible. In their work, they first modeled a software-based DRC implemented by themselves, and there are four control parameters. They also modeled an analog DRC, the Teletronix LA-2A optoelectonic compressor (LA-2A), and there are only two control parameters.

The SignalTrain Dataset

Hawley et al.'s second contribution is to present the SignalTrain dataset [27], which is a staple of modeling analog DRC using DL techniques [1, 23]. The SignalTrain dataset is a collection of recorded data from an LA-2A. LA-2A DRC is widely endorsed by musicians for its smooth, natural, and musical compression. There are four controls on LA-2A. The first is a switch toggling whether LA-2A should work in limiting mode or compressing mode. The peak reduction knob controls the threshold of the DRC. LA-2A will attenuate the input signal if the input signal's level is above the threshold. The value range from 0 to 100 corresponds to 0 dB to -40 dB. Intermediate values do not interpolate this range linearly. The gain knob applies the make-up gain to the output signal. The value range from 0 to 100 corresponds to 0 dB to 40 dB. Intermediate values do not interpolate this range linearly either. The last switch has three values, "output +4," "gain reduction," and "output +10," which control how the VU meter displays information. It is merely for data display and does not affect signal processing.

Audio samples present in the SignalTrain are either "randomly generated," downloaded audio clips with Creative Commons licenses, or are property of Scott Hawley and freely distributed as part of this dataset. There is an input-output signal pair for each of many combinations of the peak reduction knob settings and the limiting/compressing switch. Only the peak reduction knob and the limiting/compressing switch are varied. Other knobs are kept constant. Colburn and Hawley did not disclose the value they used. For the limiting/compressing switch, value 0 denotes "compress," and value 1 denotes "limit." For the peak reduction knob, only multiples of 5 are represented. In total, there are 40 possible control parameter combinations.

All audio data is mono with a sample rate of 44.1 kHz. The training, validation, and testing split are pre-defined by Colburn and Hawley. There is 70 139.22 s audio data in the training split, 14 699.81 s audio data in the validation split, and 2700.00 s audio data in the testing split. Training, validation, and testing audio data control parameters are not entirely independent. All testing audio data control parameters are overlaid with the training audio data. Some validation audio data control parameters are overlaid with the training audio data, but others are independent. Please refer to Table 2.1 for detailed information.

Regarding the total audio length for each control parameter, almost all audio data under the same control parameter have the same audio length of 2100.0 s, with two exceptions. The audio data with control parameter $(0, 70)$ has only 1199.81 s, and the audio data with control parameter $(1, 100)$ has 2340.0 s. Both control parameter pairs only exist in the training dataset split.

Compressing/limiting Switch	Peak Reduction	Dataset Split
0	0	'Train'
0	15	'Train'
0	20	'Train'
0	30	'Train'
0	35	'Train'
0	40	'Train'
0	55	'Train'
0	60	'Train'
0	70	'Train'
0	75	'Train'
0	80	'Train'
0	90	'Train'
0	95	'Train'
0	100	'Train'
1	0	'Train'
1	5	'Train'
1	15	'Train'
1	20	'Train'
1	30	'Train'
1	35	'Train'
1	40	'Train'
1	50	'Train'
1	55	'Train'
1	60	'Train'
1	75	'Train'
1	90	'Train'
1	95	'Train'
1	100	'Train'
0	10	'Val'
0	50	'Val'
1	10	'Val'
1	70	'Val'
0	5	'Val', 'Train'
0	25	'Val', 'Train'
0	45	'Val', 'Train'
0	85	'Val', 'Train'
1	25	'Val', 'Train'
1	45	'Val', 'Train'
1	85	'Val', 'Train'
0	65	'Test', 'Train'
1	65	'Test', 'Train'
1	80	'Test', 'Train'

Table 2.1: SignalTrain Dataset Control Parameter Distribution among Dataset Splits

2.1.2 Steinmetz and Reiss, 2021

Steinmetz and Reiss [1] proposed a TCN-based model to model the same LA-2A compressor, also using the SignalTrain dataset. TCN is a type of CNN that is guaranteed to keep the input data dimension in the layer output by padding zeros at the front (causal) or the end of the input data (non-causal), and it has been proven to be successful in sequence modeling [29]. Steinmetz and Reiss's work mainly tackles the model causality, inference efficiency, and output quality.

The TCN model proposed by Steinmetz and Reiss comprises a chain of TCN blocks with audio channel and receptive field dilation. The model takes audio waveforms as the input and output without transforming to the frequency domain, in contrast to the work of Hawley et al.. Each TCN block contains a one-dimension convolutional layer, a one-dimension BatchNorm layer normalizing channel information by treating audio channels as features, a feature-wise linear modulation (FiLM) layer, allowing external control parameters to tune the model behavior, and a non-linear activation function, using parameterized rectified linear unit (PReLU) functions. Residual connection is applied within each TCN block. Between the last TCN block and the audio waveform output, a linear layer contracts multiple intermediate channels to one channel with an extra hyperbolic tangent function to softly normalize the audio to ± 1.

The FiLM layer, initially proposed by Perez et al., is a conditioning layer that can factor external conditional information into a feature vector or a sequence of feature vectors [30]. It is initially used in computer vision tasks, such as image captioning, to factor in extra verbal information to push the model to focus on specific image areas. Given external information c, FiLM would process it into two vectors, γ and β, with the same feature size as the input vector via an MLP. It then processes the input vector x as Equation 2.1 demonstrates.

$$y = x \odot \beta + \gamma \tag{2.1}$$

where y is the output, and $\odot$ is element-wise vector multiplication.

When applying FiLM to TCN blocks, audio channels are treated as features. Each FiLM layer applies conditional information γ and β to all time stamps.

2.1.3 Wright et al., 2022

Wright et al. [23] also used the SignalTrain dataset to propose a grey-box model for modeling DRCs. They followed the DRC implementation structure outlined by Giannoulis et al. [21]. According to Giannoulis et al., the signal entering a DRC is split into two copies. One is sent to a variable-gain amplifier and the other to the side-chain. The side-chain is responsible for extracting loudness information from the input

audio waveform using a static gain computer, mapping the input decibel level to the expected output decibel level, and a level detector applying smoothing to prevent a sudden change in gain. The input signal is multiplied by the gain computed in the side-chain. Some DRCs have a final make-up gain component to add gain to the output signal to compensate for the reduced general loudness. As Giannoulis et al. pointed out, an ideal digital DRC can be expressed by equation 2.2.

$$y[n] = x[n] * g[n] \tag{2.2}$$

$y[n]$ is the discrete output signal, $x[n]$ is the input signal, and $g[n]$ is the variable gain calculated from the side-chain.

Wright et al.'s model explicitly factored these components mentioned above. Their model comprises a specially crafted data flow similar to the side-chain and variable-gain amplifier logic. In the side chain, the signal is firstly converted to decibels. Then, a static gain computer takes coefficients transformed from external LA-2A parameters using MLPs and use these coefficients to formulate a function to process the input sequence. In the side-chain, a level-detector is implemented either as a one-pole filter, a switching one-pole filter, or an Elman Network [31]. Before the end of the model, a make-up gain component is applied, either as a static gain controller or a GRU.

Unlike previous models, Wright et al. contains many components directly referred from DRC working theories with hand-crafted functions and explicit data flow. Such a method makes the model no longer a black-box model. However, since the model may use the Elman network or GRU, the model is also not a white-box model. Therefore, the model falls into the grey-box model category.

2.2 Structured State Space Sequence Model

Infinite impulse response (IIR) systems are characterized by their infinitely long impulse response. There are various ways to use a finite number of parameters to express an IIR system. One of them is the transfer-function form, and another is the state-space model (SSM) form, which can be expressed as Equation 2.3

$$\begin{aligned} x[t] &= \mathbf{A}x[t-1] + \mathbf{B}u[t] \\ y[t] &= \mathbf{C}x[t] + \mathbf{D}u[t] \end{aligned} \tag{2.3}$$

In the above Equation 2.3, t is the time, u is the input discrete signal, x is the intermediate discrete signal, and y is the output discrete signal. While $u[t]$ and $y[t]$ are usually a scalar, $x[t]$ can be a column vector, with the length N to be the IIR filter order. $\mathbf{A}(N \times N), \mathbf{B}(N \times 1), \mathbf{C}(1 \times N), \mathbf{D}(1 \times 1)$ are matrices

expressing linear mappings. An SSM takes the current input value $u[t]$ and the previous hidden vector $x[t-1]$ to formulate the current hidden vector $x[t]$ and then takes the current hidden vector $x[t]$ with the current input value $u[t]$ to formulate the current output $y[t]$. When $t = 0$, $x[-1] = \mathbf{0}^{N\times 1}$.

Proposed by Gu et al. [24], the structured state space sequence model (S4) is a neural network layer that directly implements an IIR system in the SSM form for sequence modeling, mapping input sequence u to output sequence y. In the language of deep learning, the sequence x "memorizes" previous input information and passes them to the current to control the current output value and associate the current input value, similar in concept to recurrent neural network (RNN) layers.

The stability of an IIR system, which means the output value must have an upper bound, is vital. Directly introducing IIR SSM without regularization could lead to vanishing/exploding gradient issues in neural network training. To tackle this, Gu et al. initially introduced various mathematical methods called HiPPO to ensure the system is stable throughout the training process [32]. Primarily, they directly tackled the matrix $\mathbf{A}$, which controls how the previously hidden information is passed to the current hidden information. An even more parameter-efficient approach has been proposed by Gupta et al., who proposed to diagonalize matrix $\mathbf{A}$ [33].

S4 is usually compared with TCN, with both LTI systems able to model sequences of arbitrary length. The significant difference is that TCN, a variant of CNN, is a finite impulse response (FIR) system, whereas S4 is IIR. Usually, to express a filter with a similar effect, IIR systems require fewer filter orders, resulting in a more parameter-efficient neural network layer. Besides, IIR systems' impulse response decay is arbitrary and more flexible than FIR systems, whose impulse response length is fixed.

In deep learning, CNNs are conventionally implemented to be able to mix input sequence channels. Major CNN layer implementations can mix sequence channels and alter the number of sequence channels. In contrast, S4 does not mix sequence channels or alter the number of sequence channels. S4 produces a dedicated filter for each sequence channel and only uses that filter to convolve with that particular channel. How S4 works does not align with how CNNs are conventionally implemented in deep learning but aligns with digital signal processing conventions. In my model, some fully connected layers are prepended to mix channels.

To ensure S4 layer training is as efficient as possible, internal SSM matrices in an S4 layer are implemented as complex numbers. That is, $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\mathbf{B} \in \mathbb{C}^{N\times 1}$, $\mathbf{C} \in \mathbb{C}^{1\times N}$, $\mathbf{D} \in \mathbb{C}^{1\times 1}$. Implementing these matrices in complex numbers allows these matrices to be efficiently converted to an impulse response using specific mathematical techniques [34]. The generated impulse response can have an arbitrary length that could be as long as the sequence to be processed so that fast Fourier transform (FFT) convolution can be applied. S4 is implemented to be causal. It is worth noting that an IIR system expressed in SSM can

always be transferred to the transfer-function form. While the current S4 implementation uses the same process for training and inference, it is possible to convert all internal S4 parameters to transfer-function coefficients to save computation load during inference.

Chapter 3

Experiments

This chapter will cover all information related to experiments being carried out throughout the thesis, including the model that I designed, the detailed experiment procedure, all customizable hyper-parameters, and the evaluation procedure.

3.1 General Experiment Procedure

Following the existing work [35, 1, 23] in modeling dynamic range compressors, I also use the SignalTrain dataset [27] for training, validating, and testing my proposed model. The SignalTrain dataset is enormous, so it takes a long time to train a model using the entire SignalTrain dataset. To speed up the search for good hyperparameters, instead of directly going for a conditional model, I created a simplified model without control parameters to conduct essential experiments for hyper-parameter searching. After determining promising hyperparameters, I trained complete models, including control parameters.

The general experiment procedure comprises two stages. First, a fixed-parameter model would be trained on a particular SignalTrain dataset control parameter. With a set of hyper-parameters being treated as the cardinal reference, one hyper-parameter would be altered with all others untouched to evaluate the impact caused by that hyper-parameter. After some understanding of how each hyper-parameter affects the model performance has been obtained, those understandings can be migrated to the conditional model that takes control parameters. Further ablation study shall be conducted on the conditional model if necessary.

3.2 Model Design

The fixed-parameter and conditional models are end-to-end neural networks with time-domain audio samples as both the input and the output. Both models can only accept mono audio with the amplitude

range of ± 1.0 as 32-bit floating point numbers. Although S4 layers are IIR systems by design, an S4 layer generates an impulse response to convolve with the signal in one forward step instead of taking one sample at a time. Therefore, input audio must be handled as one complete vector, and neither model is real-time. However, we will consider real-time implementations in Sec. 4.2.3.

Throughout the entire experiment, all S4 layers incorporated in both models are S4D layers [33]. S4 blocks in each model may be repeated several times to formulate a stack.

3.2.1 Fixed-parameter Model Design

The fixed-parameter model (Figure 3.1) design can be seen in Figure 3.1. The parameter c in all linear layers are audio channels. `Act.` means non-linear activation layers. Linear means linear layers. Components marked in dashed lines are omittable, and components filled in gray are substitutable.

The model design is chiefly inspired by Steinmetz and Reiss [1]. Since S4 applies a filter to each audio channel independently without mixing the channel, unlike traditional convolution layers in deep learning, linear layers are introduced to S4 Blocks and the entire model. They are responsible for expanding, mixing, and contracting the audio channel. The first linear layer, suffixed with $(1 * c)$, expands each time sample's mono channel to a fixed value c greater than 1. Intermediate linear layers, suffixed with $(c * c)$ in S4 blocks, mix c channels to another c. The final linear layer contracts c channels to mono.

Throughout the model, there is no channel dilation within each S4 block. All S4 layers have the same number of parameters, meaning they process the same number of channels using the same IIR filter order.

An interesting model design that can be experimented with is to introduce a multiplication operation at the end. As previously mentioned in 2.2, an ideal digital DRC can be considered as the multiplication of two signals, with one to be the original input signal and the other to be considered as the variable gain. The multiplication operation does not introduce any extra model parameters, but it can alter how those S4 layers behave completely. Without the final multiplication operation, S4 does not process the input audio directly to produce output audio, but with the final multiplication operation, S4 is responsible for taking the input audio and computing variable gain. The model without the final multiplication operation can be considered as a black-box model, whereas the model without it can be considered as a gray-box model. Whether having the final multiplication operation is effective or not is discussed in the ablation study.

`Tanh` stands for hyperbolic tangent function layer. It is introduced before the processed signal is multiplied by the original signal. The layer can be removed. Whether having it is effective or not is discussed in the ablation study.

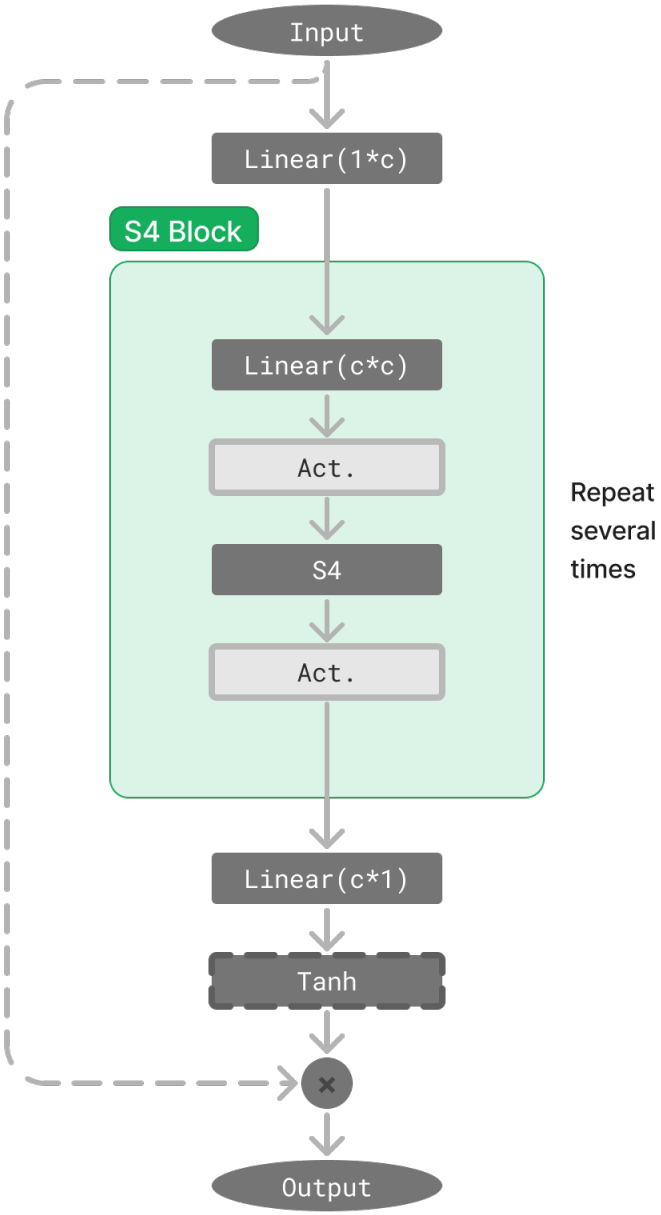


Figure 3.1: The Fixed-parameter Model

3.2.2 Conditional Model Design

The conditional model (Figure 3.2) is built on top of the fixed-parameter model.

To apply conditional information, one-dimensional BatchNorm and FiLM layers are appended after each S4 layer. All BatchNorm layers in this model apply batch normalization to each time sample by treating audio channels as features. After BatchNorm, FiLM layers apply conditional model information to the audio signal. The processed conditional information is then applied to each time sample by affine transformations, also treating audio channels as features.

As in the SignalTrain dataset, there are two external controls: The compressing/limiting switch and the gain reduction. These two values are passed to a three-layer multi-layer perceptron (MLP) illustrated at the right of Figure 3.2. The conditional information is only processed by the simple MLP once. The same MLP output is applied to all FiLM layers.

Residual connections are added to each S4 block to ease the training by default without substitution.

3.3 Experiment Procedure

All experiments are conducted on a virtual machine with Nvidia A10 GPU and 24 GB GPU RAM. The virtual machine's CPU is Intel Xeon Platinum 8358.

3.3.1 Fixed-parameter Model Ablation Study

To train the fixed-parameter model, we selected the data from the SignalTrain dataset with the control parameter (1, 100). As previously mentioned, audio data under that control parameter has 2340.0 s of audio, the longest among the dataset. File `input_179_.wav`, with 20 minutes of audio, and file `input_263_.wav`, with 15 minutes of audio, are used for training data. File `input_221_.wav`, with 4 minutes of audio, is used for validation data. There is no testing split as this experiment is only meant for hyper-parameter search. When training, we sliced the audio to 1.0 s chunks, and when validating, we sliced the audio to 30.0 s chunks.

One (and only one) data augmentation technique is applied during training: phase inversion, with a probability of 0.5. This is directly referenced from Steinmetz and Reiss [1].

All models are trained for 70 epochs, with a learning rate of 1×10^{-3} applied to all layers, as suggested by Gu et al. when they were training their S4 models [24]. There is no learning rate scheduler incorporated. The batch size has been set to 32. We use AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e - 8$, and the default weight decay 1×10^{-2}, with the exception that all parameters in S4 layers have zero weight decay, as implemented by Gu et al. [24]. The model is trained using single-precision floating point numbers.

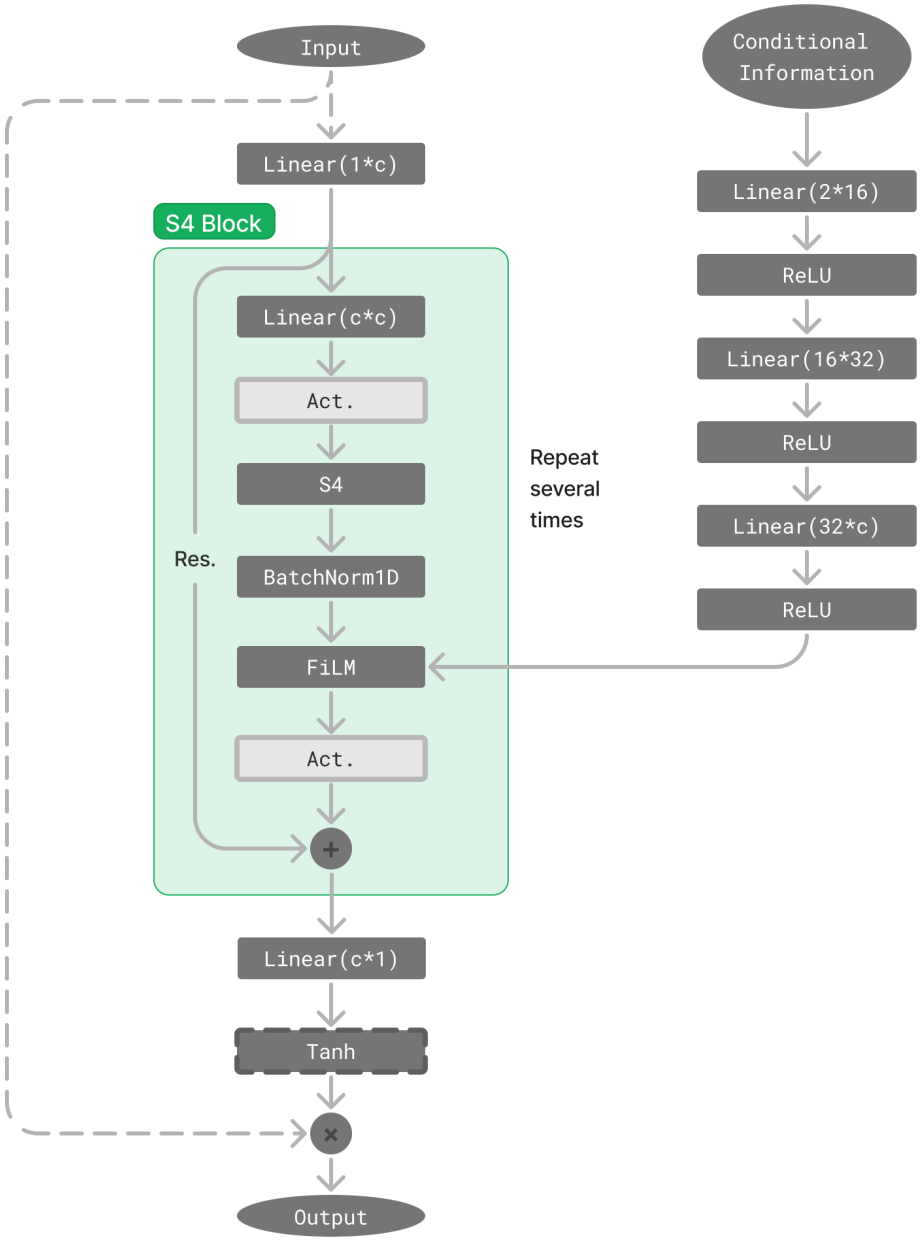


Figure 3.2: The Conditional Model

We combine three losses, the error-to-signal ratio (ESR) loss plus the DC loss [36] and the multi-resolution STFT loss [37], to train the model. All losses are weighted equally. When applying the ESR loss, a first-order high-pass filter is applied to both the synthesized and ground-truth signal, with the transfer function

$$H(z) = 1 - 0.85z^{-1}$$

The coefficient, 0.85, is determined by the sample rate 44.1 kHz. The cutoff frequency is 5380.2 Hz

For the training multi-resolution STFT loss, the following STFT calculation configurations are used:

- FFT size 1024, hop size 120, window length 600
- FFT size 2048, hop size 240, window length 1200
- FFT size 512, hop size 50, window length 240

All spectrograms are calculated using the Hanning window. Windowed signals are padded with zero at the end to fit with the FFT size if the FFT size is greater.

During validation, we tested five losses, four of them are time-domain losses, and the other is frequency-domain losses. They are mean-average error (MAE), mean-squared error (MSE), ESR loss, DC loss, and the multi-resolution STFT loss. The validation multi-resolution STFT loss is the same as the training multi-resolution STFT loss.

The following hyper-parameters were altered during the experiment:

1. Non-linear activation function (`Act.` in both models).
   The choice of non-linear activation function could be `sigmoid`, `Tanh`, ReLU, GELU, and PReLU (parameterized ReLU).
2. Number of S4 blocks.
3. S4 IIR filter order.
4. Number of audio channels (c in both models' linear layer.)
5. Taking hyperbolic tanh layer at the end or not.
6. Taking multiplication or not (turn the model to a side-chain model).

The reference hyper-parameter configuration uses GELU as the non-linear activation function, with four S4 blocks, forth order IIR filter, 32 audio channels, having the multiplication operation, and without the hyperbolic tangent function at the end. The ablation study was carried out in the following manner:

1. Altering the non-linear activation functions:
   using `Tanh`, `sigmoid`, ReLU, and PReLU.
2. Altering the number of S4 blocks to 2 and 6.
3. Altering the filter order and audio channels to the following configurations:
   (8, 32), (16, 32), (4, 16), (8, 16), and (16, 16).
4. Altering the side-chain and the final hyperbolic tangent layer.

3.3.2 Conditional Model Experiment

To train the conditional model, the entire SignalTrain dataset is used. Phase inversion with a probability of 0.5 is also applied as a data augmentation technique. Except we change the batch size to 64 for more variety per batch, the learning rate and its scheduler, optimizer, training precision, and epoch number are the same. We use the same training loss function and validation method, except that only 3 s audio is fed into the model due to memory constraints.

Following the completed fixed-parameter model ablation study, the conditional model inherits the training procedure. The following hyper-parameters are taken from the fixed-parameter ablation study. Detailed analysis will be presented in Chapter 4.

- Non-linear activation function: PReLU.
- S4 blocks: 4.
- S4 IIR filter order: 4.
- Audio channels: 32.

3.4 Objective Evaluation Procedure

For the fixed-parameter model, only those five validation losses are compared internally as the goal is to search for better hyper-parameters.

All five validation losses are tested, with the testing audio sliced to 1.0 s, 1.5 s, 4.0 s, and 10.0 s. Various lengths are selected to evaluate the model's generalizability and to compare the current work with previous work.

Audio waveform differences are also evaluated by subtracting the ground truth audio waveform from the output audio waveform. Waveform difference can tell whether the audio waveform has been appropriately

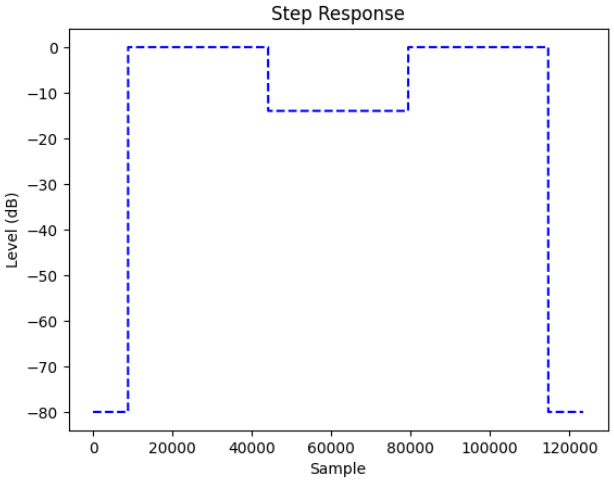


Figure 3.3: Step Response used for Evaluation, with the lowest possible level to be -80 dB

constructed. For some testing audio data that exhibits interesting results, waveform difference may be helpful.

The model step-response performance is also evaluated, aiming to evaluate the output signal envelope's level and stability. A pre-defined direct current (DC) step signal is fed to the model, with related control parameters existing in the SignalTrain dataset or not existing. This means that evaluations on peak reduction value not in the multiple of 5 are also provided to the trained model. The signal has five stages, with the beginning and the end having an amplitude of 0, the middle having an amplitude of 0.8, and the other having an amplitude of 1.0. This evaluates the model's performance when the signal is 0, at its max, or somewhere intermediate, with the previous signal being 0 at its max or somewhere intermediate. Figure 3.3 demonstrated the step response signal in decibels.

Each S4 layer's frequency response is also evaluated per channel. Internally, S4's state-space model is implemented in complex numbers. Directly inferring its frequency using the IIR information could be challenging. We first obtain a finite impulse response in the length of 44 100. Then, using the generated finite impulse response, which is all real, we calculate its frequency response with both magnitude and phase curves.

Chapter 4

Results and Analysis

This chapter will cover my experiment results and analysis for the fixed-parameter and conditional models. For each model, I will present some definitive findings and non-definitive findings along with objective evaluation data. Some cases are worth spending extra time discussing.

4.1 Fixed-parameter Model Results and Analysis

The fixed-parameter model is proposed for hyper-parameter search. In this section, all plots denote validation losses, using five single validation losses mentioned in Chapter 3 and the combined loss used for training, ESR+DC+Multi-STFT, throughout training epochs 30 to 70.

In my experiment procedure, the first element being experimented with is the non-linear activation function being used in S4 blocks. Figure 4.1 presents models with five non-linear activation functions: `GELU`, `tanh`, `sigmoid`, `ReLU`, and `PReLU`.

- `2023-5-8-13-17-6`'s model uses `GELU`.
- `2023-5-8-17-15-16`'s model uses `tanh`.
- `2023-5-8-17-52-11`'s model uses `sigmoid`.
- `2023-5-8-18-29-17`'s model uses ReLU.
- `2023-5-8-19-5-55`'s model uses `PReLU`.

We can see that except for DC loss, all other five validation loss curves have clear distinctions, with the top two curves, the orange one being `tanh` and the teal one being `sigmoid`, being on the top, and all the others being at the bottom. It is worth noting that we found DC loss tends to be very unstable during both

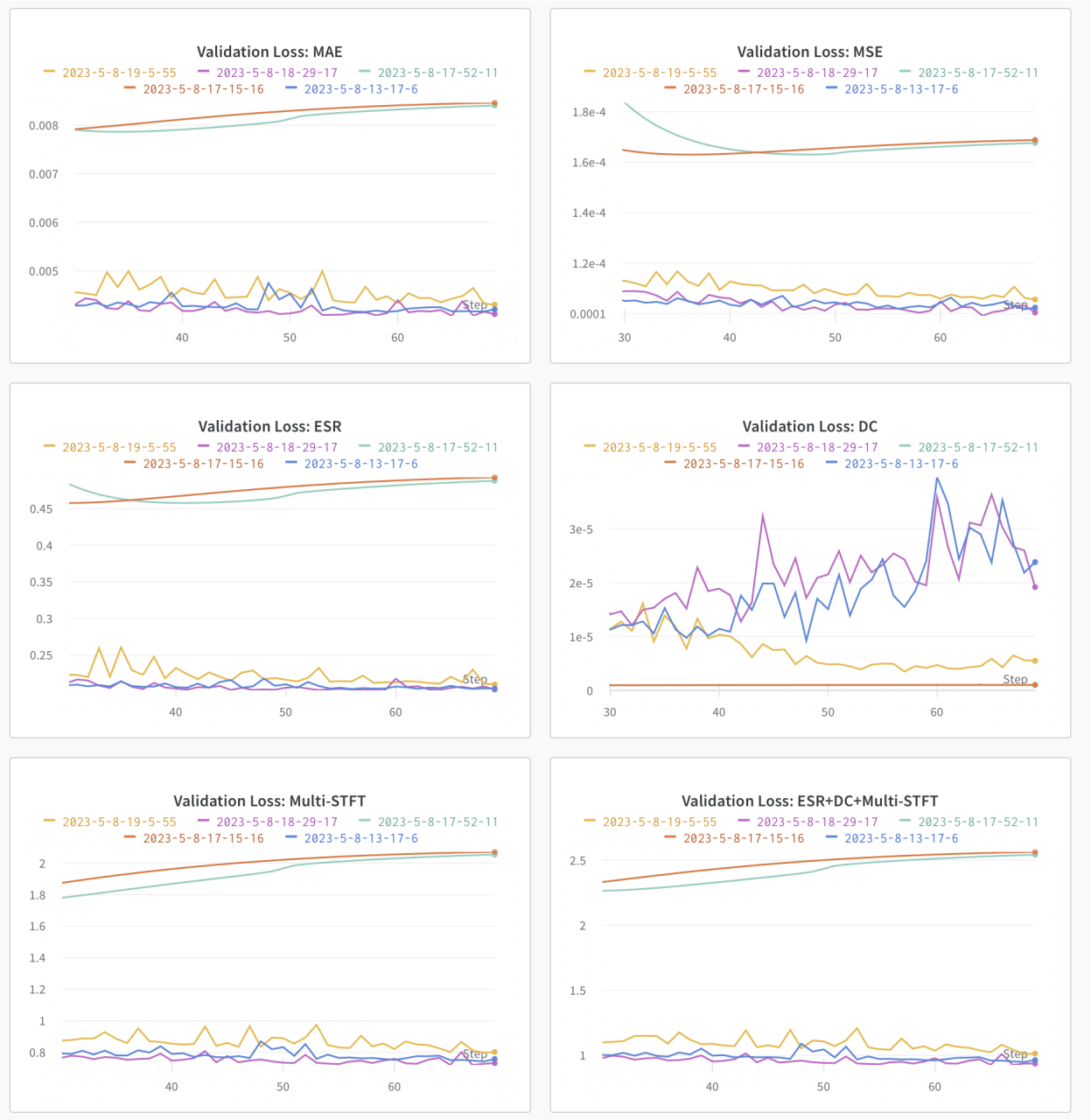


Figure 4.1: Validation loss curves among different models using different non-linear activation functions. `2023-5-8-13-17-6`'s model uses `GELU`. `2023-5-8-17-15-16`'s model uses `tanh`. `2023-5-8-17-52-11`'s model uses `sigmoid`. `2023-5-8-18-29-17`'s model uses ReLU. `2023-5-8-19-5-55`'s model uses `PReLU`. Models using `tanh` or `sigmoid` have significantly higher validation loss curves, showing that the linear unit function family is superior to normalizing non-linear activation functions.

the training and validation. Conclusions drawn from DC can often be ineffective. For most analyses based on loss functions, we choose not to take the results exhibited from DC loss.

The result shows that using a non-linear activation function like ReLU, GELU, or PReLU could yield better results. A possible reason why linear unit functions work better could be that normalizing the intermediate vectors to ± 1 or from 0 to 1 is overly restrictive, as the variable gain exhibited from a DRC could be greater than 1. In the future conditional model training, we chose to use PReLU as our non-linear activation function to align our work with Steinmetz and Reiss [1].

The second discussion is about how many S4 blocks shall we need to yield the most optimal result. Figure 4.1 presents models with 2, 4, or 6 S4 blocks.

- `2023-5-8-13-17-6`'s model uses 4 S4 blocks.
- `2023-5-9-1-4-1`'s model uses 2 S4 blocks.
- `2023-5-9-1-52-1`'s model uses 6 S4 blocks.

While for MSE, the model with two S4 blocks performs slightly worse than the others, all other losses are intertwined and not showing any distinctive compartmentalization. While more S4 blocks may correlate to better performance, more S4 blocks mean more layers and more model parameters, decreasing its efficiency and even the real-time capability to process the signal. Since the model will have more We chose to use four S4 blocks in the conditional model to balance the model performance and the space complexity.

The third discussion is on the number of audio channels and the filter order for each S4 layer.

- `2023-5-9-2-55-1`'s model has 16 audio channels and an order of 4 in the S4 IIR filter.
- `2023-5-9-3-26-44`'s model has 16 audio channels and an order of 8 in the S4 IIR filter.
- `2023-5-9-3-57-14`'s model has 16 audio channels and an order of 16 in the S4 IIR filter.
- `2023-5-9-4-29-8`'s model has 32 audio channels and an order of 4 in the S4 IIR filter.
- `2023-5-9-5-6-58`'s model has 32 audio channels and an order of 8 in the S4 IIR filter.
- `2023-5-9-5-45-23`'s model has 32 audio channels and an order of 16 in the S4 IIR filter.

Figure 4.3 shows those validation curves among the same 16 audio channels but with different filter orders of 4, 8, 16, and figure 4.4 has 32 audio channels and all the others are the same. While all three curves in Figure 4.4 are very intertwined and cannot see the exact difference, models with 16 audio channels can still see the marginal difference, with the red curve, the one with the least filter order, to be on the top in

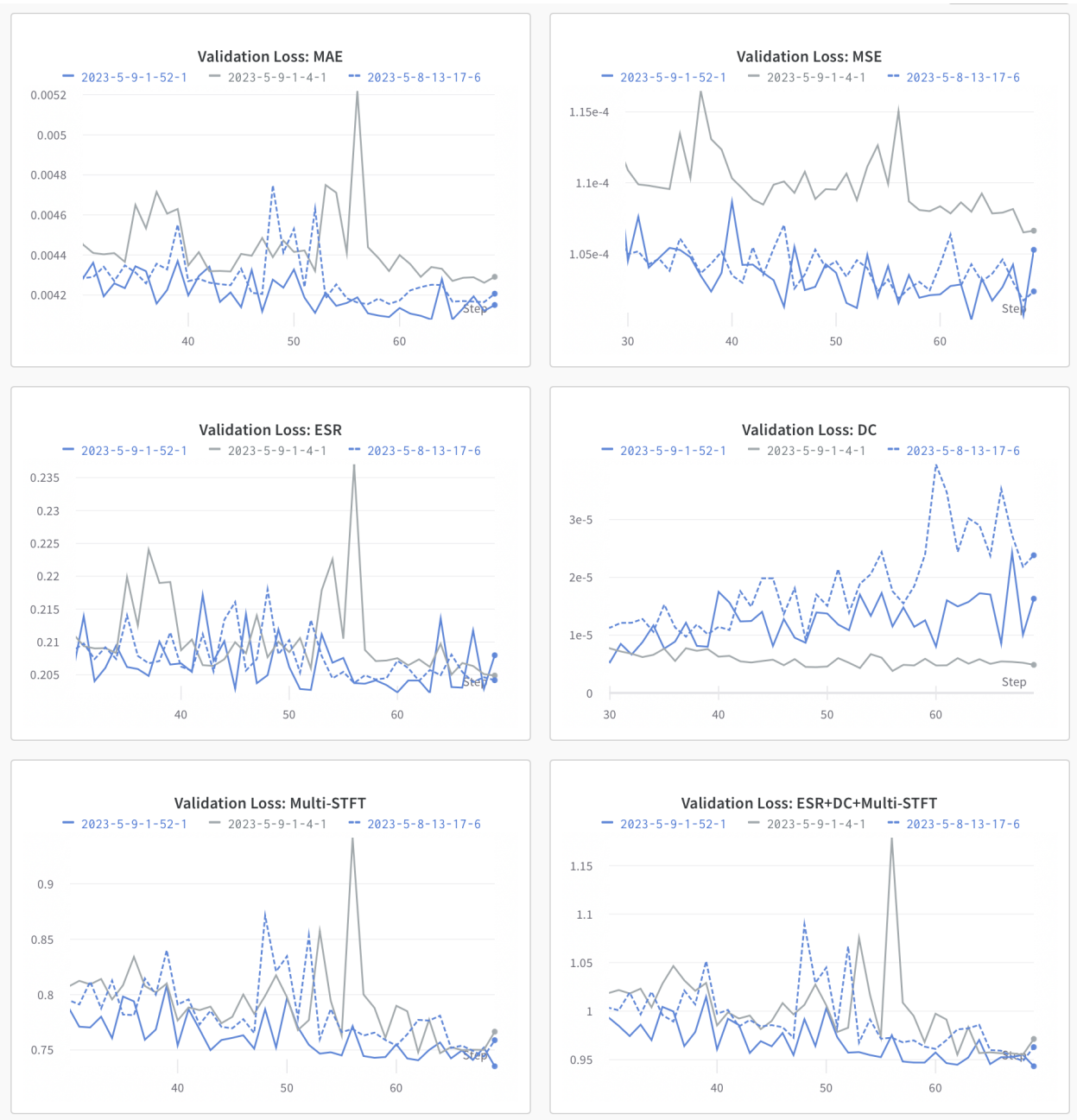


Figure 4.2: Validation loss curves among different models using different numbers of S4 blocks. `2023-5-8-13-17-6`'s model uses four S4 blocks. `2023-5-9-1-4-1`'s model uses two S4 blocks. `2023-5-9-1-52-1`'s model uses six S4 blocks. For MSE, the model with two S4 blocks performs slightly worse than the others. Other validation losses do not exhibit clear distinction.

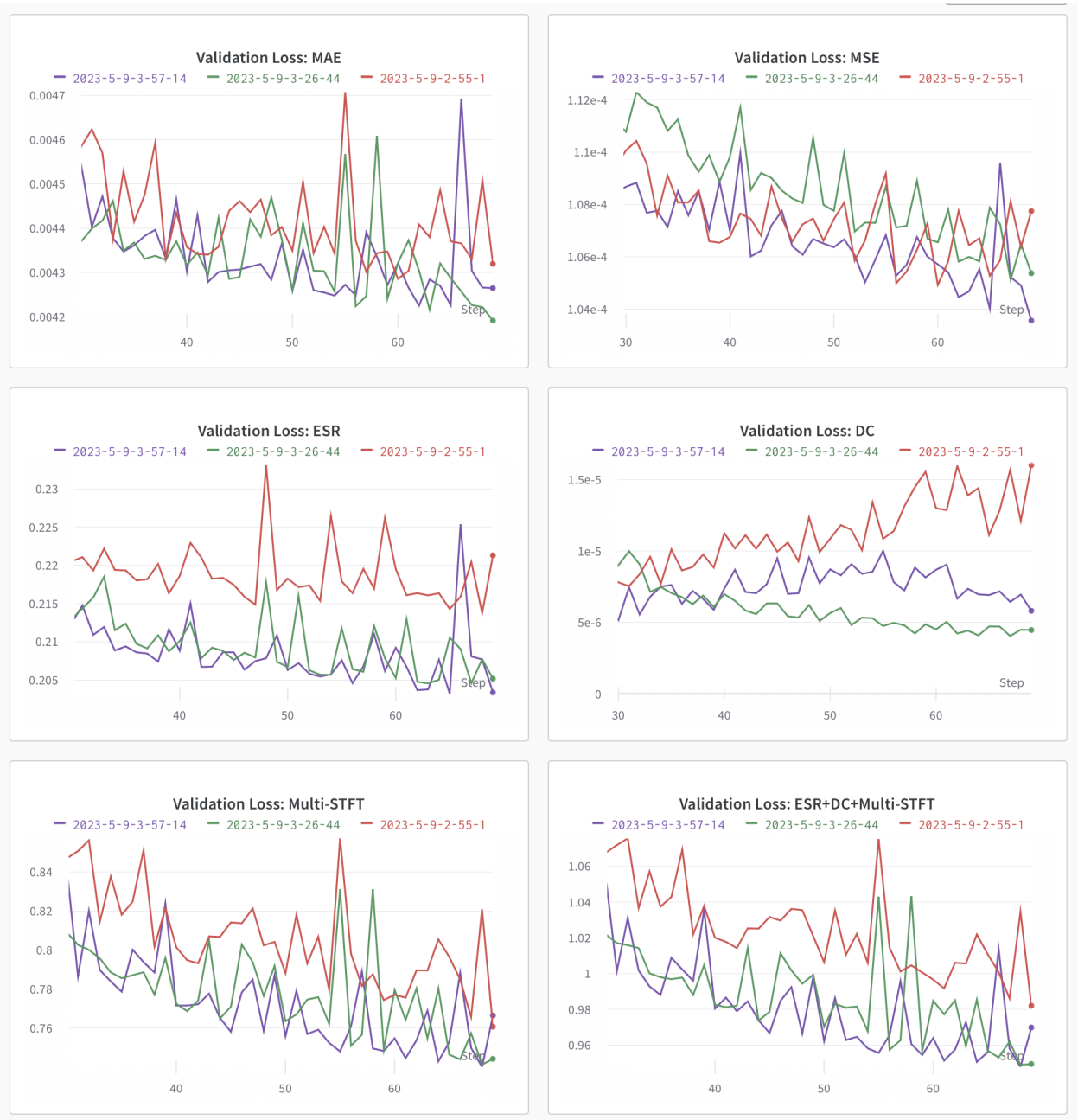


Figure 4.3: Validation loss curves among models with 16 audio channels but different S4 IIR filter order. 2023-5-9-2-55-1's model has 16 audio channels and an order of 4 in the S4 IIR filter. 2023-5-9-3-26-44's model has 16 audio channels and an order of 8 in the S4 IIR filter. 2023-5-9-3-57-14's model has 16 audio channels and an order of 16 in the S4 IIR filter.

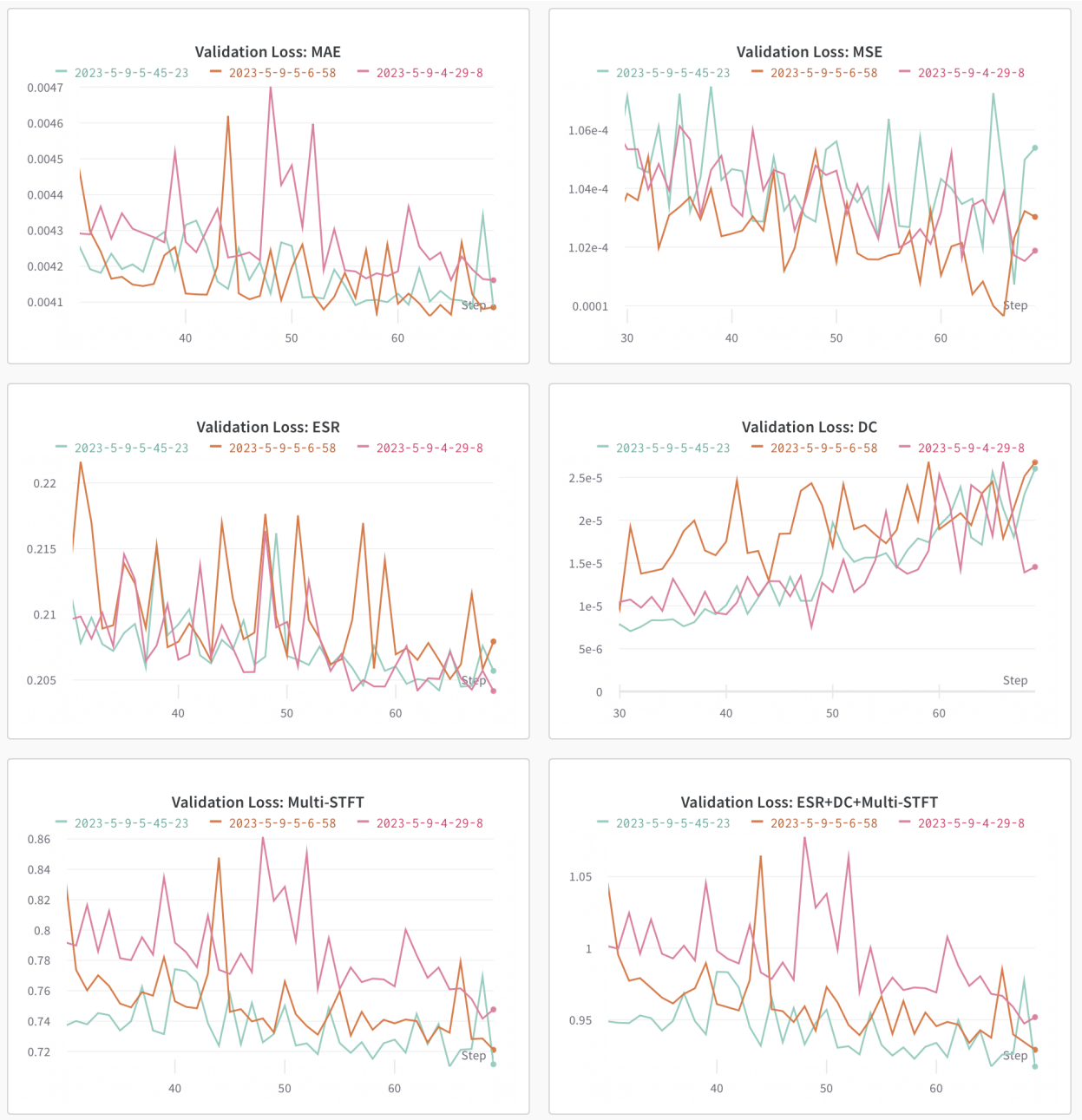


Figure 4.4: Validation loss curves among models with 16 audio channels but different S4 IIR filter order. 2023-5-9-4-29-8's model has 32 audio channels and an order of 4 in the S4 IIR filter. 2023-5-9-5-6-58's model has 32 audio channels and an order of 8 in the S4 IIR filter. 2023-5-9-5-45-23's model has 32 audio channels and an order of 16 in the S4 IIR filter.

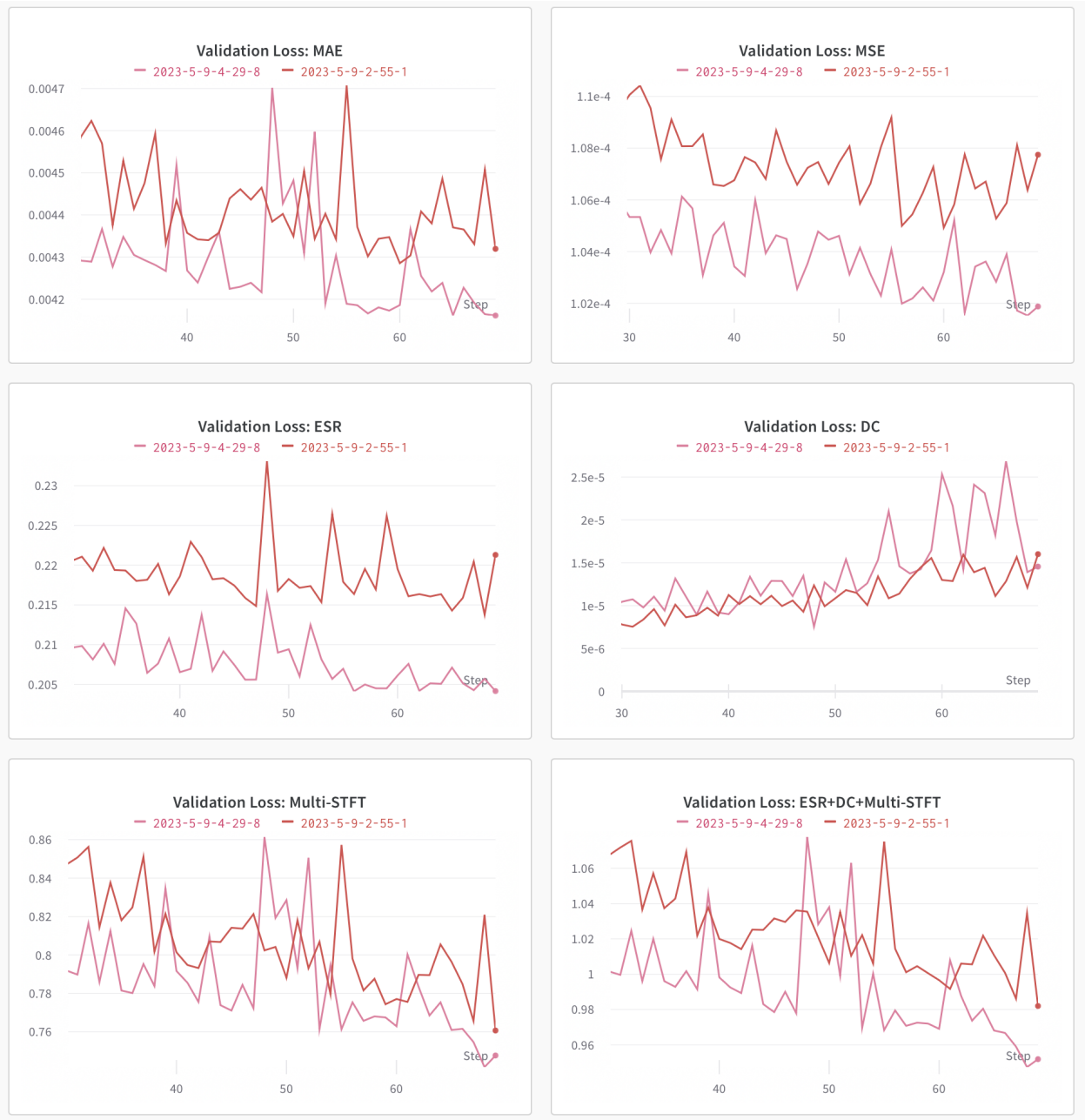


Figure 4.5: Validation loss curves among different models with the same IIR filter order using different audio channels. `2023-5-9-2-55-1`'s model has 16 audio channels and an order of 4 in the S4 IIR filter. `2023-5-9-4-29-8`'s model has 32 audio channels and an order of 4 in the S4 IIR filter.

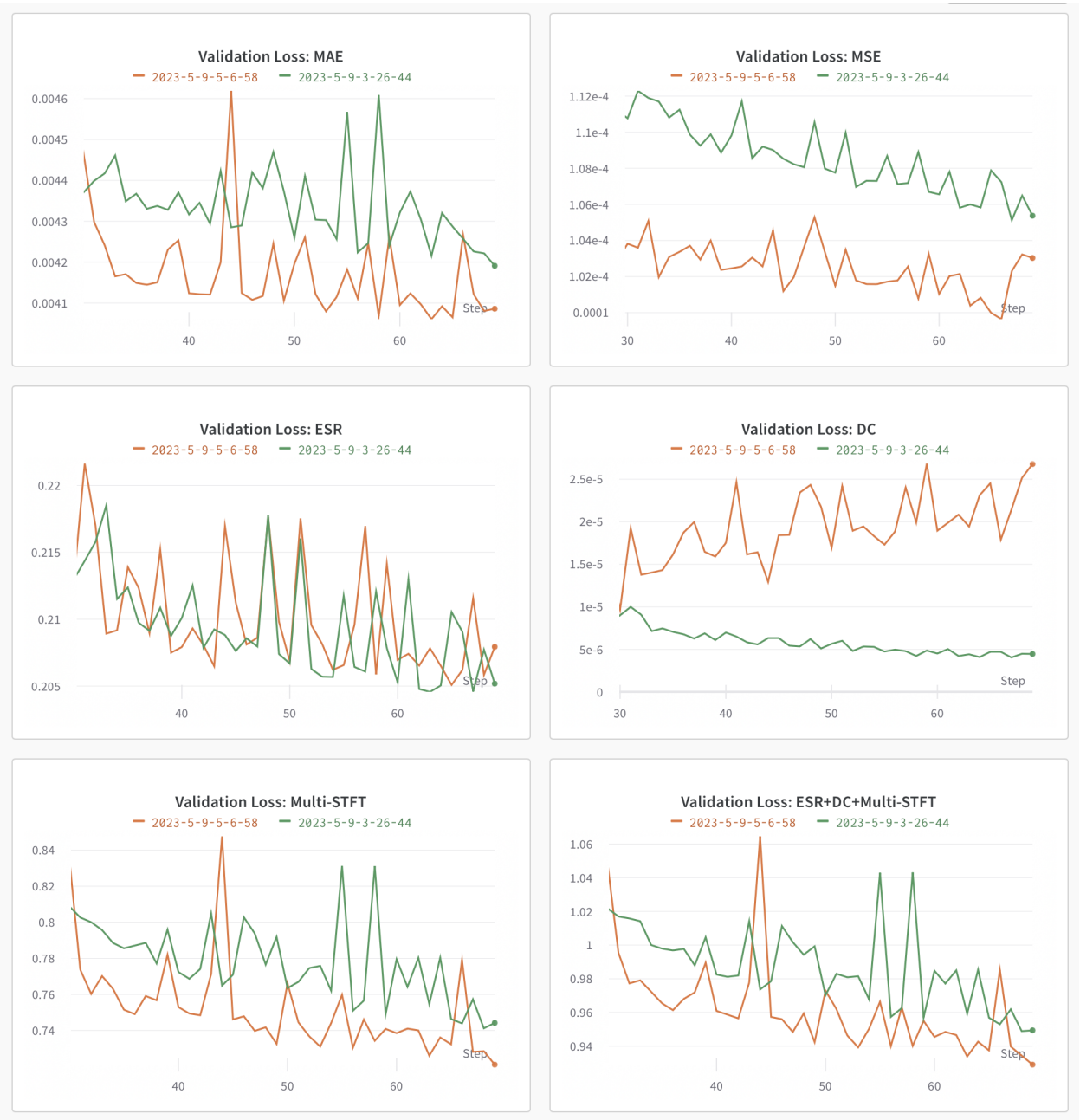


Figure 4.6: Validation loss curves among different models with the same IIR filter order using different audio channels. `2023-5-9-3-26-44`'s model has 16 audio channels and an order of 8 in the S4 IIR filter. `2023-5-9-5-6-58`'s model has 32 audio channels and an order of 8 in the S4 IIR filter.

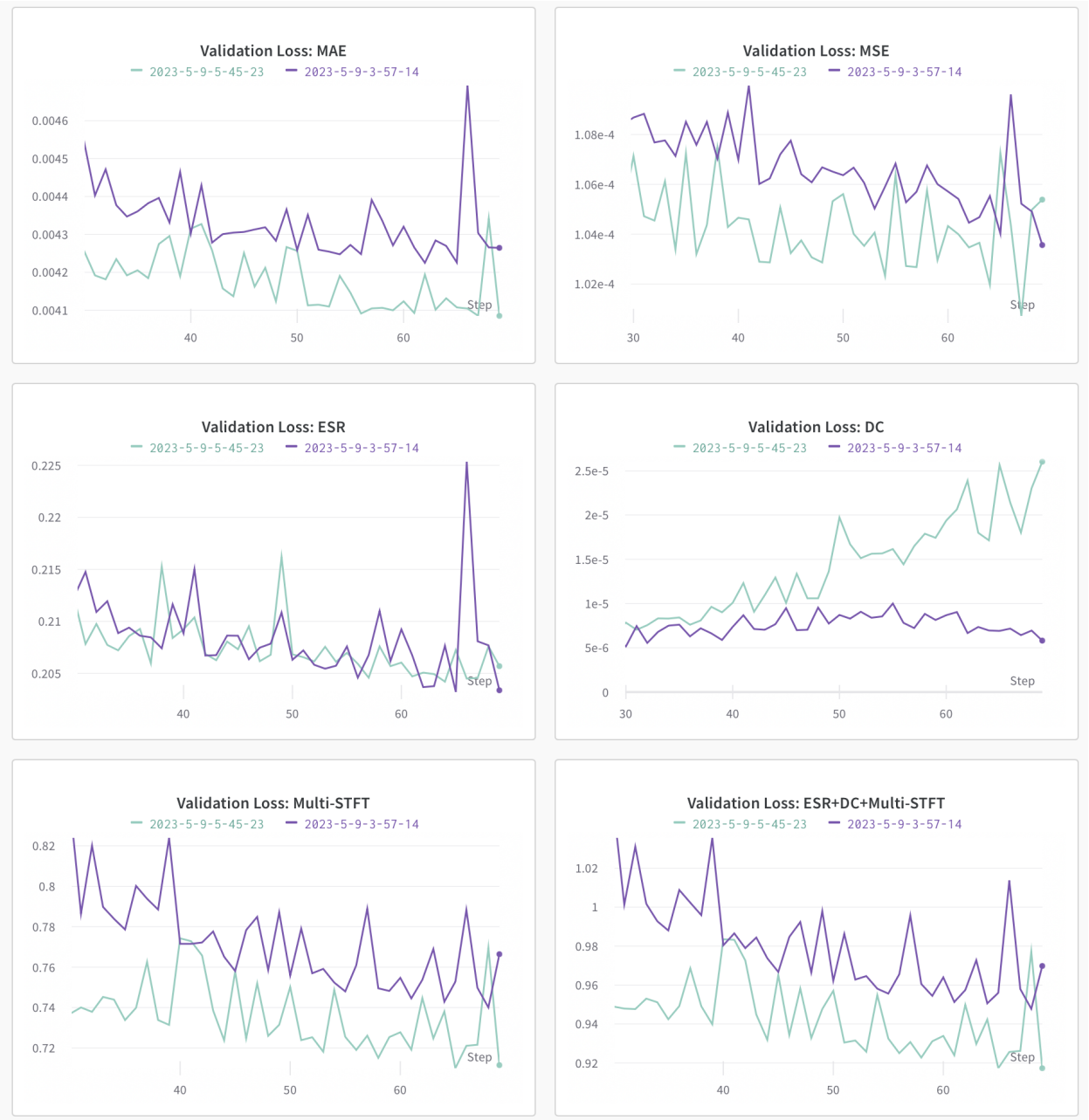


Figure 4.7: Validation loss curves among different models with the same IIR filter order using different audio channels. `2023-5-9-3-57-14`'s model has 16 audio channels and an order of 16 in the S4 IIR filter. `2023-5-9-5-45-23`'s model has 32 audio channels and an order of 16 in the S4 IIR filter.

ESR loss, Multi-STFT loss, and STFT loss. However, the pattern does not hold in MSE loss, where all three curves are interweaving.

Whereas in Figures 4.5, 4.6, and 4.7, with the same S4 IIR filter order, distinctions are clear with clear compartmentalization, especially in MSE loss, MAE loss, and STFT loss. It can be seen that under the same S4 IIR filter order, having more audio channels can bring better performance in almost all tested cases. There are exceptions in DC loss curves, where having more audio channels may bring worse performance. However, as mentioned before, DC loss tends to be unstable during training. The performance on DC may not truly reflect the model performance, and we choose to neglect its result.

The conclusion is that having more audio channels is more paramount than having more S4 IIR filter orders. It is not genuinely definitive which hyper-parameter configuration is better than the other, as our comparison is relative and the absolute gain is solely marginal. We finally decided to keep the initial model configuration that there are 32 audio channels and an S4 IIR filter order of 4.

For the final analysis, whether to have the multiplication operation or a hyperbolic tangent function at the end have experimented. Figure 4.8 presents models with or without the final multiplication operation and the final `tanh` layer.

- `2023-5-8-13-17-6`'s model has the final multiplication operation without `tanh`.
- `2023-5-9-6-25-13`'s model has no final multiplication operation and without `tanh`.
- `2023-5-9-7-2-47`'s model has the final multiplication operation with `tanh`.
- `2023-5-9-7-40-52`'s model has no final multiplication operation with `tanh`.

Models without the final multiplication operation perform worse than models with the final multiplication operation. However, the final `tanh` layer does not clearly impact performance. These variations, introducing the final multiplication operation and the final `tanh` layer, represent fundamental changes to the model structure, so we do not assume that their performance in the fixed-parameter model is identical to the conditional model. Therefore, the ablation studies for the conditional model presented in the next section include these two variations. With other parameters and variations, we assume that findings from this fixed-parameter model apply to the conditional model, and we will use only the best-performing parameter values for further evaluation. This simplification is necessary to limit the computation required for additional training.

To summarize, the ablation study conducted for the fixed-parameter model shows that having more audio channels is more important than having higher S4 IIR filter order, using a linear unit function is better than `sigmoid` and `tanh` for the model's non-linear activation function, and using four S4 blocks is

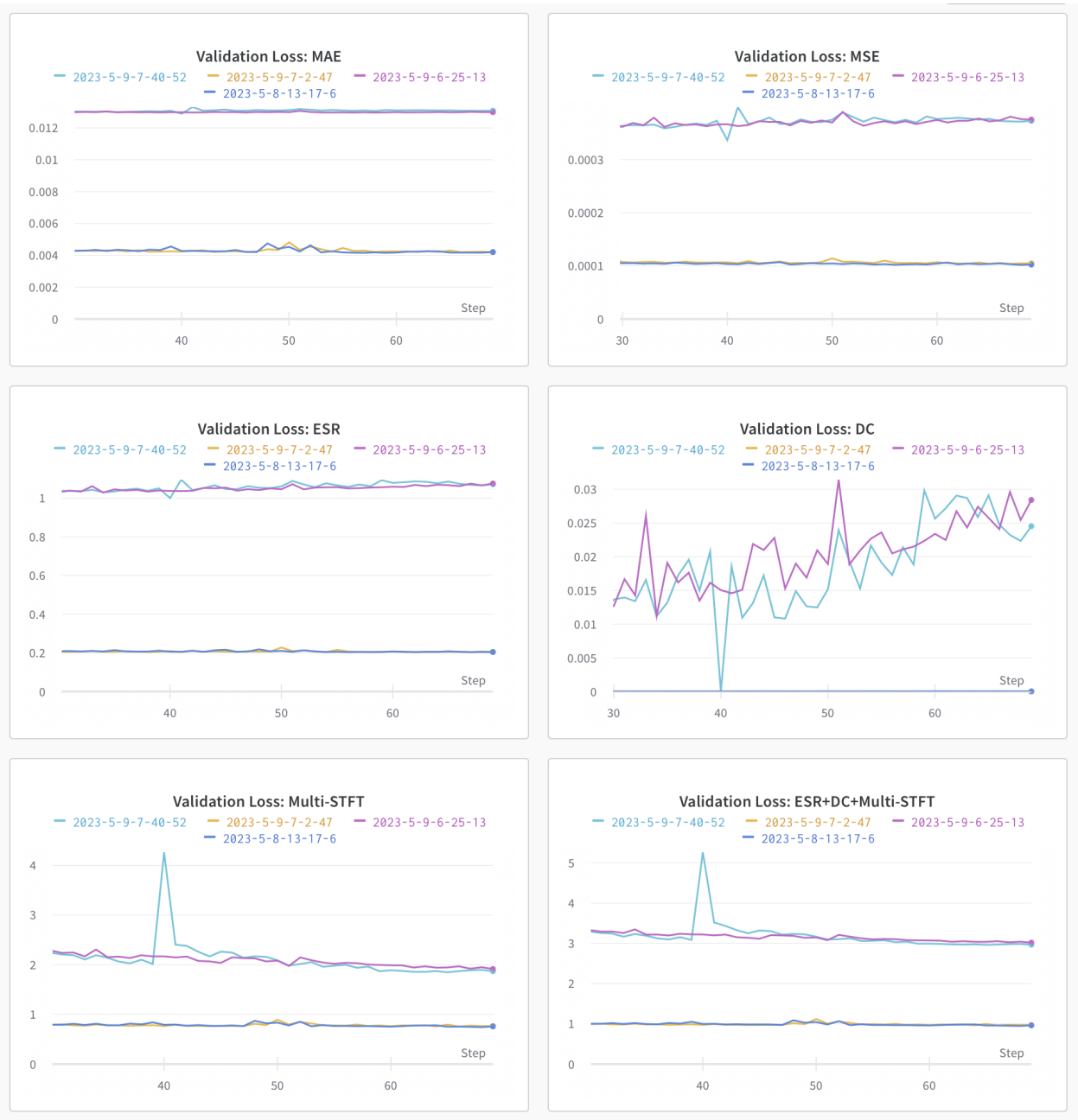


Figure 4.8: Validation loss curves among different models with the final multiplication operation or not or with the hyperbolic tangent function at last or not. `2023-5-8-13-17-6`'s model has the final multiplication operation without `tanh`. `2023-5-9-6-25-13`'s model has no final multiplication operation and without `tanh`. `2023-5-9-7-2-47`'s model has the final multiplication operation with `tanh`. `2023-5-9-7-40-52`'s model has no final multiplication operation with `tanh`.

Table 4.1: Test losses among all conditional models among the SignalTrain testing dataset, along with the model parameter number and the testing sample length. Results from Steinmetz and Reiss [1] are included at the end of the table.

Model Type	Model Parameter	Sample Length	DC	ESR	MAE	MSE	Multi-STFT
Multiplication without tanh	16,857	1.500	4.60E-04	1.18E-01	1.24E-02	4.21E-04	6.71E-01
Multiplication with tanh	16,857	1.500	1.21E-03	1.51E-01	1.97E-02	1.11E-03	6.83E-01
Single-chain without tanh	16,857	1.500	5.81E-03	1.12E-01	9.47E-03	2.18E-04	5.74E-01
Single-chain with tanh	16,857	1.500	2.36E-03	9.34E-02	8.55E-03	1.86E-04	5.43E-01
TCN-324-N [38]	162,000	1.598			1.70E-02		5.87E-01
TCN-100-N [1]	26,000	1.598			1.58E-02		7.68E-01
TCN-300-N [1]	51,000	1.598			**7.66E-03**		6.00E-01
TCN-1000-N [1]	33,000	1.598			1.20E-01		7.36E-01
TCN-100-C [1]	26,000	1.598			1.92E-02		7.70E-01
TCN-300-C [1]	51,000	1.598			1.44E-02		6.03E-01
TCN-1000-C [1]	33,000	1.598			1.17E-01		6.92E-01
LSTM-32 [1]	5,000	1.598			1.10E-01		5.51E-01

sufficient in performance and results in reasonable space complexity. We used these findings to create the conditional model, which is studied in the next section For the upcoming conditional model, whether to introduce multiplication operation and `tanh` at the end shall be discussed.

4.2 Conditional Model Results and Analysis

4.2.1 Objective Loss

Table 4.1 presented all six conditional models' testing loss among the SignalTrain testing dataset, along with model parameter numbers and related sample length in second. When selecting the model checkpoint to calculate the test loss, the model with the best overall validation loss was selected.

Among all loss values, the model without the final multiplication operation but with the final `tanh` layer performs the best. Except for the DC loss, this model yields the lowest losses among all testing criteria. Using the almost-the-same sample length in seconds for testing, the model's multi-STFT loss outperforms all models reported by Steinmetz and Reiss. Although the MAE loss is not as close as the best model `TCN-300-N`, our model uses way fewer model parameters and is causal, whereas `TCN-300-N` is a non-causal model (The suffix N means non-causal, and C means causal).

Internally, we found that for the same model, having a longer sample length during testing can usually guarantee better performance. Interestingly, for conditional models, having the final multiplication operations decreases the performance, with loss values significantly higher than single-chain models (without the multiplication operation). This goes contrary to the experiment result from those fixed-parameter models.

In general, we have found that having the final `tanh` layer is beneficial for single-chain models, whereas for models with a final multiplication operation, it might be better to not introduce any `tanh` layer at the end, although there are exceptions, which will be discussed in the next section.

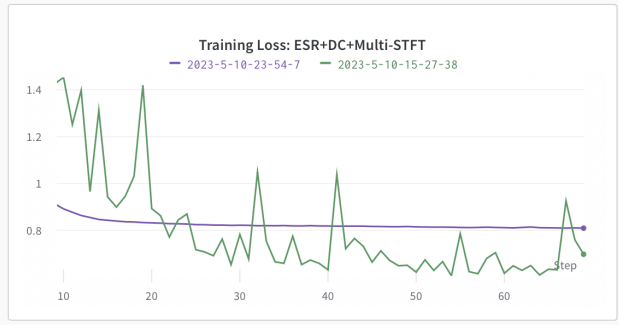


Figure 4.9: Training loss curve for models with or without the final multiplication operation. The purple curve, `2023-5-10-7-1-39` is the one with it, and the green curve, `2023-5-9-22-34-47`, is the one without it. It can be observed that the red curve is smoother than the blue curve.

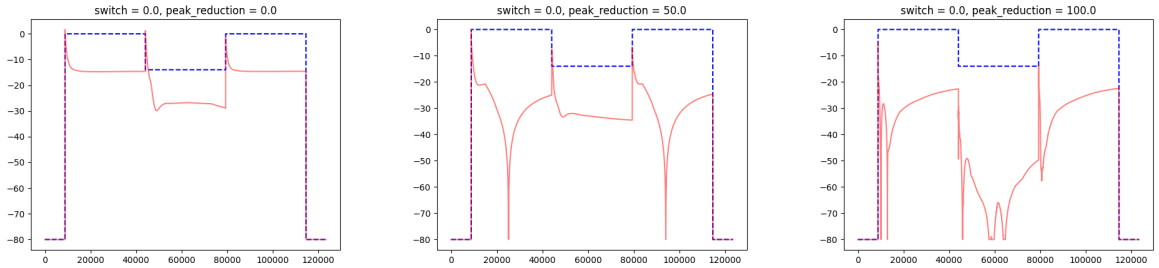
4.2.2 Discussion on Final Multiplication Operation

The introduction of the final multiplication operation is influenced by Wright et al. [23], which cited the DRC implementation structure outlined by Giannoulis et al. [21], as mentioned in Chapter 2. The general understanding is that by introducing the multiplication operation, the model could resemble an actual DRC more closely, letting all the S4 blocks work as the side-chain extracting loudness. With the introduction of FiLM layers, however, it seems like S4 blocks lose their capability to model DRCs properly. A possible reason could be that FiLM is not a proper method to apply conditional parameters to alter the model behavior, which may be done in future work.

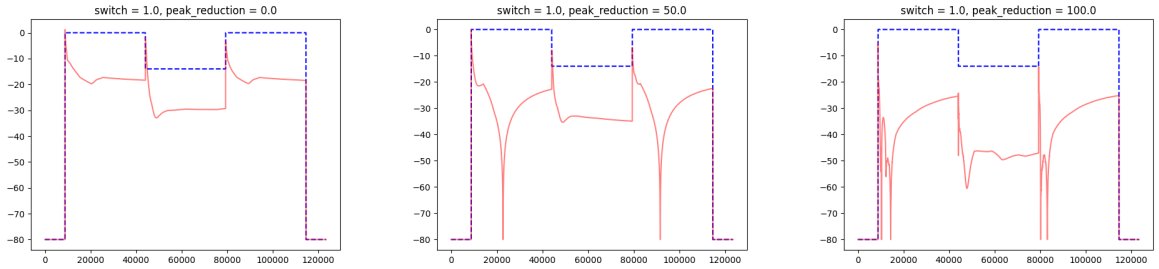
There are, however, certain benefits of introducing a side-chain to the model. One benefit observed from training is that the training loss on the model with multiplication operation had been decreasing more stably than the model without it, under the same condition that both model has the final `tanh` layer. It can be observed that the model with the multiplication operation has a smoother training loss curve than the one without it. Interestingly, even though the model without the final multiplication operation, testing loss is lower than the one with it, to some epochs, the model without it is performing worse than the one with the final multiplication operation. This shows that to find the best-performing single-chain model, Manual selection must be done, and the training process shall be monitored.

- `2023-5-10-15-27-38` is the model without the final multiplication operation.
- `2023-5-10-23-54-7` is the model with the final multiplication operation.

There are some auditory reasons why having the final multiplication operation could be a better choice, even though the model's loss is worse. The model step response has been tested among the model with the final multiplication operation but without the final `tanh` layer, shown in Figure 4.10, and the model without the final multiplication operation but with the final `tanh` layer, shown in Figure 4.11.

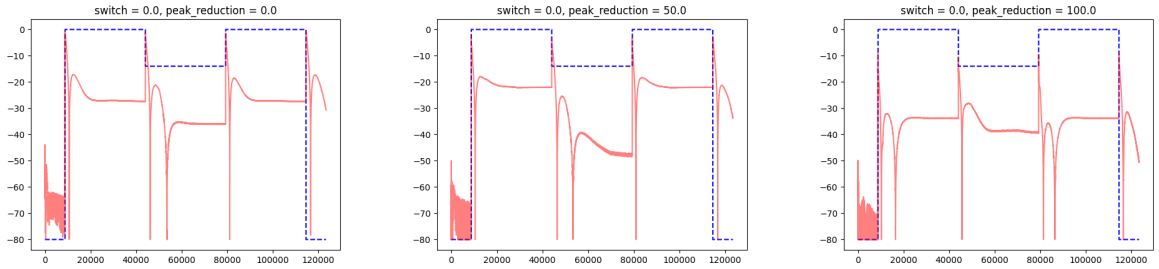


((a)) Compressing, Peak Reduction 0 ((b)) Compressing, Peak Reduction 50 ((c)) Compressing, Peak Reduction 100

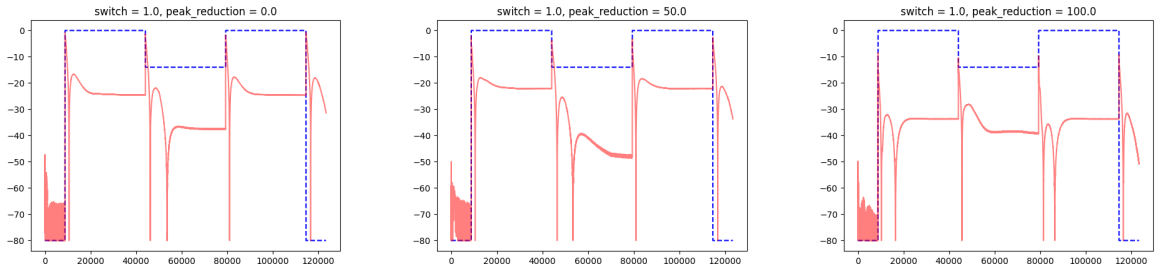


((d)) Limiting, Peak Reduction 0 ((e)) Limiting, Peak Reduction 50 ((f)) Limiting, Peak Reduction 100

Figure 4.10: Selected model step response with the final multiplication operation.



((a)) Compressing, Peak Reduction 0 ((b)) Compressing, Peak Reduction 50 ((c)) Compressing, Peak Reduction 100



((d)) Limiting, Peak Reduction 0 ((e)) Limiting, Peak Reduction 50 ((f)) Limiting, Peak Reduction 100

Figure 4.11: Selected model step response without the final multiplication operation.

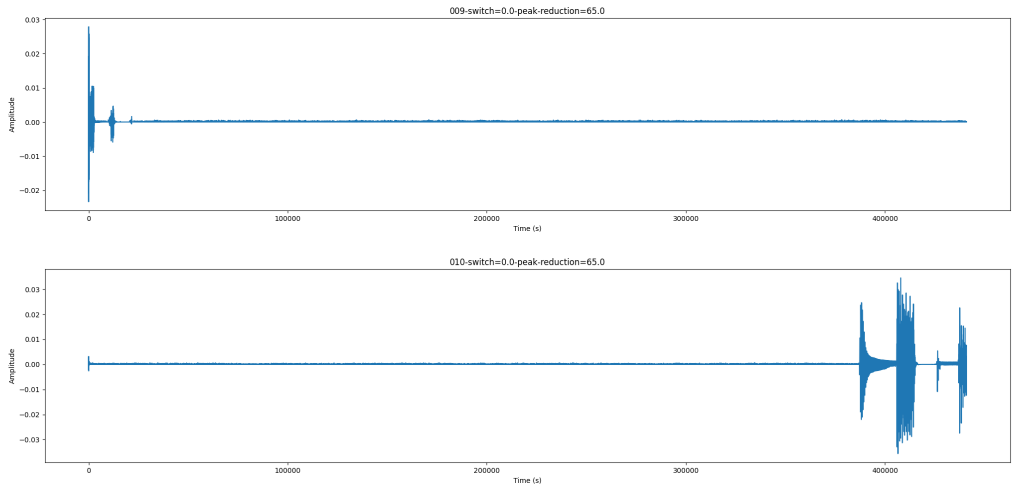


Figure 4.12: Waveform difference for two testing samples by the model with the final multiplication operation and the final `tanh` layer.

We can see that models with the final multiplication operation exhibit more stable envelope curves, with fewer sudden drops and an almost-perfect modeling when the signal level is -80 dB. Those better envelope response curves may reflect those models' better DRC loudness response detection, even though validation loss values are higher.

Even more interesting, two testing samples are almost perfectly modeled by the model with the final multiplication operation and the final `tanh` layer. Figure 4.12 demonstrates the waveform difference in amplitude. We can see that apart from some samples, which are not related to the perfectly modeled audio, the waveform difference is almost zero. Those two waveform's related original audio are string recordings with no silence.

Given that in Figure 4.10, models with the final multiplication operation do not alter audio data with loudness below the threshold, a possible explanation for those samples with an almost-zero difference is that the model with the final multiplication operation and the final `tanh` layer learned when not to apply compression. Future work might investigate whether this is the case.

4.2.3 Real-time Evaluation Analysis

As we previously mentioned, our models are not real-time. Our models require the input audio to have a finite length. Although it is possible to slice the input audio into fixed-length buffers, process each sliced audio buffer, and splice those resulting output buffers, the spliced output audio may exhibit clicks or sudden loudness changes. However, since S4 layers implement IIR systems in the state-space form, one could utilize those state-space matrices to process audio samples individually. If a real-time implementation

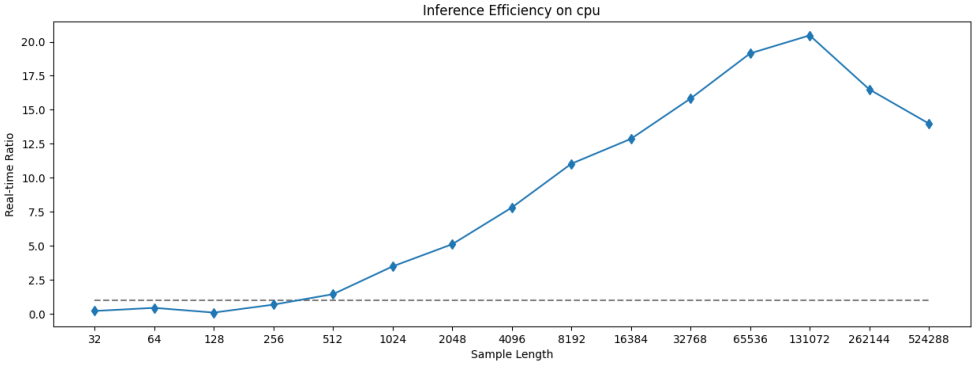


Figure 4.13: Real-time Ratio for Intel Xeon Platinum 8358

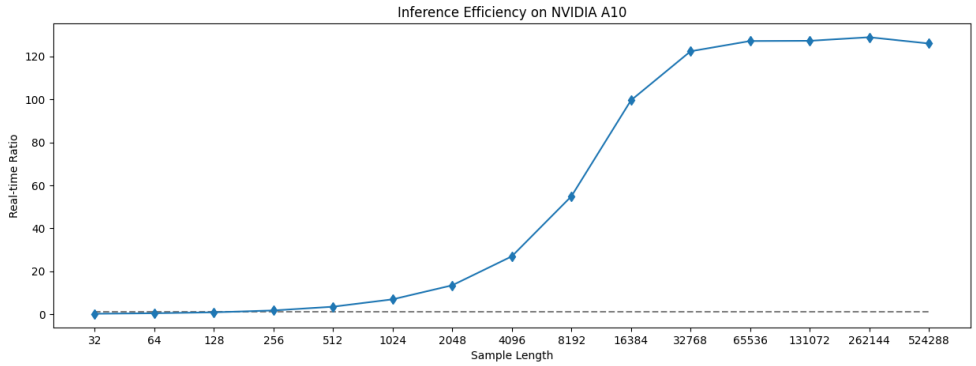


Figure 4.14: Real-time Ratio for Nvidia A10

of S4 layers is made, the model could process arbitrarily long audio data without requiring slicing input audio into pre-defined audio buffers.

We would like to analyze the inference efficiency in the non-real-time scenario first. We define the term "real-time ratio" as the audio buffer's actual playback time divided by model inference time. A ratio bigger than one means the model can process the audio buffer faster than its playback speed.

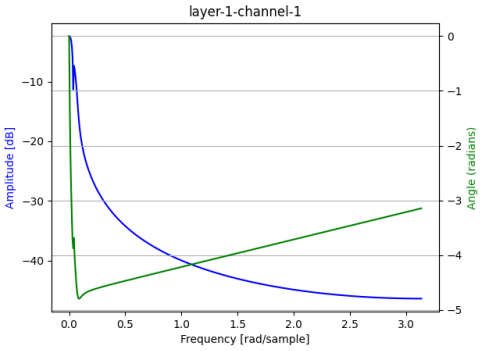
Figure 4.13 plots the model real-time ratio on the CPU chip, namely Intel Xeon Platinum 8358, and Figure 4.14 plots the model real-time ratio on the GPU chip, namely Nvidia A10. On the CPU chip, for sample lengths greater than 512, the model can do the audio processing in real-time, whereas on the GPU chip, the minimum length is 256. Since these models are batch-processing and not real-time, it is probably more interesting to consider the computation speed on sounds of one to a few seconds, where the model is most efficient. The CPU computes up to about 20 times faster than audio duration, and the GPU computes around 120 times faster (see Figures 4.13 and 4.14).

Though the real-time implementation of S4 layers is not made, we could estimate the model's real-time inference efficiency by counting the total number of floating-point operations (FLOPs) required to process

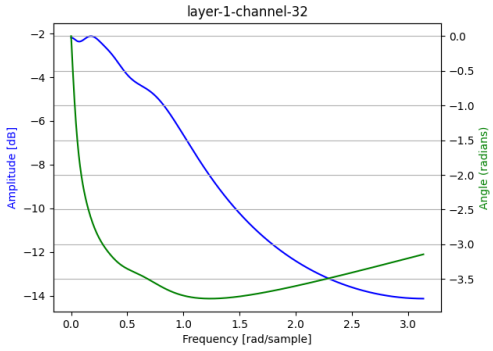
one single audio sample. We take the conditional model with the final multiplication operation and the final `tanh` layers as an example, as it has an extra floating-point number multiplication operation and the extra `tanh` calculation at the end. We use the same hyper-parameters as we used training the conditional model: using the PReLU activation function, 32 inner-audio channels, fourth-order S4 IIR filter order, and four S4 blocks. We assume that PReLU takes five FLOPs. For `tanh`, we assume that it takes twenty FLOPs. We can safely ignore the impact since conditional information is only processed once. The estimation is demonstrated as the following list giving estimated equivalent FLOPs for each layer:

- First linear layer to expand channels: 32
- In each S4 block ($\times 4$):
  - Linear layer: $32 \times 32 = 1024$
  - First PReLU layer: $5 \times 32 = 160$
  - S4 layer: $25 \times 32 = 160$ (fourth-order SSM)
  - PReLU layer: $5 \times 32 = 160$
  - Batchnorm1D layer: 32
  - FiLM: 32
  - Second PReLU Layer: $5 \times 32 = 160$
  - Residual connection: 32
- Final linear layer to contract channels: 32
- Final tanh layer: ≈ 20
- Final multiplication operation: 1

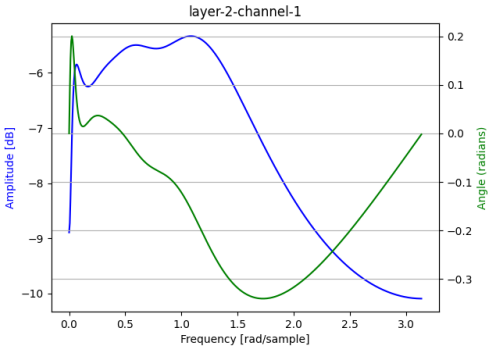
Approximately 10 000 FLOPs are required to process one audio sample. With the sampling rate 44.1 kHz, 441 000 000 FLOPs are required per each second. Considering a CPU core with 5 GFLOPs per second processing speed, it can process the audio around ten times faster than the actual playback speed. We believe the performance, while much slower than a typical DRC implementation, is feasible in an actual audio production scenario.



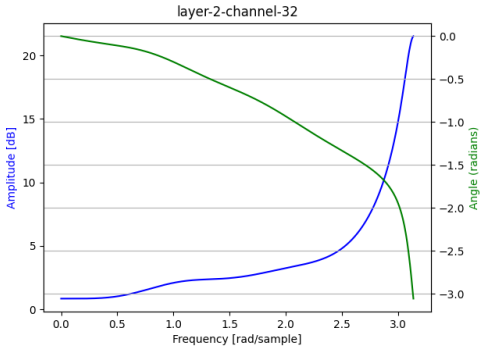
((a)) S4 Block 1 Channel 1



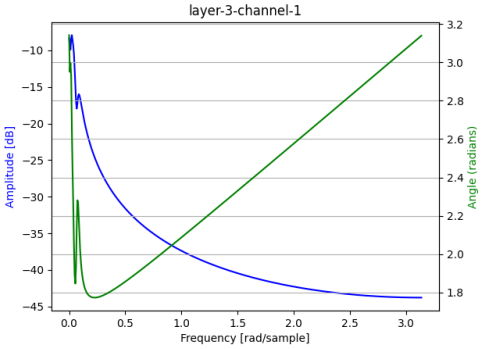
((b)) S4 Block 1 Channel 32



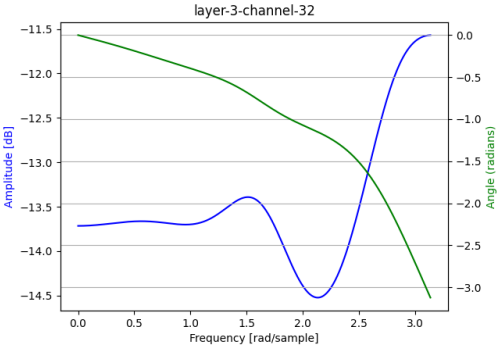
((c)) S4 Block 2 Channel 1



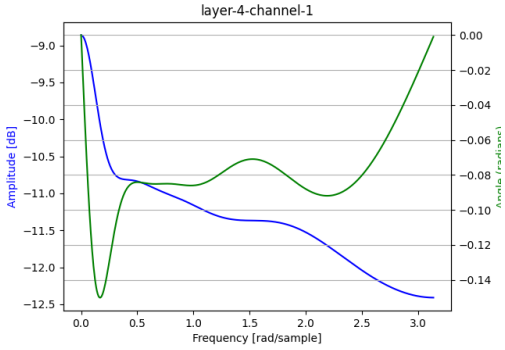
((d)) S4 Block 2 Channel 32



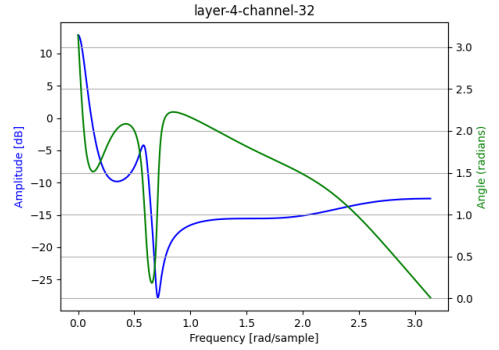
((e)) S4 Block 3 Channel 1



((f)) S4 Block 3 Channel 32



((g)) S4 Block 4 Channel 1



((h)) S4 Block 4 Channel 32

Figure 4.15: Selected S4 layer frequency response curve from the model with the final multiplication operation and final `tanh`.

4.2.4 Other Findings

The final section is about the S4 layer's frequency response, as demonstrated as an evaluation task in Chapter 3. Figure 4.15 demonstrated some selected S4 frequency response curves from the model with the final multiplication operation and final `tanh`.

Unfortunately, there aren't any direct patterns that can be observed from those frequency response curves. The magnitude curve does not show useful information and is not always a low-pass filter. So far, it is not totally possible to infer some interpretable information from those S4 layers to check some useful information from the learned DRC. Future work in model interpretability might need to dive deeper than simply inferring the frequency responses from S4 layers.

Chapter 5

Conclusions and Future Work

We presented a model using S4 layers to model an analog DRC, namely the Teletronix LA-2A optoelectonic compressor (LA-2A). Our model uses linear layers to mix channels, S4 layers to model sequence per channel, and FiLM layers to apply DRC control parameters as conditional information. The model can also feature the final multiplication operation to simulate the multiplier found in DRC hardware. In objective evaluation, our model exhibits the lowest multi-STFT loss compared to the LSTM model presented by Steinmetz and Reiss. Our model also exhibits a close MAE loss compared to Steinmetz and Reiss, with our model being causal and using fewer parameters. Our model can process audio clips faster than the actual audio playback speed with a relatively large audio buffer size. By converting S4 layers, which convolve their entire input in one step, to a real-time SSM or the transfer function form that can process one sample at a time, the model could process a continuous stream of samples, making a real-time implementation possible. It is speculated that the model could perform real-time inference ten times faster than the actual playback speed for audio with 44.1 kHz sampling rate using a single core of a CPU with 5 GFLOP/s.

We also experimented with introducing the final multiplication operation to the model. Although the objective loss is higher than the single-chain model, certain phenomena are worth discussing. The model step response envelope curve is more stable, certain audio clips in a certain scenario can be almost perfectly synthesized, and the training process is more predictable and stable. All of those phenomena may result in better audio quality, though without a subjective evaluation, the exact quality of the audio cannot be assessed.

Some future work might be conducted to evaluate the model further. First, a subjective evaluation should be conducted to evaluate the synthesized audio perceptual quality and the effectiveness of introducing the final multiplication operation to the model. Moreover, to fully evaluate the efficiency of S4, which is implemented as an IIR filter, a sample-to-sample implementation of S4 could be created along with

a real-time implementation, allowing integration of learned models into a real production environment. Finally, our evaluation of the model step response does not have a ground truth to be compared, given that we do not have access to the actual analog LA-2A compressor being used to produce the SignalTrain dataset. Modeling a software compressor allows complete access to internal data, such as the filter response, and the ability to automate the production of training data. Both could lead to a better understanding of the model behavior and optimization.

Bibliography

[1] C. J. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," *AES Europe Spring 2022 - 152nd Audio Engineering Society Convention 2022*, pp. 451–459, 2 2021. [Online]. Available: https://arxiv.org/abs/2102.06200v2

[2] J. O. Smith, "Physical modeling synthesis update," *Computer Music Journal*, vol. 20, no. 2, pp. 44–56, 1996.

[3] J. Pekonen and V. Välimäki, "The brief history of virtual analog synthesis," in *Proc. 6th Forum Acusticum. Aalborg, Denmark: European Acoustics Association*, 2011, pp. 461–466.

[4] K. J. Werner, "Virtual Analog Modeling of Audio Circuitry using Wave Digital Filters," 2016. [Online]. Available: http://purl.stanford.edu/jy057cz8322

[5] F. Eichas, S. Möller, and U. Zölzer, "Block-oriented modeling of distortion audio effects using iterative minimization," *Proc. Digital Audio Effects (DAFx-15), Trondheim, Norway*, 2015.

[6] A. Wishnick, "Time-Varying Filters for Musical Applications." in *DAFx*, 2014, pp. 69–76.

[7] F. Esqueda, B. Kuznetsov, and J. D. Parker, "Differentiable white-box virtual analog modeling," in *2021 24th International Conference on Digital Audio Effects (DAFx)*, 2021, pp. 41–48.

[8] J. Covert and D. L. Livingston, "A vacuum-tube guitar amplifier model using a recurrent neural network," *Conference Proceedings - IEEE SOUTHEASTCON*, 2013.

[9] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A Vacuum-Tube Guitar Amplifier Model Using Long/Short-Term Memory Networks," *Conference Proceedings - IEEE SOUTHEASTCON*, vol. 2018-April, 10 2018.

[10] A. Wright, E.-P. Damskägg, V. Välimäki, and others, "Real-time black-box modelling with recurrent neural networks," in *22nd international conference on digital audio effects (DAFx-19)*, 2019, pp. 1–8.

[11] E. P. Damskagg, L. Juvela, E. Thuillier, and V. Valimaki, "Deep Learning for Tube Amplifier Emulation," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 471–475, 5 2019.

[12] T. Schmitz and J.-J. Embrechts, "Nonlinear Real-Time Emulation of a Tube Amplifier with a Long Short Time Memory Neural-Network," 5 2018.

[13] J. Chowdhury, "A Comparison of Virtual Analog Modelling Techniques for Desktop and Embedded Implementations," 9 2020. [Online]. Available: https://arxiv.org/abs/2009.02833v1

[14] A. Wright, E. P. Damskägg, L. Juvela, and V. Välimäki, "Real-Time Guitar Amplifier Emulation with Deep Learning," *Applied Sciences 2020, Vol. 10, Page 766*, vol. 10, no. 3, p. 766, 1 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/3/766/htmhttps://www.mdpi.com/2076-3417/10/3/766

[15] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 890–894, 2021.

[16] E.-P. Damskägg, L. Juvela, V. Välimäki, and others, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf.(SMC-19), Malaga, Spain*, 2019, pp. 332–339.

[17] M. A. Ramírez, E. Benetos, and J. D. Reiss, "Deep Learning for Black-Box Modeling of Audio Effects," *Applied Sciences 2020, Vol. 10, Page 638*, vol. 10, no. 2, p. 638, 1 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/2/638

[18] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," *8th International Conference on Learning Representations, ICLR 2020*, 1 2020. [Online]. Available: https://arxiv.org/abs/2001.04643v1

[19] E. Perez-Gonzalez and J. D. Reiss, "Automatic mixing," *DAFX: Digital Audio Effects*, pp. 523–549, 2011.

[20] C. J. Steinmetz, "Learning to mix with neural audio effects in the waveform domain," 9 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4091203#.Y-KCNGfB4_U.mendeley

[21] D. Giannoulis, M. Massberg, and J. D. Reiss, "Digital Dynamic Range Compressor Design—A Tutorial and Analysis," *Journal of the Audio Engineering Society*, vol. 60, no. 6, pp. 399–408, 7 2012.

[22] S. Hawley, B. Colburn, and S. I. Mimilakis, "Profiling Audio Compressors with Deep Neural Networks," 10 2019.

[23] A. Wright, V. Välimäki, and others, "Grey-box modelling of dynamic range compression," in *Proc. Int. Conf. Digital Audio Effects (DAFX), Vienna, Austria*, 2022, pp. 304–311.

[24] A. Gu, K. Goel, and C. Ré, "Efficiently Modeling Long Sequences with Structured State Spaces," 10 2021. [Online]. Available: https://arxiv.org/abs/2111.00396v3

[25] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long Range Arena: A Benchmark for Efficient Transformers," 11 2020. [Online]. Available: http://arxiv.org/abs/2011.04006

[26] P. Depalle, S. Tassart, and S. Tassart, "State Space Sound Synthesis and a State Space Synthesiser Builder," pp. 1–1, 1995. [Online]. Available: https://hal.science/hal-01161430https://hal.science/hal-01161430/document

[27] B. Colburn and S. Hawley, "SignalTrain LA2A Dataset," 5 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3824876

[28] W. Mitchell and S. H. Hawley, "Exploring Quality and Generalizability in Parameterized Neural Audio Effects," *149th Audio Engineering Society Convention 2020, AES 2020*, 6 2020. [Online]. Available: https://arxiv.org/abs/2006.05584v1

[29] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," 3 2018. [Online]. Available: https://arxiv.org/abs/1803.01271v2

[30] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 3942–3951, 4 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11671

[31] J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 3 1990.

[32] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "HiPPO: Recurrent Memory with Optimal Polynomial Projections," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1474–1487. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/102f0bb6efb3a6128a3c750dd16729be-Paper.pdf

[33] A. Gupta, A. Gu, and J. Berant, "Diagonal State Spaces are as Effective as Structured State Spaces," 3 2022. [Online]. Available: https://arxiv.org/abs/2203.14343v3

[34] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, "Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers," in *Advances in*

Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 572–585. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/05546b0e38ab9175cd905eebcc6ebb76-Paper.pdf

[35] S. H. Hawley, B. Colburn, and S. I. Mimilakis, "SignalTrain: Profiling Audio Compressors with Deep Neural Networks," 5 2019. [Online]. Available: https://arxiv.org/abs/1905.11928v2

[36] A. Wright and V. Valimaki, "Perceptual loss function for neural modeling of audio systems," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 251–255, 5 2020.

[37] R. Yamamoto, E. Song, and J. M. Kim, "Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 6199–6203, 5 2020.

[38] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 71–75, 2021.