# DATA CONTINUITY MATTERS: IMPROVING SEQUENCE MODELING WITH LIPSCHITZ REGULARIZER

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

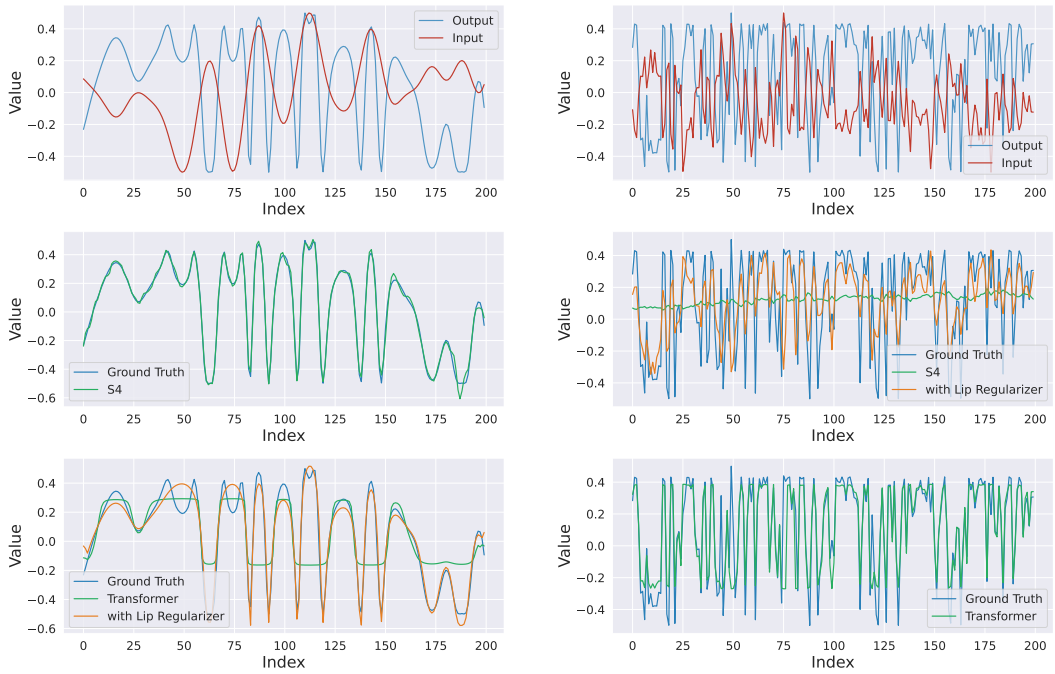
Sequence modeling is a core problem in machine learning, and various neural networks have been designed to process different types of sequence data. However, few attempts have been made to understand the inherent data property of sequence data, neglecting the critical factor that may significantly affect the performance of sequence modeling. In this paper, we theoretically and empirically analyze a generic property of sequence data, i.e., *continuity*, and connect this property with the performance of deep models. First, we empirically observe that different kinds of models for sequence modeling prefer data with different continuity. Then, we theoretically analyze the continuity preference of different models in both time and frequency domains. To further utilize continuity to improve sequence modeling, we propose a simple yet effective *Lipschitz Regularizer*, that can flexibly adjust data continuity according to model preferences, and bring very little extra computational cost. Extensive experiments on various tasks demonstrate that altering data continuity via Lipschitz Regularizer can largely improve the performance of many deep models for sequence modeling.

## 1 INTRODUCTION

Sequence modeling is a central problem in many machine learning tasks, ranging from natural language processing (Kenton & Toutanova, 2019) to time-series forecasting (Li et al., 2019). Although simple deep models, like MLPs, can be used for this problem, various model architectures have been specially designed to process different types of real-world sequence data, achieving vastly superior performance to simple models. For instance, the vanilla Transformer shows great power in natural language processing (Wolf et al., 2020), and its variant Informer (Zhou et al., 2021) is more efficient in time-series forecasting tasks. And a recent work Structured State Space sequence model (S4) (Gu et al., 2021) reaches SoTA in handling data with long-range dependencies. However, few attempts have been made to understand the inherent property of sequence data in various tasks, neglecting the critical factor which could largely influence the performance of different types of deep models. Such investigations can help us to understand the question that what kind of deep model is suitable for specific tasks, and is essential for improving deep sequence modeling.

In this paper, we study a generic property of sequence data, i.e., *continuity*, and investigate how this property connects with the performance of different deep models. Naturally, all sequence data can be treated as discrete samples from an underlying continuous function with time as the hidden axis. Based on this view, we apply continuity to describe the smoothness of the underlying function, and further quantify it with Lipschitz continuity. Then, it can be noticed that different data types have different continuity. For instance, time-series or audio data are more continuous than language sequences, since they are sampled from physical continuous signals evolved through time.

Furthermore, we empirically observe that *different deep models prefer data with different continuity*. We design a sequence-to-sequence task to show this phenomenon. Specifically, we generate two input sequences with different continuity, and map them to two output sequences using an identical randomly generated function. Then, we use two different deep models to learn this mapping. Each model has an identical 1D convolution embedding layer, and a separate sequence processing module. One uses the S4 model (Gu et al., 2021) and the other uses vanilla Transformer (Vaswani et al., 2017) (with the same number of layers and hidden dimensions). The results of this experiment are shown in Figure 1. It can be observed that the S4 model achieves significantly better performance with




(a) High Continuity

(b) Low Continuity

Figure 1: A Sequence-to-sequence task to show that different deep models prefer data with different continuity. The first raw shows input and output sequences. We generate input sequences with different continuities (left column: high continuity; right column: low continuity) and learn a mapping function using different models (second row: S4; third row: Transformer). We can see that S4 prefers more continuous sequences, while Transformer prefers more discrete sequences. And adjusting continuity according to the preferences of models with Lipschitz Regularizer can largely improve their performances. More details of this experiment are in Appendix A.

more continuous inputs, and the Transformer performs better with more discrete inputs. Note that, essentially, they are learning the same mapping only with different data continuity. This clearly shows different models prefer different data continuity.

Inspired by the above observation, we hypothesize that it is possible to enhance model performance by changing the data continuity according to their preferences. To make the proposed method simple and applicable for different deep models, we derive a surrogate that can be directly optimized for the Lipschitz continuity, and use it as a regularizer in the loss function. We call the proposed surrogate **Lipschitz Regularizer**, which depicts the data continuity and can also be used to adjust it.

Then, we investigate the data continuity property for different models and how to use Lipschitz Regularizer to change data continuity according to the model preference. We provide in-depth analyses in both time and frequency domains. On the one hand, Lipschitz continuity describes the continuity of sequences over time, which is a feature in the time domain. Here, we investigate two models. One is a continuous-time model S4, and the other model Informer is based on self-attention. As for S4, since the fitting error of the S4 model is bounded by the Lipschitz constant, S4 prefers smoother input sequences with smaller Lipschitz constant. Hence, we make the inputs of S4 layers more continuous by adding the Lipschitz Regularizer to the loss function. Experiment results on the Long Range Arena benchmark demonstrate that Lipschitz Regularizer can largely improve the performance of the S4 model, especially for tasks with discrete inputs. Conversely, Informer is built upon self-attention, which is designed to process some tokenized discrete data, e.g., text, so Informer prefers less continuous sequences. Therefore, we decrease the continuity of input sequences by subtracting the Lipschitz Regularizer from the loss function. Prediction performance and empirical analyses on many time-series tasks well prove the superiority of the Lipschitz Regularizer. Also, we observe the same results on the task mentioned above as shown in Figure 1.

On the other hand, for the frequency domain, we find that Lipschitz Regularizer represents the expectation of the frequency of data's underlying function. Here, we take the ReLU network as the studied case, and theoretically justify that Lipschitz Regularizer is related to the spectral bias - a phenomenon that neural networks tend to prioritize learning the low-frequency modes. We then propose to use Lipschitz Regularizer by subtracting it from the loss function to mitigate the spectral bias. In this way, neural networks are forced to learn high-frequency parts, and convergence can be accelerated since information in different frequency bands can be learned simultaneously.

In summary, Lipschitz Regularizer can be used to flexibly adjust data continuity for a wide range of deep models which has the preference on the data continuity. It improves various models with very little extra computational cost, shedding a light on inherent data property analyses for sequence modeling.

## 2 Related Work

**Deep Neural Networks for Sequence Modeling** Sequence modeling plays a critical role in many machine learning problems. Many general architectures, including MLPs (Rahaman et al., 2019), RNNs (Mikolov et al., 2010) and CNNs (Bai et al., 2018), can all be used for sequence modeling. And recently, two types of models show great power in addressing challenges for sequence modeling, such as handling complex interactions and long-range dependencies. The first type is self-attention-based models. For instance, the vanilla Transformer (Vaswani et al., 2017), Informer (Zhou et al., 2021) and Preformer (Choromanski et al., 2020) all show great performance on natural language processing, time-series forecasting and speech processing, respectively. Another type is continuous-time models, which is built upon the view that inputs are sampled from continuous functions. They include but not limit to Neural ODE (Chen et al., 2018), Lipschitz RNN (Erichson et al., 2020), State-space model (Gu et al., 2021). In this paper, we do not aim at proposing novel models like previous works, but we focus on understanding intrinsic preference on input sequences. We show these two type of models both have preference on the data continuity property, and we utilize it to promote their performance.

**Lipschitz Continuity of Neural Networks** The Lipschitz continuity is a general property for any function, and is widely used for analyzing different kinds of neural networks, including MLPs (Zhang et al., 2021; Gouk et al., 2021), CNNs (Zou et al., 2019), self-attention-based networks (Dasoulas et al., 2021), graph neural networks (Gama et al., 2020) and GANs (Gulrajani et al., 2017). It becomes an essential property of neural networks, and can be used in various ways, such as improving adversarial robustness (Meunier et al., 2022) and proving generalization bounds (Sokolić et al., 2017). In this paper, we focus on the Lipschitz continuity of the underlying function of sequence data, and use it as a data property, but not property of models.

## 3 Lipschitz Continuity of Sequence data

In this section, we first show the measure for continuity of sequence data, and how it can be used as a regularizer. We give a definition for Lipschitz Regularizer here, and leave detailed analyzes and usages of it in specific models in the rest of the sections. Then, we provide views for Lipschitz Regularizer in both time and frequency domains.

To define the measure for the continuity of sequence data, we view inputs as signals, and data points in the sequence are discrete samples of an underlying continuous function with certain time steps. Next, we calculate the Lipschitz constant of the underlying function, which is widely used as the measure for continuity. Specifically, suppose the sequence is $x_0, x_1, \ldots, x_n$, and the underlying function is defined as $f(t_0) = x_0, f(t_1) = x_1, \ldots, f(t_n) = x_n$, where $t_0, t_1, \ldots, t_n$ are time steps. Then, if we let $t_0 = 0, t_1 = 1, \ldots, t_n = n$, the Lipschitz constant $L_f$ of function $f$ is

$$L_f = \max_{t_i, t_j \in \{0,1,\ldots,n\}} \frac{|f(t_i) - f(t_j)|}{|t_i - t_j|} = \max_{i,j \in \{0,1,\ldots,n\}} \frac{|x_i - x_j|}{|i - j|}. \tag{1}$$

By Mean Value Theorem, for $i, j \in \{0, 1, \ldots, n\}$ and $j - i > 1$, we could always find an interval $[k, k+1]$ of time step 1 such that $i \leq k \leq j-1$, $\frac{|x_i - x_j|}{|i-j|} \leq |x_{k+1} - x_k|$. Therefore, we have

$$L_f = \max_{i,j \in \{0,1,\ldots,n\}} \frac{|x_i - x_j|}{|i-j|} = \max_{k \in \{0,1,\ldots,n-1\}} |x_{k+1} - x_k|. \tag{2}$$

However, since we would like to adjust this continuity according to the preferences of different models, this measure should be easy to be optimized, but it is hard to pass gradients due to the $\max$ operator. To help with the optimization process, we design a surrogate by taking the average over all terms and changing the norm to $\ell_2$. Moreover, since we simply use this surrogate as a regularizer in the loss function to flexibly adjust the continuity for various models, we name this term as *Lipschitz Regularizer*, and its formal definition is given as follows.

**Definition 3.1.** ***(Lipschitz Regularizer)*** *Suppose the sequence is $x_0, x_1, \ldots, x_n$. We define the Lipschitz Regularizer in the following equation:*

$$\mathcal{L}_{\text{Lip}} = \frac{1}{n} \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 \tag{3}$$

### 3.1 View Lipschitz Regularizer in Time and Frequency Domains

We then provide two views for Lipschitz Regularizer. On the one hand, Lipschitz Regularizer is a feature for sequence data in the time domain, representing the continuity of sequences over time. Thus, it can be used to alter the continuity of input sequences to specific models. As shown in Figure 1, different deep models have different preferences for data continuity. We can use the Lipschitz Regularizer to manually make the sequences more or less continuous, and therefore improve the performance of the model. An example of increasing the continuity to improve the performance of a continuous-time model is described in §4.1. A converse example of decreasing the continuity to improve the performance of an attention-based model is described in §4.2.

On the other hand, from the frequency perspective, Lipschitz Regularizer directly relates to the frequency of the function, and can be used to alter modes with different frequencies. Specifically,

$$\begin{aligned} \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 &= \sum_{i=0}^{n-1} \left( \frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i} \right)^2 \approx \int_{\mathbb{R}} \left( \frac{\mathrm{d}f(t)}{\mathrm{d}t} \right)^2 \mathrm{d}t \\ &= \int_{\mathbb{R}} (2\pi i \xi)^2 \, \hat{f}(\xi)^2 (-\mathrm{d}\xi) = 4\pi^2 \int_{\mathbb{R}} \xi^2 \hat{f}(\xi)^2 \mathrm{d}\xi = 4\pi^2 \mathbb{E}_{\hat{f}(\xi)}[|\xi^2|], \end{aligned} \tag{4}$$

where $\hat{f}(\xi) \coloneqq \int f(x) e^{-i2\pi\xi x} \mathrm{d}x$ is the Fourier transform of $f$, and $\xi$ is the frequency. Essentially, the Lipschitz Regularizer of sequence data represents the exception of the frequency of data's underlying function. Besides, previous literature shows that neural networks tend to prioritize the learning of low-frequency parts of the target function (Rahaman et al., 2019). We find that Lipschitz Regularizer can be utilized to mitigate this phenomenon by emphasizing high-frequency parts, which allows the network to fit all spectra simultaneously and results in a faster convergence rate. The details of this discussion are in §5.1.

## 4 Time Domain

In this section, we view Lipschitz Regularizer in the time domain, and show how it can be used to make the sequence more discrete or continuous over time, catering to the preference of different models. Generally, to change the continuity of the input sequence to different models with the Lipschitz Regularizer, we apply it on the output of the embedding layer before the sequence is sent to different models. We describe the details of two different models in the following sections.

### 4.1 State Space Model

The State Space Model is a classic model in control engineering. Gu et al. (2021) extended it to the deep sequence model, and proposed the S4 model. S4 is a continuous-time sequence model.

It advances SoTA on long-range sequence modeling tasks by a large margin. An S4 layer can be denoted as follows:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du, \end{aligned} \tag{5}$$

where $u$ is the input function, $x$ is the hidden state , $y$ is the output. $A$, $B$, $C$, $D$ are trainable matrices.

The critical and essential design in the S4 layer is the transition matrix $A$, which is initialized with the HiPPO matrix. The HiPPO matrix makes the S4 layer optimally remember the history of the input's underlying function, so the S4 model can substantially outperform previous methods on long-range sequence modeling tasks. Particularly, the HiPPO matrix is designed to find the best polynomial approximation of the input's underlying function given a measure that defines the optimal history, and a memory budget which is the hidden dimension in the model. Each measure corresponds to an optimal HiPPO matrix.

**Theoretical Analyses** To connect the continuity property with the S4 model, we have the following proposition.

**Proposition 4.1.** *Suppose $f_1, f_2 : \mathbb{R}_+ \to \mathbb{R}$ are two differentiable functions of input sequences, and their Lipschitz constant are $L_{f_1}$ and $L_{f_2}$. The HiPPO matrix with scaled Legendre measure (LegS) is denoted as HiPPO-LegS. Let the error of the HiPPO-LegS projection of $f_1, f_2$ at time $t$ be $\delta_1(t), \delta_2(t)$, respectively. For any time $t$, if $L_{f_1} \leq L_{f_2}$, then $\delta_1(t) \leq \delta_2(t)$.*

Proof is presented in Appendix D. Proposition 4.1 shows that if the input sequence is more continuous, the error will be smaller. This is also true for S4 layers, since the HiPPO matrix is the most critical design in the S4 layer. However, in many tasks, such as natural language processing, the input sequence are not particularly smooth. This will deteriorate the performance of the S4 model.

Lipschitz Regularizer can be used to solve the above problem, because it can adjust the continuity of sequences. Specifically, since we cannot directly manipulate the underlying function of the input sequence, we add a 1D convolutional layer that does not change the sequence length as an embedding layer before the S4 layer, and then apply Lipschitz Regularizer to the output of the embedding layer as follows:

$$\mathcal{L}(y, \hat{y}, \hat{l}) = \mathcal{L}_{\mathrm{S4}}(y, \hat{y}) + \lambda \mathcal{L}_{\mathrm{Lip}}(\hat{l}), \tag{6}$$

where $y$ is the ground-truth, and $\hat{y}$ is the output of the S4 model. $\hat{l}$ is the output of the embedding layer, and $L_{\mathrm{S4}}$ is the original loss of the S4 model. $\lambda$ is a hyperparameter to control the magnitude of the Lipschitz Regularizer. By using Equation (6) as the loss function, the input of the S4 layers becomes more continuous, so the error of the HiPPO-LegS projection and S4 layer can be reduced, leading to better model performance.

**Experiments** To demonstrate the effectiveness of the Lipschitz Regularizer, we use a modified version of the Long Range Arena (LRA) (Tay et al., 2020) benchmark with harder tasks. The descriptions of the original LRA are in Appendix B. In addition to the original LRA, we create 2 harder tasks with more discrete sequences. Particularly, we notice that among these 5 tasks, 2 of them use pixels as inputs (i.e., Image and Pathfinder), which could be more continuous than texts in the other 3 tasks. So we design *Image-c* and *Pathfinder-c*, in which the contrast of images is increased, and the increasing degree is randomly sampled from 50% to 100% for each image.

We test three methods on the modified LRA. The first one is the original S4 model (denoted as *S4*). The second one is the S4 model with a 1D convolutional layer as the embedding layer, and Lipschitz Regularizer is applied to the outputs of the embedding layer (denoted as *S4 + Emb + Lip*). Furthermore, we also design the third model to ablate the effect of the extra embedding layer. Here, we use the S4 model with the same embedding layer as the previous method, and Lipschitz Regularizer is not applied (denoted as *S4 + Emb*). Hyperparameter $\lambda$ is chosen from $\{1, 2, 3, 4, 5\}$ when the model performs best on the validation set.

Results are listed in Table 1, and we have the following observations. (1) It is obvious that our method (i.e., S4 + Emb + Lip) significantly outperforms other methods in almost all tasks, especially in tasks with discrete inputs, such as Text and Retrieval. Improved performance in Image-c and Pathfinder-c shows that Lipschitz Regularizer can mitigate the influence of increased contrasts.

Table 1: Accuracy of the S4 model and its variant with our proposed Lipschitz Regularizer (S4 + Emb + Lip) in LRA. S4 + Emb is set to ablate the effect of the extra embedding layer.

| | ListOps | Text | Retrieval | Image | Image-c | Pathfinder | Pathfinder-c |
|---|---|---|---|---|---|---|---|
| S4 | 59.53 | 86.51 | 91.07 | 88.54 | 84.27 | **94.02** | 89.11 |
| S4 + Emb | 58.94 | 87.12 | 90.28 | 87.25 | 85.13 | 92.37 | 90.32 |
| **S4 + Emb + Lip** | **61.37** | **89.74** | **93.83** | **89.19** | **88.43** | 93.52 | **91.39** |

These results well demonstrate the effectiveness of the Lipschitz Regularizer, indicating that it can make input sequences of the S4 layer more continuous, and better cater to the preference of the S4 model. (2) Comparing results of S4 on Image/Pathfinder and Image-c/Pathfinder-c, it can be observed that the performance of the S4 model degenerates with the increasing contrasts of images. The cause is the deceased continuity against the preference of the S4 model, verifying that the S4 model indeed prefers continuous inputs. (3) Only adding the extra embedding layer (S4 + Emb) makes the accuracy decrease in 4 out of 7 tasks, indicating that improvements come from the effect of the Lipschitz Regularizer, but not the extra layer.

### 4.2 INFORMER

In this section, we show that Lipschitz Regularizer can improve the performance of Transformer-based models when inputs are continuous. In particular, we choose time-series forecasting tasks whose inputs are highly continuous, and we use a typical variant of Transformer, i.e., Informer, which is designed for these tasks. Although Informer already has good performance on time-series forecasting tasks, due to the preference of Transformer-based models for discrete sequences (shown in Figure 1) and highly continuous inputs, we can still apply Lipschitz Regularizer to further improve the model by decreasing the continuity of input sequences. Specifically, since Informer has an embedding layer, we directly apply Lipschitz Regularizer to the output of the embedding layer as follows:

$$\mathcal{L}(y, \hat{y}, \hat{l}) = \mathcal{L}_{\text{Informer}}(y, \hat{y}) - \lambda \mathcal{L}_{\text{Lip}}(\hat{l}), \tag{7}$$

where $y$ is the ground-truth, and $\hat{y}$ is the output of the Informer model. $\hat{l}$ is the output of the embedding layer, and $L_{\text{Informer}}$ is the original loss of the Informer model. $\lambda$ controls the magnitude of the Lipschitz Regularizer. Note that different from the usage in the S4 model, here we subtract Lipschitz Regularizer to make the input discrete, and cater to the model preference.

**Experiments** We use 5 datasets in this experiments and their descriptions are in Appendix C. Evaluation metrics are Mean Square Error (MSE) and Mean Absolute Error (MAE). Hyperparameter $\lambda$ is chosen from $\{1, 2, 3, 4, 5\}$ when the model performs best on the validation set.

Results of *Informer* and Informer with Lipschitz Regularizer (denoted as *Informer + Lip*) are shown in Table 2. We can see that Informer + Lip generally outperforms the original model on most of the tasks. This indicates that Transformer-based models prefer discrete sequences and reducing input continuity with Lipschitz Regularizer can be helpful for them.

Besides, to investigate whether Lipschitz Regularizer changes the data continuity, we also show curves tracking the Lipschitz constant of the output of the embedding layer during training in Figure 2. Curves show that with the Regularizer, the Lipschitz constant becomes larger than it in the original model, and continuously increases during training. Results demonstrate that Lipschitz Regularizer can indeed change the continuity, and thus improve the model.

We also show an ablation study for the hyperparameter $\lambda$ in Figure 3. We can observe that (1) MSE increases when $\lambda < 0$, while decreases when $\lambda > 0$. Since positive $\lambda$ reduces the data continuity, we can conclude that Informer prefers discrete sequences, and Lipschitz Regularizer can reduce the continuity and cater to the preference; (2) MSEs do not have a large variance for different positive $\lambda$, indicating that the performance improvement is not sensitive to hyperparameter changes.

Table 2: Mutivraiate and univariate time-series forecasting results of Informer and Informer with Lipschitz Regularizer (Informer + Lip). Note in this table, prediction window sizes are converted to lengths of sequences used in the model.

| | | Multivariate | | | | Univariate | | | |
|---|---|---|---|---|---|---|---|---|---|
| Methods | | Informer | | Informer + Lip | | Informer | | Informer + Lip | |
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| $ETTh_1$ | 24 | 0.577 | 0.549 | **0.535** | **0.518** | 0.098 | 0.247 | **0.089** | **0.237** |
| | 48 | 0.685 | 0.625 | **0.683** | **0.616** | 0.158 | 0.319 | **0.126** | **0.283** |
| | 168 | **0.931** | **0.752** | 1.022 | 0.799 | 0.183 | 0.346 | **0.164** | **0.326** |
| | 336 | 1.128 | 0.873 | **0.991** | **0.754** | 0.222 | 0.387 | **0.118** | **0.270** |
| | 720 | 1.215 | 0.896 | **1.203** | **0.889** | 0.269 | 0.435 | **0.131** | **0.287** |
| $ETTh_2$ | 24 | 0.72 | 0.665 | **0.337** | **0.431** | 0.093 | 0.240 | **0.086** | **0.226** |
| | 48 | **1.457** | **1.001** | 1.811 | 1.083 | 0.155 | 0.314 | **0.137** | **0.289** |
| | 168 | 3.489 | 1.515 | **3.053** | **1.443** | **0.232** | **0.389** | 0.254 | 0.417 |
| | 336 | 2.723 | 1.340 | **2.355** | **1.287** | 0.263 | 0.417 | **0.249** | **0.408** |
| | 720 | **3.467** | **1.473** | 3.723 | 1.667 | 0.277 | 0.431 | **0.236** | **0.396** |
| $ETTm_1$ | 24 | **0.323** | **0.369** | 0.351 | 0.391 | 0.030 | 0.137 | **0.018** | **0.091** |
| | 48 | 0.494 | 0.503 | **0.444** | **0.453** | 0.069 | 0.203 | **0.058** | **0.197** |
| | 96 | 0.678 | 0.614 | **0.474** | **0.477** | 0.194 | 0.372 | **0.133** | **0.301** |
| | 288 | 1.056 | 0.786 | **1.033** | **0.808** | 0.401 | 0.554 | **0.303** | **0.469** |
| | 672 | 1.192 | 0.926 | **0.905** | **0.721** | 0.512 | 0.644 | **0.275** | **0.454** |
| Weather | 24 | 0.335 | **0.381** | **0.327** | 0.383 | 0.117 | 0.251 | **0.113** | **0.238** |
| | 48 | 0.395 | 0.459 | **0.381** | **0.431** | **0.178** | **0.318** | 0.191 | 0.324 |
| | 168 | 0.608 | 0.567 | **0.606** | **0.564** | 0.266 | 0.398 | **0.251** | **0.374** |
| | 336 | 0.702 | **0.620** | **0.673** | 0.621 | 0.297 | 0.416 | **0.247** | **0.377** |
| | 720 | 0.831 | 0.731 | **0.658** | **0.604** | 0.359 | 0.466 | **0.265** | **0.392** |
| ECL | 48 | 0.344 | 0.393 | **0.258** | **0.357** | 0.239 | 0.359 | **0.173** | **0.305** |
| | 168 | **0.368** | **0.424** | 0.373 | 0.434 | 0.447 | 0.503 | **0.188** | **0.315** |
| | 336 | 0.381 | 0.431 | **0.346** | **0.421** | 0.489 | 0.528 | **0.378** | **0.437** |
| | 720 | 0.406 | 0.443 | **0.359** | **0.437** | 0.540 | 0.571 | **0.445** | **0.504** |
| | 960 | 0.460 | 0.548 | **0.445** | **0.504** | 0.582 | 0.608 | **0.377** | **0.458** |
| Count | | 12 | | 38 | | 4 | | 46 | |

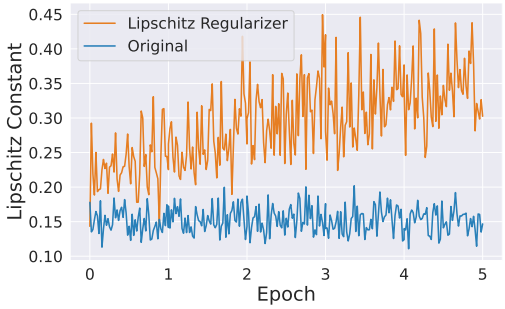



Figure 2: The Lipschitz constant of the output of the embedding layer during the training process. The experiment is the multivariate $ETTh_2$ with the prediction window size of 24h.

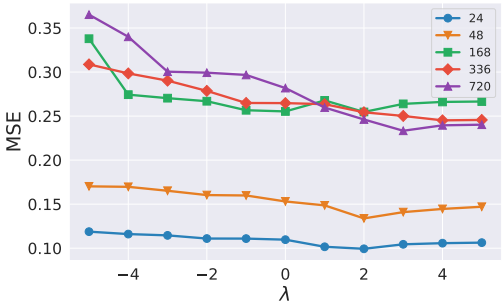



Figure 3: MSE of Informer + Lip on $EETh_2$ with different $\lambda$. Colors represent different window sizes.

## 5 Frequency Domain

In this section, we study how continuity affects the performance of deep models from the frequency perspective. We take the ReLU network as the study case, and provide theoretical analyses and experiment results to show the effectiveness of applying the Lipschitz Regularizer on ReLU networks.

### 5.1 ReLU Network

A ReLU network $g : \mathbb{R}^d \to \mathbb{R}$ with $L$ hidden layers of width $d_1, \ldots, d_L)$ is defined as:

$$g(\mathbf{x}) = \left(T^{(L+1)} \circ \sigma \circ T^{(L)} \circ \cdots \circ \sigma \circ T^{(1)}\right)(\mathbf{x}), \tag{8}$$

where $T^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ is an affine function ($d_0 = d$, $d_{L+1} = 1$) and $\sigma$ is the ReLU activation function.

**Theoretical Analyses** In the previous literature, Rahaman et al. (2019) showed that the low-frequency part of the sequence data is learned faster by the ReLU network, and such phenomenon is called the "spectral bias". We claim that the Lipschitz Regularizer could help mitigate the spectral bias. Consider the following proposition:

**Proposition 5.1.** *Suppose there are two ReLU network $g_{\theta_1}, g_{\theta_2}$ with identical architecture, and the Lipschitz constant of them are $L_1, L_2$, respectively. If $L_1 \leq L_2$, then $\hat{g}_{\theta_1}(\xi) \leq \hat{g}_{\theta_2}(\xi)$, where $\hat{g}_\theta(\xi)$ is the Fourier component of $g_\theta$.*

Proof is presented in Appendix D. According to Proposition 5.1, when the Lipschitz constant of the ReLU network is increased, the frequency will also increase. This inspires us to balance frequency modes via changing the Lipschitz continuity of functions. Besides, suppose we use a ReLU network to learn a sequence-to-sequence mapping, where values of data in the input sequence (length $n$) increase linearly in the interval $(0, 1)$ with step size $\frac{1}{n}$, and the output is generated by the mapping function $h(t)$. Note that since values of data in the input sequence increase linearly, the Lipschitz constant of the ReLU network is the same as the output sequence. Therefore, we design the decayed Lipschitz Regularizer as follows:

$$\mathcal{L}(y, \hat{y}) = \mathcal{L}_{\text{MSE}}(y, \hat{y}) - \lambda e^{-\epsilon t} \mathcal{L}_{\text{Lip}}(\hat{y}), \tag{9}$$

where $y$ is the ground-truth generated by $h(t)$ and $\hat{y}$ is the prediction. $\mathcal{L}_{\text{MSE}}$ is the MSE Loss. $\lambda$ and $\epsilon$ are hyperparameters that control the magnitude and decay rate of the Lipschitz Regularizer, respectively.

We further explain the reasons why this regularizer can mitigate spectral bias in two aspects. First, by Equation (4), the added term could be seen as a direct penalty to the low-frequency part of the output sequence. Since the value of data in the input sequence increases linearly, this is equivalent to penalizing the low-frequency part of the ReLU network, and prioritizing the learning of the high-frequency part.

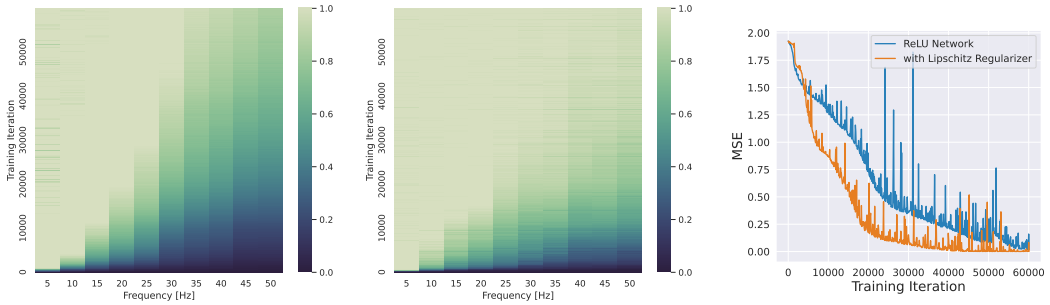
In another perspective, Rahaman et al. (2019) claimed that the origin of the spectral bias is the gradually increasing parameter norm, and Lipschitz Regularizer can intentionally relieve it. Specifically, Fourier components of the ReLU network $\hat{g}_\theta(\xi)$ is bounded by $O(L_g)$, and $L_g$ is bounded by the parameter norm, which can only increase by a small step during the optimization step. Hence, gradually increasing parameter norms can hinder the learning of high-frequency parts at the early optimization stage. Besides, due to the fact that Lipschitz Regularizer can intentionally change $L_g$, subtracting Lipschitz Regularizer as Equation (9) can enlarge the parameter norm, making it possible for optimizing both high and low-frequency parts. This can be seen as a warm-up process for the network where the parameter norm increases at the beginning of optimization, and then the convergence can be significantly accelerated, since modes of all frequencies can be learned simultaneously after the warm-up.

**Experiments** We choose a sequence-to-sequence mapping task to evaluate the proposed Lipschitz Regularizer. Specifically, we try to learn the mapping function whose input is the sequence with linearly increasing values, and output is a highly periodic sequence. Given frequencies $K = \{k_1, k_2, \ldots, k_n\}$, amplitudes $A = \{a_1, a_2, \ldots, a_n\}$, and phases $\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$, the mapping function is defined as $h(x) = \sum_{i=1}^{n} A_i \sin(2\pi k_i x + \phi_i)$.

In this experiment, we take $n = 10$ and frequencies $K = \{5, 10, \ldots, 45, 50\}$, amplitudes $A = \{0.1, 0.2, \ldots, 1\}$. The phases are uniformly sampled from 0 to $2\pi$, i.e., $\phi_i \sim U(0, 2\pi)$. The input samples in the sequence are uniformly placed over $(0, 1)$ with the number of samples $N = 100$, and the output is generated by $h(x)$.

As for the model, we use a 6-layer deep ReLU network with the hidden dimension set to 256 for all layers. To verify the effectiveness of the proposed Lipschitz Regularizer, we train two identical networks with the same training procedure. One is trained with the decayed Lipschitz Regularizer and the other without it. We set hyperparameter $\lambda \in \{1, 2, 3, 4, 5\}$ and $\epsilon \in \{0.00001, 0.0001, 0.001, 0.01, 0.1\}$ when the model performs best on the validation set.

We show the frequency and MSE of ReLU networks during the training process in Figure 4. From Figure 4 (a) and (b), we notice that low-frequency parts are learned first in both networks, but




(a) Frequency of a ReLU network

(b) Frequency of an identical ReLU network with Lipschitz Regularizer

(c) MSE of the ReLU network w and w/o Lipschitz Regularizer

Figure 4: Evolution of the frequency and MSE of ReLU networks during the training process. In (a) and (b), color indicates the normalized amplitude of the Fourier component at the corresponding frequency, i.e., $|\hat{g}_\theta(k_i)|/A_i$. Lipschitz Regularizer enables faster learning of high frequencies and faster convergence.

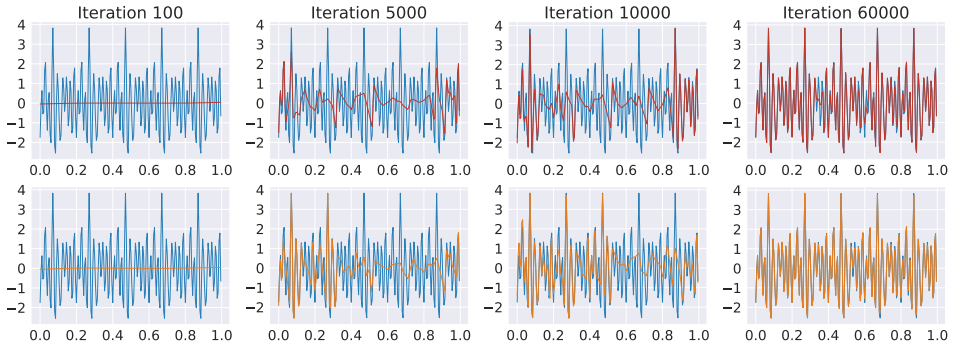



Figure 5: Predictions of ReLU networks (first row: without Lipschitz Regularizer; second row: with Lipschitz Regularizer) during the training process. With Lipschitz Regularizer, high-frequency parts can be learned faster.

with the decayed Lipschitz Regularizer, high frequencies can be learned significantly faster. Figure 4 (c) shows that Lipschitz Regularizer can accelerate convergence. We also show predictions of two models during the training process in Figure 5, which gives a more intuitive result, indicating that high frequencies can be learned faster when we use decayed Lipschitz Regularizer to warm-up optimization. All results demonstrate that Lipschitz Regularizer enables almost simultaneous learning for all frequencies, so spectral bias can be relieved in this way, and the convergence is accelerated.

## 6 Summary

We investigate a generic property of sequence data, i.e., *continuity*, which is closely related to the performance of different models, and propose *Lipschitz Regularizer* to flexibly adjust the continuity for various models. We first empirically observe that the different deep models prefer different data continuity. Then, from both time and frequency domains, we provide in-depth theoretical and experimental studies for specific models. For the time domain, we show that the continuous-time model S4 prefers continuous sequences, while the Transformer-based model Informer prefers discrete inputs. We use Lipschitz Regularizer to adjust the data continuity for both of them and largely improve their performance by catering to their preference. For the frequency domain, we show that Lipschitz Regularizer can help mitigate the spectral bias, and accelerate convergence for ReLU networks. In general, Lipschitz Regularizer is available for any sequence modeling tasks and models which have a preference for data continuity, and can accordingly facilitate learning for various models with very little computational cost.

## REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In *International Conference on Machine Learning*, pp. 2456–2466. PMLR, 2021.

N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W Mahoney. Lipschitz recurrent neural networks. In *International Conference on Learning Representations*, 2020.

Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33: 1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Laurent Meunier, Blaise J Delattre, Alexandre Araujo, and Alexandre Allauzen. A dynamical system perspective for lipschitz neural networks. In *International Conference on Machine Learning*, pp. 15484–15500. PMLR, 2022.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 1045–1048. Makuhari, 2010.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *International Conference on Machine Learning*, pp. 12368–12379. PMLR, 2021.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021.

Dongmian Zou, Radu Balan, and Maneesh Singh. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 66(3):1738–1759, 2019.

# APPENDIX

## A DETAILS OF EXPERIMENT IN THE INTRODUCTION

In this experiment, we generate two input sequences with different continuity, and map them to output with the same randomly generated function $h(x) = \sum_{i=1}^{n} \sin(2\pi k_i x + \phi_i)$, where $n = 20, k_i \sim N(0, 3)$, $\phi_i \sim N(0, 1)$ and $N$ denotes the normal distribution. The Lipschitz constant of the input and output sequence is shown in Table 3. Note that high Lipschitz constant represents low continuity, while low Lipschitz constant represents high continuity. Then, we train the S4 model and the Transformer model with generated input and output sequences. Each model has an 1D convention embedding layer with kernel size 5, stride 1, and padding 2. Both Transformer and S4 have 1 separated layer with hidden dimension set to 64. We also apply Lipschitz Regularizer to the output of the embedding layer and train models again. MSE of these 4 models is shown in Table 3. We could observe that S4 performs better with continuous inputs and Transformer is better with discrete inputs. Also, Lipschitz Regularizer can improve the performance of S4 and Transformer by changing the data continuity into their prefers ones.

Table 3: Details of experiment in the Introduction.

| | | High Continuity | Low Continuity |
|---|---|---|---|
| Lipschitz Constant | Input | 0.1137 | 0.5263 |
| | Output | 0.4381 | 0.9309 |
| MSE | S4 | 0.0006 | 0.0887 |
| | S4 + Lip | - | 0.0399 |
| | Transformer | 0.0482 | 0.0189 |
| | Transformer + Lip | 0.0051 | - |

## B DETAILS OF EXPERIMENT FOR THE S4 MODEL

In this experiment, we test the Lipschitz Regularizer on the S4 model. The testing dataset is the Long Range Arena (LRA) benchmark. Specifically, LRA contains various long sequence modeling tasks with sequence lengths ranging from 1K to 16K, which is very challenging for deep models. Tasks in LRA include (1) *ListOps*, in which the model needs to calculate the answer with a list of algebraic operations, (2) *Text*, a binary classification task with byte-level texts from IMDB reviews, (3) *Retrieval*, a document retrieval task with byte-level texts and documents from ACL Anthology Network, (4) *Image*, an image classification task with the image in CIFAR-10 dataset flattened into the pixel sequence, (5) *Pathfinder*, in which the model needs to deduce whether two points in the image are connected by dashed lines, and the image is also flattened into the pixel sequence.

## C DETAILS OF EXPERIMENT FOR THE INFORMER MODEL

In this experiment, we show that the Lipschitz Regularizer could improve the performance of the Informer model. We use 3 real-world datasets: ETT (Electricity Transformer Temperature), ECL (Electricity Consuming Load), and Weather. ETT has 3 separate datasets, i.e., $\{\text{ETTh}_1, \text{ETTh}_2\}$ for 1-hour-level with 2 separated countries, and $\text{ETTm}_1$ for 15-minute-level. And we use multiple prediction window sizes, including $\{24h, 48h, 168h, 336h, 720h, 960h\}$ for ETTh, ECL, Weather, and $\{6h, 12h, 24h, 72h, 168h\}$ for ETTm.

## D PROOFS OF PROPOSITIONS

### D.1 PROOF OF PROPOSITION 4.1

**Proposition.** *Suppose $f_1, f_2 : \mathbb{R}_+ \to \mathbb{R}$ are two differentiable functions of input sequences, and their Lipschitz constant are $L_{f_1}$ and $L_{f_2}$. The HiPPO matrix with scaled Legendre measure (LegS)*

*is denoted as HiPPO-LegS. Let the error of the HiPPO-LegS projection of $f_1, f_2$ at time $t$ be $\delta_1(t), \delta_2(t)$, respectively. For any time $t$, if $L_{f_1} \leq L_{f_2}$, then $\delta_1(t) \leq \delta_2(t)$.*

*Proof.* By Gu et al. (2020, Proposition 6), the LegS measure, which uniformly weighs all history, has the following property. Suppose the HiPPO-LegS projection for the target function $f(t)$ at time $t$ is $p^{(t)} = \mathrm{proj}_t(f)$, then the error $\delta_f(t) = \left\| f_{\leq t} - p^{(t)} \right\| = O(tL_f/\sqrt{N})$, where $L_f$ is the Lipschitz constant of $f(t)$, and the maximum polynomial degree is $N-1$. Therefore, when $L_{f_1} \leq L_{f_2}$, then $\delta_1(t) \leq \delta_2(t)$. □

### D.2 Proof of Proposition 5.1

**Proposition.** *Suppose there are two ReLU network $g_{\theta_1}, g_{\theta_2}$ with identical architecture, and the Lipschitz constant of them are $L_1, L_2$, respectively. If $L_1 \leq L_2$, then $\hat{g}_{\theta_1}(\xi) \leq \hat{g}_{\theta_2}(\xi)$, where $\hat{g}_\theta(\xi)$ is the Fourier component of $g_\theta$.*

*Proof.* By Rahaman et al. (2019, Theorem 1), for a ReLU network $g_\theta$ with parameter $\theta$, its Fourier component is,

$$\hat{g}_\theta(\xi) = \sum_{n=0}^{d} \frac{G_n(\theta, \xi)}{\|\xi\|^{n+1}} \tag{10}$$

where the numerator $G_n(\theta, \cdot) : \mathbb{R}^d \to \mathbb{C}$ is bounded by $O(L_g)$. Therefore, when $L_1 \leq L_2$, we have $\hat{g}_{\theta_1}(\xi) \leq \hat{g}_{\theta_2}(\xi)$. □