Tuning Frequency Bias of State Space Models

Annan Yu, Dongwei Lyu, Soon Hoe Lim, Michael W. Mahoney, N. Benjamin Erichson

[Paper] [Poster]

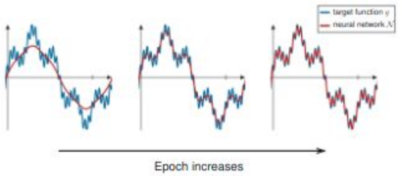
16 November, 2025

Tuning Frequency Bias of State Space Models

Annan Yu[1], Dongwei Lyu[2], Soon Hoe Lim[3,4], Michael W. Mahoney[5,6,7], N. Benjamin Erichson[5,6]
[1]Cornell University [2]University of Chicago [3]Department of Mathematics, KTH Royal Institute of Technology [4]Nordita, KTH Royal Institute of Technology and Stockholm University
[5]Lawrence Berkeley National Laboratory [6]International Computer Science Institute [7]University of California, Berkeley

What is Frequency Bias?

The term "frequency bias" originated from the study of an overparameterized multilayer perceptron (MLP), where it was observed that the low-frequency content was learned much faster than the high-frequency content. It is a form of implicit regularization.



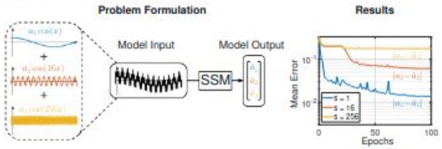
Frequency bias is a double-edged sword: it partially explains the good generalization capability of deep learning models but also puts a curse on learning the useful high-frequency information in the target.

State Space Models (SSMs)

State-space models (SSMs) leverage linear, time-invariant (LTI) systems,

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t),$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t),$$

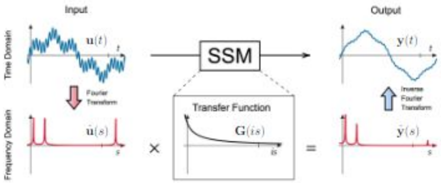
to model long sequential data. Compare to MLPs that usually takes high-dimensional inputs, the unidimensional time domain maintains a clear notion of frequency. Empirically, we observe frequency bias of SSMs.



A Frequency Perspective of SSMs

Fourier domain gives us a useful way to view the action of an SSM:

$$\hat{\mathbf{y}}(s) = \mathbf{G}(is)\hat{\mathbf{u}}(s), \quad \mathbf{G}(is) = \mathbf{C}(is\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}.$$

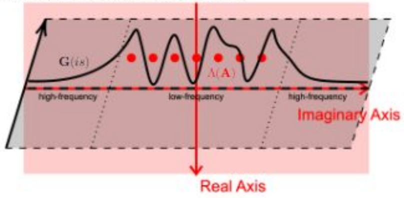


Why Do SSMs Have Frequency Bias?

So, why do SSMs have frequency bias? If we take a closer look at the transfer function $\mathbf{G}(is)$, then we have

$$\mathbf{G}(is) = [c_1\ c_2 \cdots c_n]\left(is\mathbf{I} - \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix}\right)^{-1}\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} + \mathbf{D} = \sum_{j=1}^{n} \frac{b_j c_j}{is - a_j} + \mathbf{D}.$$

Hence, $\mathbf{G}$ is a rational function with poles at $\Lambda(\mathbf{A})$. The more poles we have in a region, the better we can learn those frequencies.



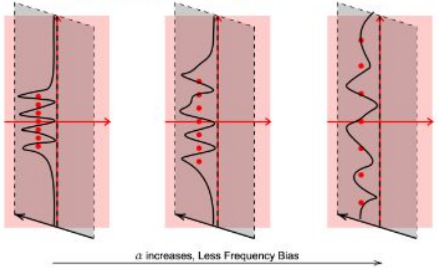
We have identified two sources of frequency bias:

- **Initialization:** When we initialize the matrix $\mathbf{A}$, we place its poles in the low-frequency region, introducing an inborn frequency bias.
- **Training:** We can show that, during training, an eigenvalue $a_j \in \Lambda(\mathbf{A})$ is mostly affected by the local frequency losses near $s = a_j$.

> The gradient of a generic loss $\mathcal{L}$ with respect to $\text{Im}(a_j)$ satisfies
> $$\frac{\partial \mathcal{L}}{\partial \text{Im}(a_j)} = \int_{-\infty}^{\infty} \frac{\partial \mathcal{L}}{\partial \mathbf{G}(is)} \cdot K_j(s)\, ds, \qquad |K_j(s)| = \mathcal{O}\left(|s - \text{Im}(a_j)|^{-2}\right).$$

Tuning Frequency Bias of SSMs via Initialization

The first way to tune the frequency bias is by scaling the initialization. We multiply a hyperparameter $\alpha \geq 0$ to the imaginary parts of $\Lambda(\mathbf{A})$. The larger the α, the more poles we place in the high-frequency region, and the less frequency bias we will get.



Tuning Frequency Bias of SSMs via Training

Another way to tune the frequency bias is by changing the training dynamics. Instead of applying the LTI system naively, we first scale the transfer function:

$$\hat{\mathbf{y}}(s) = (1 + |s|)^{\beta}\mathbf{G}(is)\hat{\mathbf{u}}(s), \quad \mathbf{G}(is) = \mathbf{C}(is\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D},$$

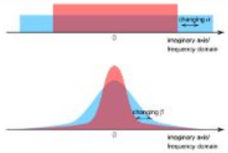
where $\beta \in \mathbb{R}$ is a hyperparameter. With the new system, we can change the sensitivity of the gradient to the high-frequency losses.

> The gradient of a generic loss $\mathcal{L}$ with respect to $\text{Im}(a_j)$ satisfies
> $$\frac{\partial \mathcal{L}}{\partial \text{Im}(a_j)} = \int_{-\infty}^{\infty} \frac{\partial \mathcal{L}}{\partial \mathbf{G}(is)} \cdot K_j^{(\beta)}(s)\, ds, \qquad |K_j^{(\beta)}(s)| = \mathcal{O}\left((1 + |s|)^{\beta}|s - \text{Im}(a_j)|^{-2}\right).$$

- If $\beta < 0$, the gradient of $\mathcal{L}$ with respect to $\text{Im}(a_j)$ is less sensitive to high-frequency losses, enhancing frequency bias.
- If $\beta > 0$, the gradient of $\mathcal{L}$ with respect to $\text{Im}(a_j)$ is more sensitive to high-frequency losses, reducing frequency bias.

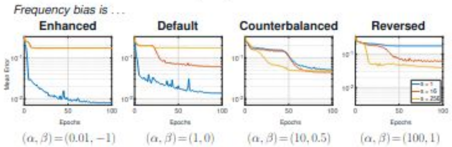
Comparing the Two Tuning Mechanisms

Changing the initialization serves as a "hard tuning strategy" that marks out the regions in the frequency domain that can be learned by an SSM; rescaling the transfer function is a "soft tuning strategy" that reweighs each location in the frequency domain.



Experiments and Discussion

Using α and β, we can tune the frequency bias of SSMs.



Tuning frequency bias also improves the performance of SSMs on long-range sequential tasks. By carefully selecting α and β, we achieve state-of-the-art performance on Long-Range Arena benchmark tasks with an S4D model.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.
DSS	57.60	76.60	87.60	85.80	84.10	85.00	79.45
S4++	57.30	86.28	84.82	82.91	80.24	-	-
Reg. S4D	61.48	88.19	91.25	88.12	94.93	95.63	86.60
Spectral SSM	60.33	89.60	90.00	-	95.60	90.10	-
Liquid S4	**62.75**	89.02	91.20	89.50	94.80	96.66	87.32
S5	62.15	89.31	91.40	88.00	95.33	**98.58**	87.46
S4	59.60	86.82	90.90	88.65	94.20	96.35	86.09
S4D	60.47	86.18	89.46	88.19	93.06	91.95	84.89
Ours	**62.75**	**89.76**	**92.45**	**90.89**	**95.89**	97.84	**88.26**

History

On the Spectral Bias of Neural Networks

Nasim Rahaman [*1,2] **Aristide Baratin** [*1] **Devansh Arpit** [1] **Felix Draxler** [2] **Min Lin** [1] **Fred A. Hamprecht** [2]
Yoshua Bengio [1] **Aaron Courville** [1]

[Paper]

- Deep ReLU networks naturally prefer learning low-frequency components of a function before high-frequency ones during gradient-based training.

Overview

- SSMs too exhibit an implicit bias toward capturing low-frequency components more effectively than high-frequency ones.
- Initialization of an SSM assigns it an innate frequency bias and conventional training does not change this bias.
- Tuning frequency bias:
    - Scale the initialization (tune the inborn frequency bias)
    - Sobolev-norm-based filter (change frequency bias via training)
- We can strengthen, weaken, or even reverse the frequency bias using the above methods.
- Improved performance on Long-Range Arena (LRA) benchmark tasks.

Images are from the paper and poster.

Introduction

- LTI system:

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t),$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t),$$

- Time domain:

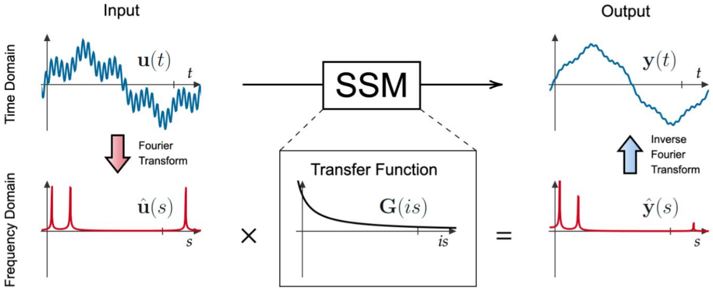
$$\mathbf{y}(t) = (\mathbf{h} * \mathbf{u} + \mathbf{D}\mathbf{u})(t) = \int_{-\infty}^{\infty} \mathbf{h}(t-\tau)\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t), \qquad \mathbf{h}(t) = \mathbf{C}\exp(t\mathbf{A})\mathbf{B}.$$

- Frequency domain:

$$\hat{\mathbf{y}}(s) = \mathbf{G}(is)\hat{\mathbf{u}}(s), \qquad \mathbf{G}(is) := \mathbf{C}(is\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}, \quad s \in \mathbb{R},$$

- We are interested in the frequency domain.

Frequency Perspective of SSMs



- Frequency domain math:

$$\mathbf{x}'(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t)$$

$$\mathcal{F}\{\mathbf{x}'(t)\} = is\,\hat{\mathbf{x}}(s)$$

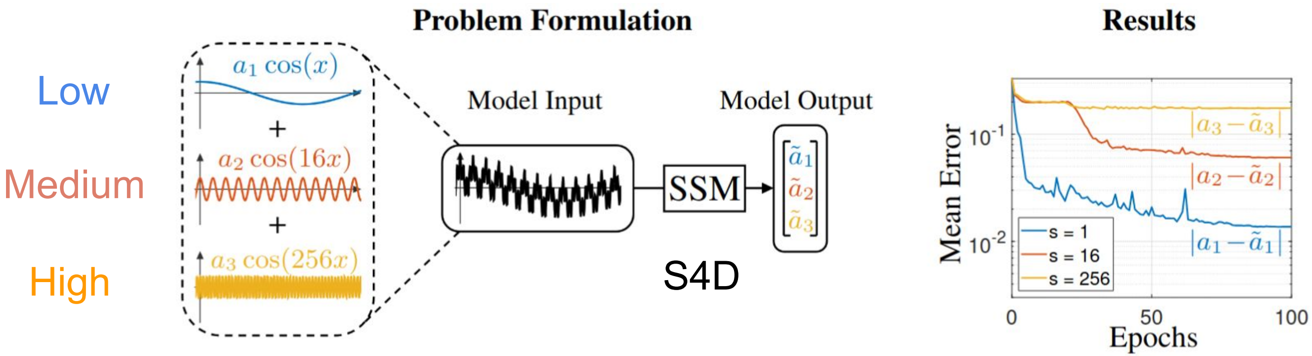
$$is\,\hat{\mathbf{x}}(s) = \mathbf{A}\hat{\mathbf{x}}(s) + \mathbf{B}\hat{\mathbf{u}}(s)$$
$$(is\mathbf{I} - \mathbf{A})\,\hat{\mathbf{x}}(s) = \mathbf{B}\hat{\mathbf{u}}(s)$$
$$\hat{\mathbf{x}}(s) = (is\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,\hat{\mathbf{u}}(s)$$

$$\hat{\mathbf{y}}(s) = \mathbf{C}\hat{\mathbf{x}}(s) + \mathbf{D}\hat{\mathbf{u}}(s)$$
$$\hat{\mathbf{y}}(s) = \mathbf{C}(is\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,\hat{\mathbf{u}}(s) + \mathbf{D}\hat{\mathbf{u}}(s)$$
$$\hat{\mathbf{y}}(s) = \boxed{\left[\mathbf{C}(is\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\right]}\hat{\mathbf{u}}(s)$$

$\mathbf{G}(is)$: Transfer function

$$\hat{\mathbf{y}}(s) = \mathbf{G}(is)\hat{\mathbf{u}}(s)$$

Frequency Bias



Low frequencies are approximated much better.

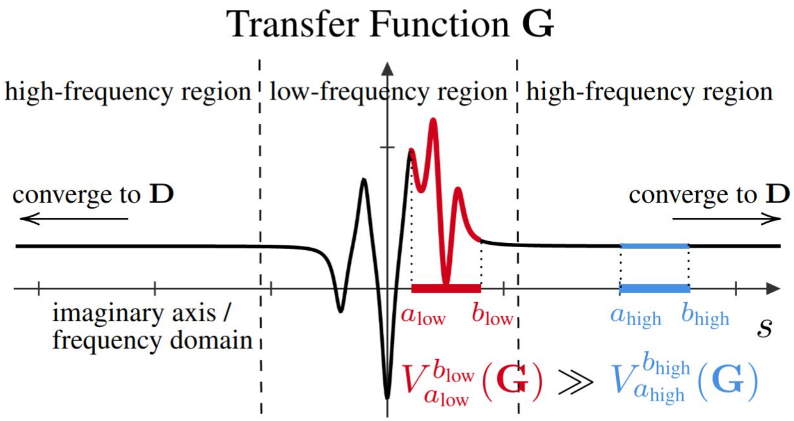
- Spectrum of $\mathbf{A}$ is related to the SSM's capability of processing high-frequency signals.
  - Popular initialization schemes (like HiPPO) place the spectrum in low-frequency region in the *s*-plane.
  - If an eigenvalue a_j is initialized in low-frequency region, then its gradient is insensitive to the loss induced by the high-frequency input content.

More on this later…

Frequency Bias

- What is frequency bias exactly?

Frequency bias of an SSM means that the frequency responses (i.e., the transfer functions $\mathbf{G}$*) of LTI systems have more variation in the low-frequency area than the high-frequency area.*



$V_a^b(\mathbf{G})$ is larger when [a, b] is near the origin.

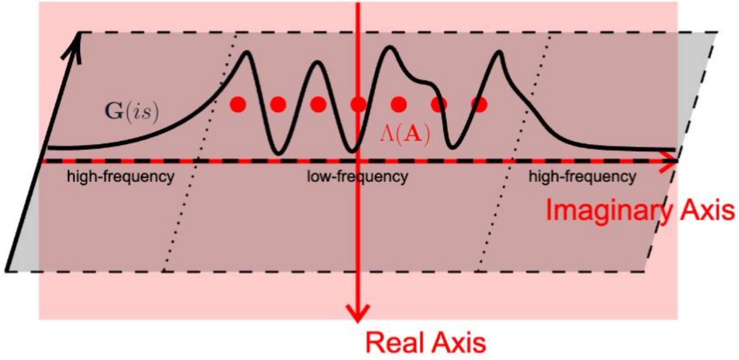
$V_a^b(\mathbf{G})$: total change of $\mathbf{G}(is)$ when s moves from a to b.

$$\mathbf{G}(is) = \frac{c_1}{is - a_1} + \cdots + \frac{c_n}{is - a_n} + \mathbf{D} = \frac{\xi_1 + i\zeta_1}{-v_1 + i(s - w_1)} + \cdots + \frac{\xi_n + i\zeta_n}{-v_n + i(s - w_n)} + \mathbf{D}.$$

$$V_a^b(\mathbf{G}) := \sup_{\substack{a = s_0 < s_1 < \cdots < s_N = b, \\ N \in \mathbb{N}}} \sum_{j=1}^{N} |\mathbf{G}(is_j) - \mathbf{G}(is_{j-1})| = \int_a^b \left| \frac{d\mathbf{G}(is)}{ds} \right| ds, \quad -\infty \le a < b \le \infty.$$

Frequency Bias

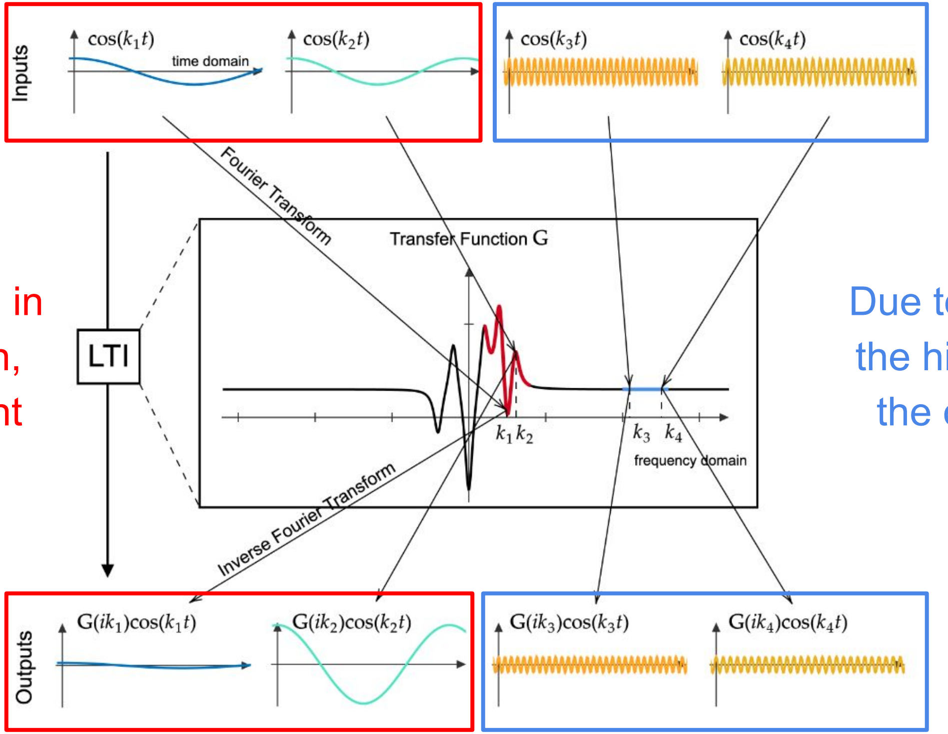
$$\mathbf{G}(is) = [c_1\ c_2 \cdots c_n] \left(is\mathbf{I} - \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix} \right)^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} + \mathbf{D} = \sum_{j=1}^{n} \frac{b_j c_j}{is - a_j} + \mathbf{D}$$



- The poles of $\mathbf{G}(is)$ are at $\Lambda(\mathbf{A})$.
- The more poles we have in a region, the better we can learn those frequencies. Why?
- Greater variation means stronger sensitivity to parameter changes (larger derivative), which in turn allows the model to better fit and learn signals at those frequencies.

Frequency Bias

Due to high variation in G in the low-frequency region, the outputs have different pass rates.



Due to low variation in G in the high-frequency region, the outputs have similar pass rates.

Frequency Bias at Initialization

- Notation: $\mathbf{B} \circ \mathbf{C}^\top = [c_1 \quad \cdots \quad c_n]^\top \in \mathbb{C}^n$ $\quad a_j = v_j + iw_j \quad c_j = \xi_j + i\zeta_j$

$$\mathbf{G}(is) = \frac{c_1}{is - a_1} + \cdots + \frac{c_n}{is - a_n} + \mathbf{D} = \frac{\xi_1 + i\zeta_1}{-v_1 + i(s - w_1)} + \cdots + \frac{\xi_n + i\zeta_n}{-v_n + i(s - w_n)} + \mathbf{D}.$$

- In most cases the input **u**(t) is real-valued. To make sure output is also real-valued:

$$\tilde{\mathbf{G}}(is) := \mathrm{Re}(\mathbf{G}(is)) = \sum_{j=1}^{n} \frac{\zeta_j(s - w_j) - \xi_j v_j}{v_j^2 + (s - w_j)^2} + \mathbf{D}.$$

- **Lemma 1.** Let $\tilde{\mathbf{G}}$ be the transfer function defined in eq. (3). Given any $B > \max_j |w_j|$, we have

$$V_{-\infty}^{-B}(\tilde{\mathbf{G}}) \leq \sum_{j=1}^{n} \frac{|c_j|}{|w_j + B|}, \qquad V_B^{\infty}(\tilde{\mathbf{G}}) \leq \sum_{j=1}^{n} \frac{|c_j|}{|w_j - B|}.$$

If the imaginary parts of a_j are distributed in the low-frequency region, i.e., $|w_j|$ are small, the transfer function has a small total variation in the high-frequency areas $(-\infty, -B]$ and $[B, \infty)$ as $B \to \infty$, inducing a frequency bias of the SSM.

Frequency Bias at Initialization

- When we initialize the matrix $\mathbf{A}$, we place its poles in the low-frequency region, introducing an inborn frequency bias.
- This is what happens with HiPPO initialization.

Corollary 1. Assume that $a_j = -0.5 + i(-1)^j \lfloor j/2 \rfloor \pi$ and $\xi_j, \zeta_j \sim \mathcal{N}(0,1)$ i.i.d., where $\mathcal{N}(0,1)$ is the standard normal distribution. Then, given $B > n\pi/2$ and $\delta > 0$, we have

$$V_{-\infty}^{-B}(\tilde{\mathbf{G}}), V_B^{\infty}(\tilde{\mathbf{G}}) \leq \frac{\sqrt{2n}(\sqrt{n} + \sqrt{\ln(1/\delta)})}{B - n/2} \qquad \text{with probability} \geq 1 - \delta.$$

In particular, Corollary 1 tells us that the HiPPO initialization only captures the frequencies $s \in [-B, B]$ up to $B = \mathcal{O}(n)$, because when $B = \omega(n)$, we see that $V_{-\infty}^{-B}(\tilde{\mathbf{G}}), V_B^{\infty}(\tilde{\mathbf{G}})$ vanish as n increases. This means that no complicated high-frequency responses can be learned.

- If $B = \omega(n)$ (i.e., B grows faster than n), $B - n/2 \to \infty$.

Frequency Bias during Training

- During training, an eigenvalue $a_j \in \Lambda(\mathbf{A})$ is mostly affected by the local frequency losses near $s = a_j$.

The gradient of a generic loss $\mathcal{L}$ with respect to Im(a_j) satisfies

$$\frac{\partial \mathcal{L}}{\partial \mathrm{Im}(a_j)} = \int_{-\infty}^{\infty} \frac{\partial \mathcal{L}}{\partial \mathbf{G}(is)} \cdot K_j(s)\, ds, \qquad |K_j(s)| = \mathcal{O}\left(|s - \mathrm{Im}(a_j)|^{-2}\right).$$

- The factor $K_j(s)$ tells us:

$$K_j(s) := \frac{\zeta_j((s-w_j)^2 - v_j^2) - 2\xi_j v_j (s - w_j)}{[v_j^2 + (s-w_j)^2]^2},$$

The gradient of $\mathcal{L}$ with respect to w_j highly depends on the part of the loss that has "local" frequencies near $s = w_j$. It is relatively unresponsive to the loss induced by high frequencies, with a decaying factor of $\mathcal{O}(|s|^{-2})$ as the frequency increases, i.e., as $|s| \to \infty$.

- The loss landscape of the frequency domain contains many local minima, and an LTI system can rarely learn the high frequencies with the usual training.

Frequency Bias during Training

- They train an S4D initialized by HiPPO to learn sCIFAR-10 for 100 epochs, and measure the relative change for each parameter.

Table 1: The average relative change of each LTI system matrix in an S4D model trained on the sCIFAR-10 task. We see that the imaginary parts of $\mathrm{diag}(\mathbf{A})$ are almost unchanged during training.

Parameter	$\mathrm{Re}(\mathrm{diag}(\mathbf{A}))$	$\mathrm{Im}(\mathrm{diag}(\mathbf{A}))$	$\mathbf{B} \circ \mathbf{C}^\top$	$\mathbf{D}$
Δ	1002.705	0.0143	1.1801	0.8913

- The imaginary part is trained very little, as it is easily trapped by a low-frequency local minimum.

A Synthetic Example

- Target function:

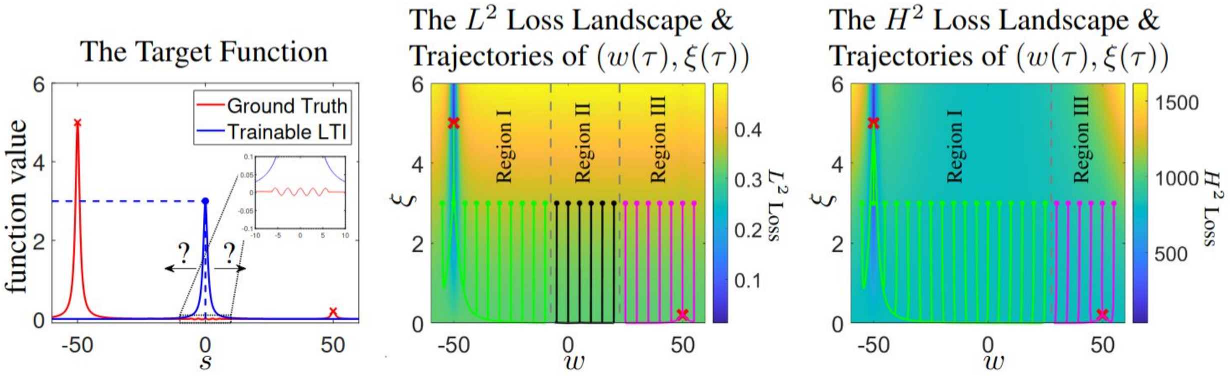
mode1 mode2 noise

$$\tilde{\mathbf{F}}(is) = \mathrm{Re}\left(\frac{5}{is-(-1-50i)} + \frac{0.2}{is-(-1+50i)} + 0.01\cos\left(\frac{9}{4}s\right)\cdot \mathbb{1}_{[-2\pi,2\pi]}\right), \qquad s \in \mathbb{R},$$

- Transfer function ($v = -1$ and $\zeta = 0$):

unimodal

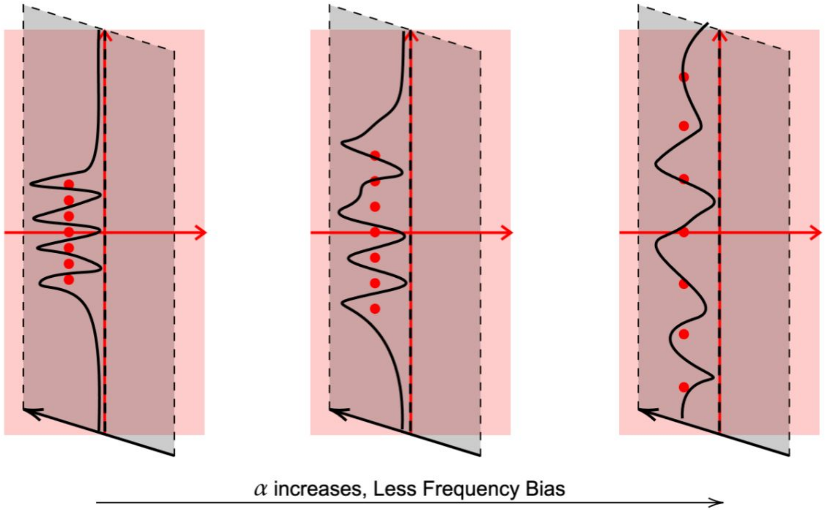
$$\tilde{\mathbf{G}}(is) = \mathrm{Re}\left(\frac{\xi}{is-(-1-wi)}\right), \qquad s \in \mathbb{R},$$



Region II: once *w* enters the noisy region, it gets stuck in local minimum and never converges to one of the two modes.

Tuning Frequency Bias via Initialization

- We tune by scaling the initializations. We multiply a hyperparameter $\alpha \geq 0$ to $\text{Im}(\Lambda(\mathbf{A}))$.
- The larger the α, the more poles we place in the high-frequency region, and the less frequency bias we will get.



Tuning Frequency Bias via Training

- We tune by changing the training dynamics.

Sobolev-norm-based filter

$$\hat{\mathbf{y}}(s) = (1+|s|)^{\beta}\mathbf{G}(is)\hat{\mathbf{u}}(s), \quad \mathbf{G}(is) = \mathbf{C}(is\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}+\mathbf{D},$$

- With this, we can change the sensitivity of the gradient to high frequency losses.

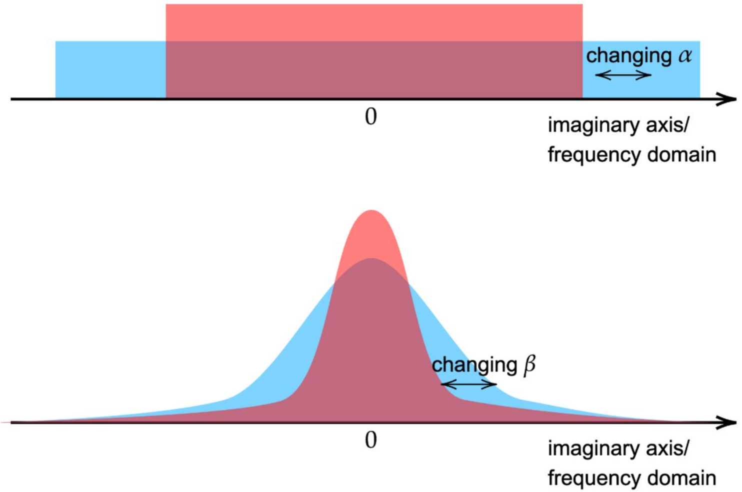
The gradient of a generic loss $\mathcal{L}$ with respect to $\text{Im}(a_j)$ satisfies

$$\frac{\partial \mathcal{L}}{\partial \text{Im}(a_j)} = \int_{-\infty}^{\infty} \frac{\partial \mathcal{L}}{\partial \mathbf{G}(is)} \cdot K_j^{(\beta)}(s)\, ds, \qquad |K_j^{(\beta)}(s)| = \mathcal{O}\left((1+|s|)^{\beta}|s-\text{Im}(a_j)|^{-2}\right).$$

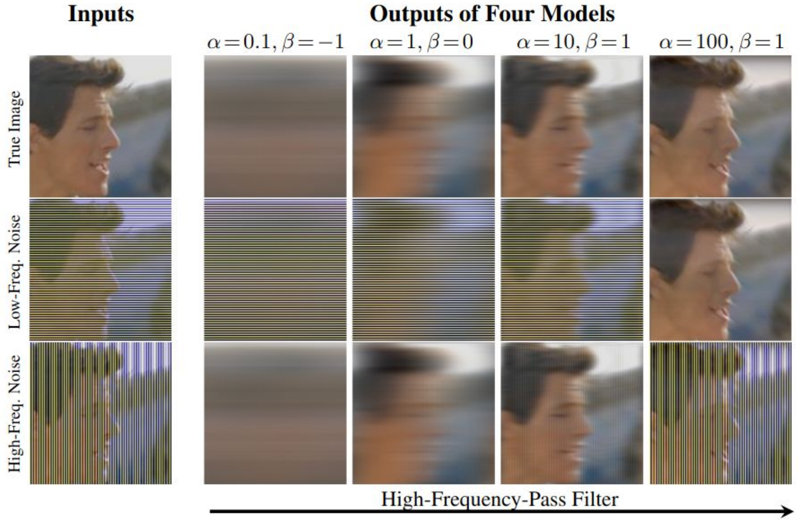
- If $\beta < 0$, the gradient is less sensitive to high-frequency losses, thus enhancing frequency bias.
- If $\beta > 0$, the gradient is more sensitive to high-frequency losses, thus reducing frequency bias.

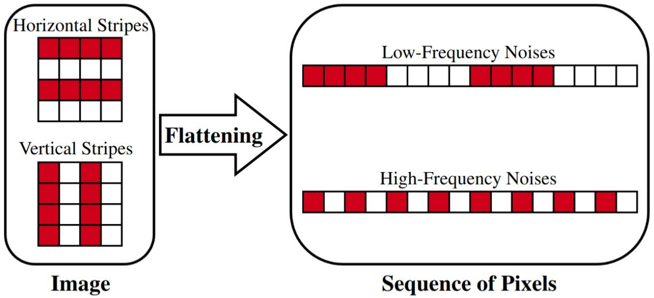
Comparison b/w Tuning Methods

- α: Hard tuning strategy - marks out the regions in the frequency domain that can be learned by an SSM.
- β: Soft tuning strategy - re-weighs each location in the frequency domain.

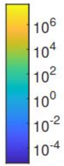


Experiments(Autoencoder)





		β				
		−1.0	−0.5	0.0	0.5	1.0
α	0.1	4.463e+07	2.409e+06	1.198e+05	4.613e+03	1.738e+02
	1	4.912e+05	2.124e+05	1.758e+04	9.595e+02	5.730e+01
	10	9.654e+04	7.465e+03	6.073e+02	5.699e+01	6.394e+00
	100	3.243e+00	3.745e−02	3.801e−03	7.299e−05	5.963e−06



(LPR/HPR) decreases as α and β increase.

- As α and β increase, model learns high frequencies better.

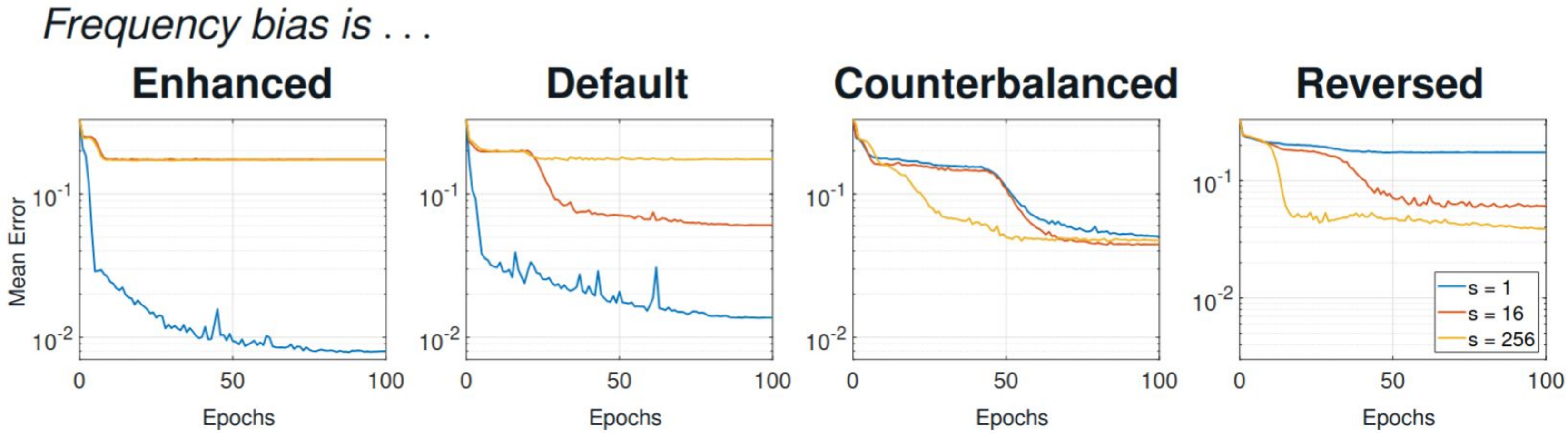
Experiments(Long-Range Arena)

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.
DSS (Gupta et al., 2022)	57.60	76.60	87.60	85.80	84.10	85.00	79.45
S4++ (Qi et al., 2024)	57.30	86.28	84.82	82.91	80.24	-	-
Reg. S4D (Liu & Li, 2024a)	61.48	88.19	91.25	88.12	94.93	95.63	86.60
Spectral SSM (Agarwal et al., 2023)	60.33	<u>89.60</u>	90.00	-	<u>95.60</u>	90.10	-
Liquid S4 (Hasani et al., 2023)	**62.75**	89.02	91.20	<u>89.50</u>	94.80	96.66	87.32
S5 (Smith et al., 2023)	62.15	89.31	<u>91.40</u>	88.00	95.33	**98.58**	<u>87.46</u>
S4 (Gu et al., 2022b)	59.60	86.82	90.90	88.65	94.20	96.35	86.09
S4D (Gu et al., 2022a)	60.47	86.18	89.46	88.19	93.06	91.95	84.89
Ours	**62.75** ± 0.78	**89.76** ± 0.22	**92.45** ± 0.16	**90.89** ± 0.35	**95.89** ± 0.13	<u>97.84</u> ± 0.21	**88.26**

Task	Depth	#Features	Norm	Prenorm	α	LR	BS	Epochs	WD	Δ Range
ListOps	8	256	BN	False	3	0.002	50	80	0.05	(1e-3,1e0)
Text	6	256	BN	True	5	0.01	32	300	0.05	(1e-3,1e-1)
Retrieval	6	128	BN	True	3	0.004	64	40	0.03	(1e-3,1e-1)
Image	6	512	LN	False	3	0.01	50	1000	0.01	(1e-3,1e-1)
Pathfinder	6	256	BN	True	3	0.004	64	300	0.03	(1e-3,1e-1)
Path-X	6	128	BN	True	5	0.001	20	80	0.03	(1e-4,1e-1)

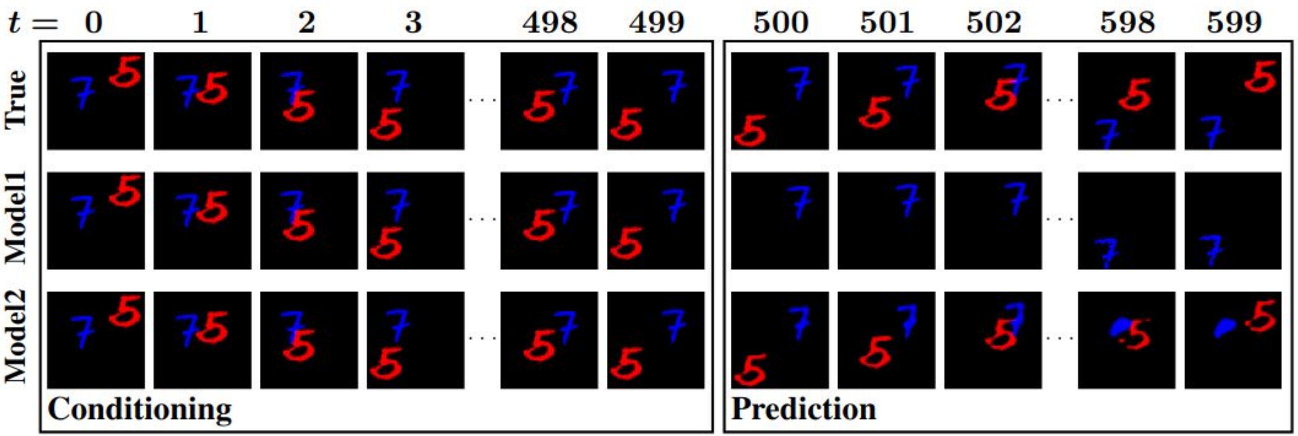
β is trained.

Experiments(Magnitudes of Waves)



$(\alpha, \beta) = (0.01, -1)$ $(\alpha, \beta) = (1, 0)$ $(\alpha, \beta) = (10, 0.5)$ $(\alpha, \beta) = (100, 1)$

Experiments(MNIST Video Prediction)



HiPPO
unable to predict 5

Tuned

Speed of RED is 10x faster than BLUE.
High frequency Low frequency