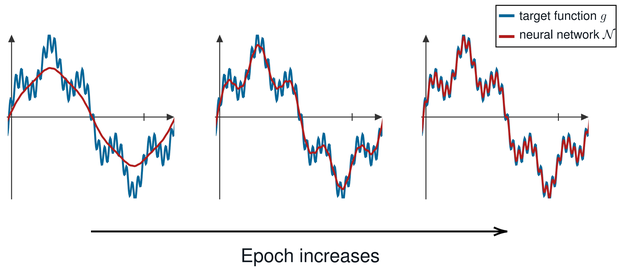
Tuning Frequency Bias of State Space Models

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.



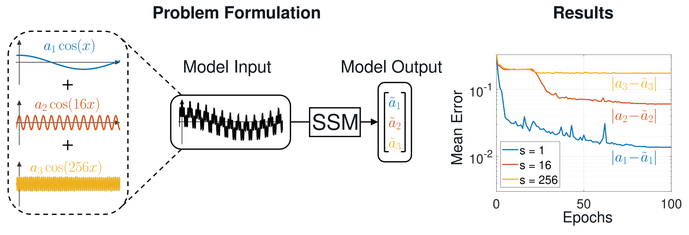
Frequency bias is a double-edged sword: it partially explains the good generalization capability of deep learning models but also puts a curse on learning the useful high-frequency information in the target.

State Space Models (SSMs)

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

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

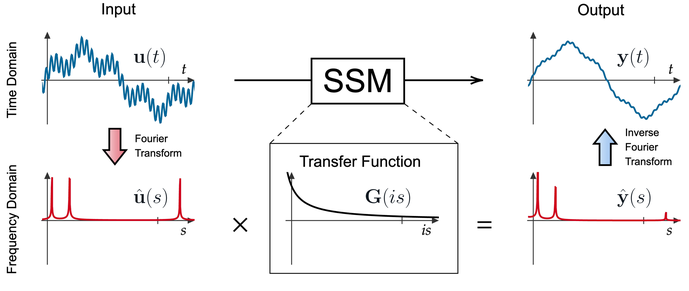
to model long sequential data. Compare to MLPs that usually takes high-dimensional inputs, the unidimensional time domain maintains a clear notion of frequency. Empirically, we observe frequency bias of SSMs.



A Frequency Perspective of SSMs

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

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

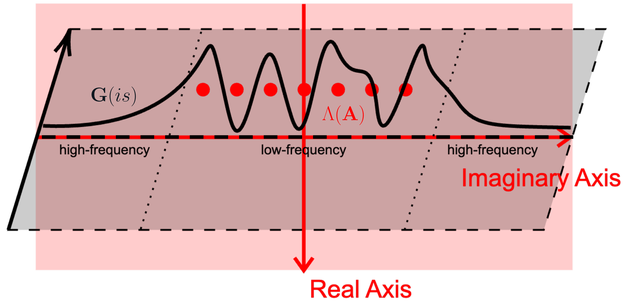


Why Do SSMs Have Frequency Bias?

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

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

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:

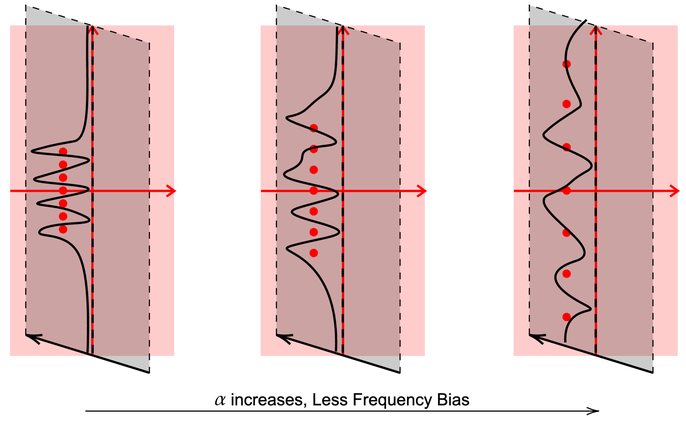
- **Initialization:** When we initialize the matrix $\mathbf{A}$, we place its poles in the low-frequency region, introducing an inborn frequency bias.
- **Training:** We can show that, during training, an eigenvalue $a_j \in \Lambda(\mathbf{A})$ is mostly affected by the local frequency losses near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

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

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

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.

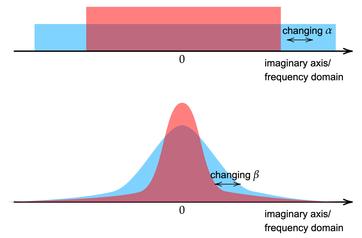
The gradient of a generic loss $\mathcal{L}$ with respect to $\mathrm{Im}(a_j)$ satisfies

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

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

Comparing the Two Tuning Mechanisms

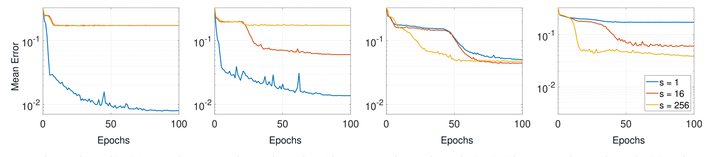
Changing the initialization serves as a "hard tuning strategy" that marks out the regions in the frequency domain that can be learned by an SSM; rescaling the transfer function is a "soft tuning strategy" that reweighs each location in the frequency domain.



Experiments and Discussion

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

Frequency bias is . . .



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	<u>89.60</u>	90.00	-	<u>95.60</u>	90.10	-
Liquid S4	**62.75**	89.02	91.20	<u>89.50</u>	94.80	96.66	87.32
S5	62.15	89.31	<u>91.40</u>	88.00	95.33	**98.58**	<u>87.46</u>
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**	<u>97.84</u>	**88.26**