A Method Details

Listing 1 Full Numpy example of the parameterization and computation of a 1D S4D-Lin model

```
def parameters(N, dt_min=1e-3, dt_max=1e-1):
  # Initialization
  log_dt = np.random.rand() * (np.log(dt_max)-np.log(dt_min)) + np.log(dt_min)
  A = -0.5 + 1j * np.pi * np.arange(N//2) # S4D-Lin initialization
  B = np.ones(N//2) + 0j
  C = np.random.randn(N//2) + 1j * np.random.randn(N) # Variance preserving initialization
  return log_dt, np.log(-A.real), A.imag, B, C

def kernel(L, log_dt, log_A_real, A_imag, B, C):
  # Discretization (e.g. bilinear transform)
  dt, A = np.exp(log_dt), -np.exp(log_A_real) + 1j * A_imag
  dA, dB = (1+dt*A/2) / (1-dt*A/2), dt*B / (1-dt*A/2)

  # Computation (Vandermonde matrix multiplication - can be optimized)
  # Return twice the real part - same as adding conjugate pairs
  return 2 * ((B*C) @ (dA[:, None] ** np.arange(L))).real

def forward(u, parameters):
    L = u.shape[-1]
    K = kernel(L, *parameters)
    # Convolve y = u * K using FFT
    K_f, u_f = np.fft.fft(K, n=2*L), np.fft.fft(u, n=2*L)
    return np.fft.ifft(K_f*u_f, n=2*L)[..., :L]
```

A.1 Proofs

We prove Theorem 3, and then show why this it is a surprising result that is not true in general to low-rank perturbations of SSMs.

We start with the interpretation of the S4-LegS matrix shown in [10], which corresponds to Fig. 1 (Left).

Theorem 5. *Let $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{P}$ be the matrices defined in equation* (4). *The SSM kernels $K_n(t) = \boldsymbol{e}_n^\top e^{t\boldsymbol{A}}\boldsymbol{B}$ have the closed form formula*

$$K_n(t) = L_n(e^{-t})e^{-t}$$

where L_n are the Legendre polynomials shifted and scaled to be orthonormal on the interval $[0, 1]$.

Lemma A.1. *The functions $L_n(e^{-t})$ are a complete orthonormal basis with respect to the measure $\omega(t) = e^{-t}$.*

Proof. The polynomials are defined to be orthonormal on $[0, 1]$, i.e.

$$\int_0^1 L_n(t)L_m(t)\,dt = \delta_{n,m}.$$

By the change of variables $t = e^{-s}$ with $\frac{dt}{ds} = -e^{-s}$,

$$-\int_{-\infty}^{0} L_n(e^{-s})L_m(e^{-s})e^{-s}\,ds = \delta_{n,m} = \int_0^{\infty} L_n(e^{-s})L_m(e^{-s})e^{-s}\,ds$$

which shows the orthonormality.

Completeness follows from the fact that polynomials are complete. □

Proof of Theorem 3. We start with the standard interpretation of SSMs as convolutional systems. The SSM $x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$ is equivalent to the convolution

$$x_n(t) = (u * K_n)(t) = \int_{-\infty}^{t} u(s)K_n(t-s)\,ds = \int_0^{\infty} u(t-s)K_n(s)\,ds$$

for the SSM kernels (equation (3)).

Defining $u^{(t)}(s) = u(t-s)$, we can write this as

$$x_n(t) = \langle u^{(t)}, K_n \rangle_\omega$$

where $\omega(s) = e^{-s}$ and $\langle p(s), q(s) \rangle_\omega = \int_0^\infty p(s)q(s)\omega(s)\, ds$ is the inner product in the Hilbert space of L2 functions with respect to measure ω.

By Theorem 5, the K_n are a complete orthonormal basis in this Hilbert space. There $x_n(t)$ represents a decomposition of the function $u^{(t)}$ with respect to this basis, and can be recovered as a linear combination of these projections

$$u^{(t)} = \sum_{n=0}^{\infty} x_n(t) K_n.$$

Pointwise over the inner times s,

$$u^{(t)}(s) = \sum_{n=0}^{\infty} x_n(t) K_n(s).$$

This implies that

$$\begin{aligned} u(t) = u^{(t)}(0) &= \sum_{n=0}^{\infty} x_n(t) K_n(0) \\ &= \sum_{n=0}^{\infty} x_n(t) L_n(0) = \sum_{n=0}^{\infty} x_n(t)(2n+1)^{\frac{1}{2}} \\ &= \boldsymbol{B}^\top x(t) \end{aligned}$$

Intuitively, due to the function reconstruction interpretation of HiPPO [10], we can approximate $u(t)$ using knowledge in the current state $x(t)$. There in the limit $N \to \infty$, the original SSM is equivalent to

$$\begin{aligned} x'(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\ &= \boldsymbol{A}x(t) + \frac{1}{2}\boldsymbol{B}u(t) + \frac{1}{2}\boldsymbol{B}u(t) \\ &= \boldsymbol{A}x(t) + \frac{1}{2}\boldsymbol{B}\boldsymbol{B}^\top x(t) + \frac{1}{2}\boldsymbol{B}u(t) \\ &= \boldsymbol{A}x(t) + \boldsymbol{P}\boldsymbol{P}^\top x(t) + \frac{1}{2}\boldsymbol{B}u(t) \\ &= \boldsymbol{A}^N x(t) + \frac{1}{2}\boldsymbol{B}u(t) \end{aligned}$$

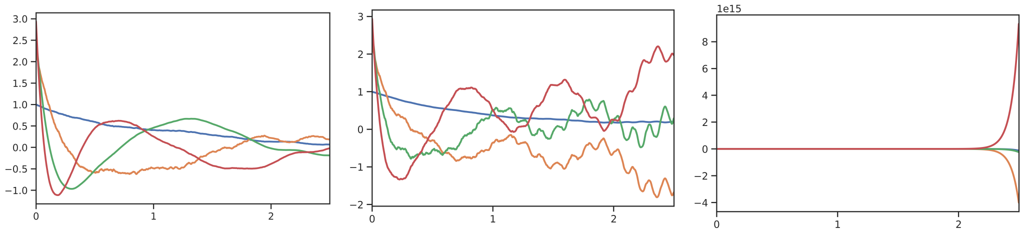
□

General low-rank perturbations. Finally, we remark that this phenomenon where removing the low-rank correction to a DPLR matrix approximates the original dynamics, is unique to this HiPPO-LegS matrix. We note that if instead of $\boldsymbol{P}\boldsymbol{P}^\top$, a *random* rank-1 correction is added to the HiPPO-LegS matrix in Theorem 3, the resulting SSM kernels look completely different and in fact diverge rapidly as the magnitude of $\boldsymbol{P}$ increases (Fig. 4).

Similarly, Fig. 5a shows a new S4 variant called S4-FouT that is also DPLR [10], but removing the low-rank component dramatically changes the SSM kernels.

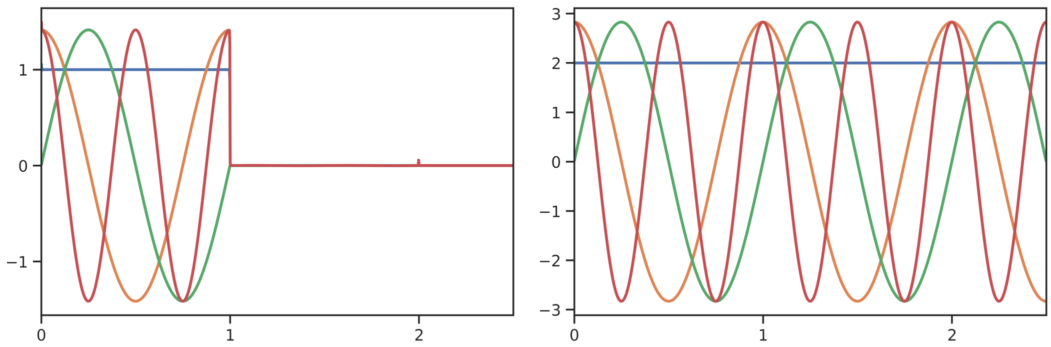
B Experiment Details

Ablation datasets training protocol. The architecture has 4 layers and hidden dimension $H = 128$, resulting in around 100K trainable parameters. The $\boldsymbol{A}$ and $\boldsymbol{B}$ parameters were tied across the H SSM copies; therefore the S4 models have only $H \times \{\text{num. layers}\}$ more parameters than S4D models, arising from the $\boldsymbol{P}$ tensor in the DPLR representation $\boldsymbol{A} = \boldsymbol{\Lambda} - \boldsymbol{P}\boldsymbol{P}^\top$. This choice was made



(a) $\sigma = 0.3$ (b) $\sigma = 0.4$ (c) $\sigma = 0.5$

Figure 4: Basis kernels for $(\boldsymbol{A}+\boldsymbol{P}\boldsymbol{P}^\top, \boldsymbol{B})$ for HiPPO-LegS $(\boldsymbol{A}, \boldsymbol{B})$ and random i.i.d. Gaussian $\boldsymbol{P}$ with varying std σ, illustrating that the SSM basis is very sensitive to low-rank perturbations. Note that the normal-HiPPO matrix $\boldsymbol{A}^{(N)} = \boldsymbol{A}+\boldsymbol{P}\boldsymbol{P}^\top$ for $\boldsymbol{P}$ with entries of magnitude $N^{\frac{1}{2}}$ which is far larger, highlighting how unexpected the theoretical result Theorem 3 is.



(a) S4-FouT (b) Diagonal approximation to S4-FouT

Figure 5: (a) S4-FouT is a version of S4 that produces *truncated Fourier* basis functions choosing a particular $(\boldsymbol{A}, \boldsymbol{B})$. This captures sliding Fourier transforms as a state space model. (b) Removing the low-rank term from the FouT matrix does *not* approximate S4-FouT. This diagonal state matrix has real part 0 that produces infinite oscillations and does not perform well empirically.

because it generally does not affect performance much, while reducing parameter count and ensuring that S4 vs. S4D models have very similar numbers of parameters.

All results are averaged over 2 or 3 seeds.

All models use learning rate 0.004, 0.01 weight decay, and no other regularization or data augmentation. For the classification tasks (sCIFAR and SC). we use a cosine scheduler with 1 epoch warmup and decaying to 0. For the regression task (BIDMC), we use a multistep scheduler following [21, 8].

Reported results are all best validation accuracy, except for the large models in Table 4.

Full results for parameterization ablations. Table 6 and Table 7 contain the raw results for Table 2 including standard deviations.

Full results for large models on ablations datasets. Tables 8 to 10 show full results comparing our proposed methods against the best models from the literature; citations indicate numbers from prior work.

Table 6: Full results for Table 2 (Left) including standard deviations.

Trainable $\boldsymbol{B}$	Method	sCIFAR	SC (AR)	BIDMC (SpO2)
No	Softmax	85.04 (0.22)	89.80 (0.21)	0.1299 (0.0048)
No	Vandermonde	84.78 (0.16)	89.62 (0.03)	0.1355 (0.0039)
Yes	Softmax	85.37 (0.43)	90.06 (0.11)	0.1170 (0.0039)
Yes	Vandermonde	85.37 (0.43)	90.34 (0.18)	0.1274 (0.0020)

Table 7: Full results for Table 2 (Right) including standard deviations.

Discretization	Real part	sCIFAR	SC (AR)	BIDMC (SpO2)
Bilinear	Exp	85.20 (0.18)	89.52 (0.01)	0.1193 (0.0069)
Bilinear	-	85.35 (0.27)	90.58 (0.37)	0.1102 (0.0075)
Bilinear	ReLU	85.06 (0.06)	90.22 (0.25)	0.1172 (0.0063)
ZOH	Exp	85.02 (0.24)	89.93 (0.07)	0.1303 (0.0014)
ZOH	-	85.15 (0.13)	90.19 (0.58)	0.1289 (0.0035)
ZOH	ReLU	84.98 (0.72)	90.03 (0.13)	0.1232 (0.0065)

Table 8: (**Sequential CIFAR image classification.** Test accuracy (Std. dev.)

Model	sCIFAR
S4-LegS	**91.80** (0.43)
S4-FouT	91.22 (0.25)
S4-(LegS/FouT)	91.58 (0.17)
S4D-LegS	89.92 (1.69)
S4D-Inv	90.69 (0.06)
S4D-Lin	90.42 (0.03)
Transformer [25]	62.2
FlexConv [20]	80.82
TrellisNet [2]	73.42
LSTM [12, 7]	63.01
r-LSTM [25]	72.2
UR-GRU [7]	74.4
HiPPO-RNN [6]	61.1
LipschitzRNN [4]	64.2

Table 9: (**BIDMC Vital signs prediction.**) RMSE for predicting respiratory rate (RR), heart rate (HR), and blood oxygen (SpO2).

Model	HR	RR	SpO2
S4-LegS	**0.332** (0.013)	0.247 (0.062)	0.090 (0.006)
S4-FouT	0.339 (0.020)	0.301 (0.030)	**0.068** (0.003)
S4-(LegS/FouT)	0.344 (0.032)	**0.163** (0.008)	0.080 (0.007)
S4D-LegS	0.367 (0.001)	0.248 (0.036)	0.102 (0.001)
S4D-Inv	0.373 (0.024)	0.254 (0.022)	0.110 (0.001)
S4D-Lin	0.379 (0.006)	0.226 (0.008)	0.114 (0.003)
UnICORNN [21]	1.39	1.06	0.869
coRNN [21]	1.81	1.45	-
CKConv	2.05	1.214	1.051
NRDE [15]	2.97	1.49	1.29
LSTM	10.7	2.28	-
Transformer	12.2	2.61	3.02
XGBoost [23]	4.72	1.67	1.52
Random Forest [23]	5.69	1.85	1.74
Ridge Regress. [23]	17.3	3.86	4.16

Note that earlier works on the Speech Commands dataset typically use pre-processing such as MFCC features, or a 10-class subset of the full 35-class dataset [14, 19, 9]. As we are not aware of a collection of strong baselines for raw waveform classification using the full dataset, we trained several baselines from scratch for Table 10. The InceptionNet, ResNet-18, and XResNet-50 models are 1D adaptations from Nonaka and Seita [16] of popular CNN architectures for vision. The ConvNet architecture is a generic convolutional neural network that we tuned for strong performance, comprising:

- Four stages, each composed of three identical residual blocks.
- The first stage has model dimension (i.e. channels, in CNN nomenclature) $H = 64$. Each stage doubles the dimension of the previous stage (with a position-wise linear layer) and ends in an average pooling layer of width 4. Thus, the first stage operates on inputs of length 16384, dimension 64 (the input is zero-padded from 16000 to 16384) and the last on length 256, dimension 512.
- Each residual block has a (pre-norm) BatchNorm layer followed by a convolution layer and GeLU activation.
- Convolution layers have a kernel size of 25.

Long Range Arena. Our Long Range Arena experiments follow the same setup as the original S4 paper with some differences in model architecture and hyperparameters. The main global differences are as follows:

Bidirectional The original S4 layer is unidirectional or causal, which is an unnecessary constraint for the classification tasks appearing in LRA. Goel et al. [5] propose a bidirectional version of S4 that simply concatenates two S4 convolution kernels back-to-back. We use this for all tasks.

GLU feedforward S4 consists of H independent 1-dimensional SSMs, each of which are processed by an independent S4 SSM mapping $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$. These outputs are then mixed with a

Table 10: (**Speech Commands classification.**) Test accuracy on 35-way keyword spotting. Training examples are 1-second audio waveforms sampled at 16000Hz, or a 1-D sequence of length 16000. Last column indicates 0-shot testing at 8000Hz where examples are constructed by naive decimation.

Model	Parameters	16000Hz	8000Hz
S4-LegS	307K	96.08 (0.15)	91.32 (0.17)
S4-FouT	307K	95.27 (0.20)	91.59 (0.23)
S4-(LegS/FouT)	307K	95.32 (0.10)	90.72 (0.68)
S4D-LegS	306K	95.83 (0.14)	91.08 (0.16)
S4D-Inv	306K	96.18 (0.27)	**91.80** (0.24)
S4D-Lin	306K	**96.25** (0.03)	91.58 (0.33)
InceptionNet	481K	61.24 (0.69)	05.18 (0.07)
ResNet-18	216K	77.86 (0.24)	08.74 (0.57)
XResNet-50	904K	83.01 (0.48)	07.72 (0.39)
ConvNet	26.2M	95.51 (0.18)	07.26 (0.79)

Table 11: The values of the best hyperparameters found for all datasets; full models on ablation datasets (Top) and LRA (Bottom). LR is learning rate and WD is weight decay. BN and LN refer to Batch Normalization and Layer Normalization.

	Depth	**Features** H	**State Size** N	**Norm**	**Pre-norm**	**Dropout**	**LR**	**Batch Size**	**Epochs**	**WD**	$(\Delta_{min}, \Delta_{max})$
sCIFAR	6	512	64	LN	False	0.1	0.01	50	200	0.05	$(0.001, 0.1)$
SC	6	128	64	BN	True	0	0.01	16	40	0.05	$(0.001, 0.1)$
BIDMC	6	128	256	LN	True	0	0.01	32	500	0.05	$(0.001, 0.1)$
ListOps	8	128	64	BN	False	0	0.01	50	40	0.05	$(0.001, 0.1)$
Text	6	256	64	BN	True	0	0.01	16	32	0.05	$(0.001, 0.1)$
Retrieval	6	256	64	BN	True	0	0.01	64	20	0.05	$(0.001, 0.1)$
Image	6	512	64	LN	False	0.1	0.01	50	200	0.05	$(0.001, 0.1)$
Pathfinder	6	256	64	BN	True	0	0.004	64	200	0.03	$(0.001, 0.1)$
Path-X	6	256	64	BN	True	0	0.0005	32	50	0.05	$(0.0001, 0.01)$

position-wise linear layer, i.e. $\boldsymbol{W}y$ for a learned matrix $\boldsymbol{W} \in \mathbb{R}^{H \times H}$. Instead of this linear mapping, we use a GLU activation $(\boldsymbol{W}_1 y) \circ \sigma(\boldsymbol{W}_2 y)$ for $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{H \times H}$ [3]. These have been empirically found to improve linear layers of DNNs in general [22].

Cosine scheduler Instead of the plateau scheduler used in [9], we use a cosine annealing learning rate scheduler for all tasks.

Regularization Almost all tasks used no dropout and 0.05 weight decay.

Architecture Almost all tasks used an architecture with 6 layers, $H = 256$ hidden units, BatchNorm, pre-norm placement of the normalization layer.

Exceptions to the above rules are described below. Full hyperparameters are in Table 11.

sCIFAR / LRA Image. This dataset is grayscale sequential CIFAR-10, and the settings for this task were taken from S4's hyperparameters on the normal sequential CIFAR-10 task. In particular, this used LayerNorm [1] instead of BatchNorm [13], a larger number of hidden features H, post-norm instead of pre-norm, and minor dropout. We note that the choice of normalization and increased H do not make a significant difference on final performance, still attaining classification accuracy in the high 80's. Dropout does seem to make a difference.

BIDMC. We used a larger state size of $N = 256$, since we hypothesized that picking up higher frequency features on this dataset would help. We also used a step scheduler that decayed the LR by 0.5 every 100 epochs, following prior work [21, 8].

ListOps. We hypothesized that this task benefits from deeper models, because of the explicit hierarchical nature of the task, so the architecture used here had 8 layers and $H = 128$ hidden features. However, results are very close with much smaller models. We also found that post-norm generalized better than pre-norm, but results are again close (less than 1% difference).

PathX. As described in [10], the initialization range for PathX is decreased from $(\Delta_{min}, \Delta_{max}) = (0.001, 0.1)$ to $(\Delta_{min}, \Delta_{max}) = (0.0001, 0.01)$.