DELAY-SSD: READING PAST THE DECAY WITH LAGGED STATE UPDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured state space models such as Mamba-2 forget under one scalar decay per head, so a token's influence is fixed by its distance in the scan. A dependency at a fixed lag p, such as the pixel above in a flattened image or the period of a seasonal series, must therefore be carried through p decay steps and competes for state with every token in between. We introduce Delay-SSD, which writes the input from p steps earlier directly into the SSD state update, so that the lagged token is read at decay distance zero. All lags share one state and one decay, so the state size is unchanged and Mamba-2's chunked scan still applies, because the sequence transformation stays semiseparable of order N outside a band of width $\max |p|$. We show that neither more state nor another scan order buys this reach. With input-independent parameters, n_h scalar-decay heads cannot isolate a lag beyond n_h at any state size, input-dependent heads reading independent tokens are bound the same way on average, and no input-independent linear recurrence with fewer than τ states approximates a τ-step delay, whereas a single lag does both. On synthetic routing tasks, a scan order changes which offsets one SSD layer reaches but not how many, while one Delay-SSD layer also reaches the rows above and below. With our TileLang kernels, two lags cost 1.36–$1.48\times$ the device time of the four-scan SSD mixer, compared with about $2\times$ for a second layer or a $4\times$ larger state. As a vision backbone, Delay-SSD matches Mamba2D on ImageNet-1k (85.3% top-1 at 50M parameters) and COCO, and leads it on ADE20K segmentation at every size (by 3.0, 1.8 and 0.5 mIoU). In forecasting, with only the mixer changed, lags lower the error on each of five traffic datasets, by 2.2% relative to a no-lag control and by 1.4% relative to Mamba-2.

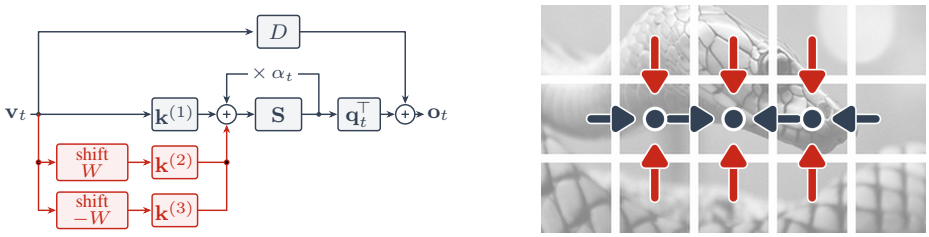
1 INTRODUCTION

State space models (SSMs) train in parallel like attention and run as a recurrence at inference, with memory linear in sequence length. Mamba (Gu & Dao, 2024) made the state transition input dependent, and Mamba-2 (Dao & Gu, 2024) restricted it to a scalar per head. In this formulation, structured state space duality (SSD), the sequence transformation is semiseparable, with every block below the diagonal of rank at most the state size N, so it can be evaluated as a chunked matrix multiplication.

SSD pays for this efficiency with an inductive bias toward locality in scan order. The state decays under one scalar per head, $\alpha_t = \exp(-A\Delta_t)$, so a token's influence fades monotonically with its distance in the scan, which suits structure that is local in scan order but not structure at a fixed lag. Once a W-wide image is flattened in row-major order, the pixel above is exactly W steps back, and the partial autocorrelation of raw images flattened at our first-stage resolution is concentrated at lag 1 and around lag W, with nothing beyond one row (Figure C.1); a seasonal time series has the same structure at its period. An SSD head can reach such a token only by carrying it through p decay steps, in competition with everything in between (§3). Vision models work around this with more scan orders (Zhu et al., 2024; Liu et al., 2024b) or kernels over the states (Xiao et al., 2024), and forecasters with decompositions around the block (An et al., 2026); none of these lets the recurrence read a token at a fixed lag directly.

Delay-SSD instead changes the recurrence, adding the input from a small set of fixed lags $\mathcal{P}$ directly to the state update with the decay and the state shared across lags. A token at lag $p \in \mathcal{P}$ is then read

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Delay-SSD head, $\mathcal{P} = \{0, W, -W\}$ (b) Lags $\pm W$

Figure 1: (a) One Delay-SSD head. Black: the Mamba-2 (SSD) head it extends, which writes $\mathbf{k}_t\mathbf{v}_t^\top$, scaled by the step size Δ_t, into the state $\mathbf{S}$, feeds the previous state back scaled by the decay α_t, and reads the state out with $\mathbf{q}_t$. Red: what Delay-SSD adds, namely shifted copies of the input, each with its own projection $\mathbf{k}^{(i)}$, written into the *same* state, with the one decay α_t computed from the step sizes of all lags (§2.2). (b) In a row-major scan (ink), lags $\pm W$ (red) fetch the tokens above and below, so the state sees a three-row band (§2.4).

at decay distance zero, and Mamba-2's chunked scan applies unchanged. Our contributions are as follows.

- **Delay-SSD** (§2): lagged inputs to the SSD state update that keep the state size and, through one shared state, the structure of the chunked scan (Proposition 1); lags come from the row stride for images and the partial autocorrelation for time series.
- **An account of what a lag buys** (§3): with input-independent parameters, n_h scalar-decay heads cannot isolate a lag beyond n_h at any state size (Theorem 1) or approximate a longer delay (Theorem 2), and input-dependent heads reading independent tokens are bound the same way on average (Corollary 1); a scan order moves one layer's reach but cannot enlarge it (Proposition 2); and among the SSD-family alternatives we measure, a lag is the least expensive way to extend reach, at 1.36× the time of the four-scan mixer compared with 2× for a second layer.
- **Kernels and results** (§4): open-source TileLang (Wang et al., 2025) kernels for the multi-lag chunked scan, and evaluations on ImageNet-1k, COCO, ADE20K and long-term forecasting.

2 Delay-SSD

Delay-SSD is a drop-in replacement for the SSD token mixer of Mamba-2. This section states the update (§2.2), shows that it keeps the structure the chunked scan relies on (§2.3), explains how the lags are chosen (§2.4), and gives the two architectures we evaluate (§2.5).

2.1 Background: the SSD head

For a token $t \in \{1, \dots, L\}$ an SSD head (Dao & Gu, 2024) keeps a state $\mathbf{S}_t \in \mathbb{R}^{N\times C}$ and computes

$$\mathbf{S}_t = \alpha_t\, \mathbf{S}_{t-1} + \Delta_t\, \mathbf{k}_t \mathbf{v}_t^\top, \qquad \mathbf{o}_t = \mathbf{q}_t^\top \mathbf{S}_t + D\, \mathbf{v}_t, \tag{1}$$

where $\mathbf{v}_t \in \mathbb{R}^C$ is the value and C the head dimension, and $\mathbf{k}_t, \mathbf{q}_t \in \mathbb{R}^N$ are input-dependent state projections, known as B_t and C_t in the SSM literature and as keys and queries in linear attention. The step size $\Delta_t = \mathrm{softplus}(\delta_t + b_\delta)$ is computed from a linear projection δ_t of the input at t and a learned bias b_δ, $\alpha_t = \exp(-A\Delta_t)$ is the resulting decay with a scalar $A > 0$ per head, and D is a diagonal skip connection. Multi-head models run n_h such heads in parallel; we drop the head index. The scalar decay is what makes SSD fast, because with one decay per head the *chunked scan* needs only dense intra-chunk products and an inter-chunk recurrence on the state. It also fixes the model's bias, since every column of the state fades by the same α_t. Writing $\tau = t - s$ for the *distance* in scan order from a source token s to a target t, the source is attenuated by $\alpha_{s+1:t} = \prod_{j=s+1}^{t} \alpha_j$, a product of τ decays, regardless of its content.

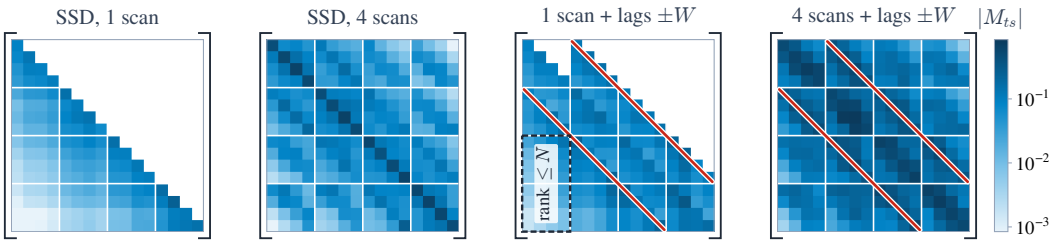


Figure 2: The mixer as a matrix $|M_{ts}|$, materialized exactly for one head on a 4×4 grid ($L = 16$, $N = 4$; white: structural zeros). One row-major scan is lower-triangular and four scans fill it by summation. Lags $\pm W$ add the diagonals $t - s = \pm W$ (red) and an upper band of width W; every block below the band keeps rank $\leq N$ (dashed; Proposition 1). §2.3.

2.2 The Lagged State Update

A *lag* p is a distance at which a source enters the state undecayed, so that through lag p a source at distance τ is at *decay distance* $\tau - p$. Given a set of lags $\mathcal{P} = \{p_1 = 0, p_2, \ldots, p_{|\mathcal{P}|}\}$, which includes $p_1 = 0$ so that two lags give $|\mathcal{P}| = 3$, Delay-SSD replaces equation 1 by

$$\Delta_t^{(i)} = \text{softplus}\left(\delta_t^{(i)} + b_\delta\right), \tag{2}$$

$$\alpha_t = \exp\Big(- A \textstyle\sum_{i=1}^{|\mathcal{P}|} \Delta_{t-p_i}^{(i)} \Big), \tag{3}$$

$$\mathbf{S}_t = \alpha_t \, \mathbf{S}_{t-1} + \textstyle\sum_{i=1}^{|\mathcal{P}|} \Delta_{t-p_i}^{(i)} \, \mathbf{k}_{t-p_i}^{(i)} \mathbf{v}_{t-p_i}^\top, \tag{4}$$

$$\mathbf{o}_t = \mathbf{q}_t^\top \mathbf{S}_t + D \, \mathbf{v}_t. \tag{5}$$

One state, one decay. All lags write into the same state and the decay is computed from the sum of the active step sizes. The state therefore stays $N \times C$ per head, the inter-chunk recurrence is unchanged, and the model cannot "spend" state on a lag, since the token at lag p_i enters at decay distance zero and is thereafter treated exactly like a token that arrived at time t. §3 shows that a layer of standard SSD heads cannot reproduce this with fewer heads than the lag.

Which parameters are per lag. The value $\mathbf{v}_{t-p_i}$ is never per lag, because it is the signal we want to re-inject and because it is the most expensive of the input-dependent sequences ($C \gg N$). The projection $\mathbf{k}^{(i)}$ and the step size $\Delta^{(i)}$ may each be shared across lags or computed per lag from the input at $t - p_i$. The reported vision models compute both per lag; sharing $\mathbf{k}$ removes the lags' parameter overhead (§4.1).

Negative lags. On non-causal data such as images we also use negative lags, which read a token *ahead*, such as the row below in Figure 1b. Because a lag is a shift of the input sequence, this needs no change to the recurrence.

2.3 Structure and Cost

Unrolling equation 4 gives the output as a sum over lags of standard SSD terms applied to shifted sequences,

$$\mathbf{o}_t = \sum_{i=1}^{|\mathcal{P}|} \sum_{s \leq t} \alpha_{s+1:t} \left(\mathbf{q}_t^\top \mathbf{k}_{s-p_i}^{(i)}\right) \Delta_{s-p_i}^{(i)} \, \mathbf{v}_{s-p_i} + D \, \mathbf{v}_t, \tag{6}$$

so each lag contributes an SSD term over its own shifted copy of the inputs, and all terms share the decay $\alpha_{s+1:t}$. Because they also share the state, the sum keeps the chunked scan's structure.

Proposition 1 (Structure). *Write equation 6 as $\mathbf{o} = M\mathbf{v}$ for one head with $D = 0$, and let $p_{\max} = \max_i |p_i|$. For every m, the block of M with rows $t \geq m$ and columns $s \leq m - 1 - p_{\max}$ has rank at most N. Entries above the diagonal exist only for negative lags and only within distance $p_{\max}$ of it. With a separate state and decay per lag the bound is $|\mathcal{P}|N$.*

The rank-N factor is the state that crosses a chunk boundary, so the inter-chunk state passing and the final state reduction are computed once, and only the intra-chunk terms are computed $|\mathcal{P}|$ times

(Figure 2); the proof is in Appendix B and the algorithm in Appendix A. Analytically the mixer's FLOPs grow by up to $|\mathcal{P}|\times$, but the measured cost is far lower because the per-lag kernels are a minority of the mixer's time (§3.4). We chose lagged *inputs* over lagged *states* so that the cost is a constant factor on the intra-chunk work, not a second state carried between chunks.

2.4 Choosing the lags

A lag is an inductive bias, so we take it from the structure of the data.

Vision. For a row-major scan over an $H \times W$ token grid the vertical neighbors are at lags $\pm W$ and for a column-major scan at $\pm H$ (Figure C.1). We use exactly these, one lag above and one below, in every scan direction, so that each scan reads the neighboring rows, or columns, at the same decay distance as its own (Figure 1b).

Time series. We take the lags whose partial autocorrelation (PACF) on the training split is significant after a Bonferroni correction, up to $|\mathcal{P}| = 3$ (Appendix H.2). The PACF ranks a lag by the variance it explains *given* the shorter lags, which a state carrying the recent past already has, so on a random walk it selects no extra lag. Our forecasting runs choose between this set and multiples of the period on validation data (Appendix H.3).

2.5 Architectures

Vision. We adopt the hierarchical four-stage macro-architecture of Mamba2D (M2D) (Baty et al., 2026; Yu et al., 2022): a strided convolutional stem, Delay-SSD blocks in the two high-resolution stages, and global attention with QK-normalization and rotary position embeddings (RoPE, Su et al., 2024) in the two low-resolution stages, followed by global average pooling and a linear classifier. Each Delay-SSD block follows VMamba2, the VMamba module (Liu et al., 2024b) with the SSD mixer of Mamba-2. A 3×3 depthwise convolution feeds the input-dependent projections, there is no gating branch, and the token mixer sums the outputs of four scan directions, row-major and column-major, each forward and reversed. The lags are $\pm W$ in the row-major scans and $\pm H$ in the column-major scans, and both $\mathbf{k}$ and Δ are computed per lag. The three sizes, DSSD-N, DSSD-T and DSSD-S, differ in depth and width (Table I.1).

Time series. A forecasting lag has to move along time, so we take the encoder of S-Mamba (Wang et al., 2024b) and change only what a token is: one time step (all variates at that step) instead of one variate's whole window, so the SSM scans along time and a lag p reads the series p steps back. With Mamba, Mamba-2 or Delay-SSD as the mixer at the same width and depth, the models are T-Mamba, T-Mamba-2 and T-DSSD. T-DSSD with $\mathcal{P} = \{0\}$, the *no-lag control*, is Mamba-2 up to kernels, activation and step-size initialization, so T-DSSD against it measures the lags alone (Appendix H.3).

3 What one SSD layer can route, and what a lag buys

Ablations can show that lags help at equal parameters; they cannot show what a lag adds that a larger state, more scan directions, a different scan order or another layer would not. This section compares a lag with those alternatives in positional reach (§3.1), on synthetic routing tasks (§3.2), on a one-dimensional delay (§3.3), and in device time and memory (§3.4). We show that, among the SSD-family alternatives we measure, a lag is the least expensive way to give a single layer reach beyond the positions just before the query in scan order, and that in one dimension no amount of SSD state substitutes for it.

3.1 Positional reach

Consider one SSD head with a per-token input projection and no positional embedding. Its weight on a source at distance $\tau = t - s$ in equation 6 is $\Delta_s \, \mathbf{q}_t^\top \mathbf{k}_s \, \alpha_{s+1:t}$. The step size and the projections depend on the content of the two tokens, and only the cumulative decay depends on the distance between them, so what a head can address by position is set by its decays. We therefore analyze the head first with input-independent parameters, the same Δ, $\mathbf{k}$ and $\mathbf{q}$ at every token. This removes addressing by content but leaves addressing by position intact, and it turns the head into a convolution with a sum-of-exponentials kernel, the form in which linear SSMs such as S4 are analyzed (Gu et al.,

2022). Corollary 1 then carries the result to input-dependent heads. On a grid of independent tokens, singling out the token at distance W, such as the pixel above in a row-major scan, means building from the heads' decays a kernel $h(\tau)$ that vanishes at $\tau = 0, \dots, W-1$ but not at $\tau = W$.

Theorem 1 (Isolating a lag)**.** *With input-independent parameters, one SSD head has the kernel $h(\tau) = c\,\alpha^\tau$ with $c = \Delta\,\mathbf{q}^\top\mathbf{k}$ for every state size N, so a layer of n_h heads whose outputs are mixed linearly has a kernel $h(\tau) = \sum_{j=1}^{n_h} c_j \alpha_j^\tau$. If $n_h \le W$ and $h(\tau) = 0$ for $\tau = 0, \dots, W-1$, then $h \equiv 0$. Isolating distance W therefore takes $n_h \ge W+1$ heads, or $n_h \ge W$ with the skip D, which acts only at $\tau = 0$, whereas a lag $p = W$ does it with one.*

Input dependence lets a head choose by content what to write and what to read, but when the tokens are independent their content says nothing about where they are. On average, an input-dependent head then addresses positions like an input-independent head with its mean decay.

Corollary 1 (Input-dependent heads)**.** *Let the tokens be independent and identically distributed, and let Δ_t, $\mathbf{k}_t$, $\mathbf{q}_t$ and $\mathbf{v}_t$ depend only on the token at t. Far from the start of the sequence, the expected change in a head's output due to the token at distance $\tau \ge 1$, $\mathbb{E}[\mathbf{o}_t \mid x_{t-\tau}] - \mathbb{E}[\mathbf{o}_t]$, is $\mathbf{g}(x_{t-\tau})\,\bar\alpha^{\,\tau-1}$, where $\bar\alpha = \mathbb{E}[\alpha_t]$ and $\mathbf{g}$ does not depend on τ. A layer of $n_h \le W-1$ input-dependent heads whose outputs do not depend on average on the tokens at distances $1, \dots, W-1$ therefore does not depend on average on the token at distance W either.*

Approximate isolation needs heads whose contributions nearly cancel, with coefficients that grow exponentially in W. A kernel that is small before lag W can be built that way, but in our numerical fits one that is also small after W cannot, even with $n_h = W \le 8$ heads (Appendix B). At initialization, the gradient of a single row-major scan from a query to the pixel above is 4×10^{-5} of the gradient to its left neighbor at $W = 16$ and 10^{-9} at $W = 32$ (Appendix C).

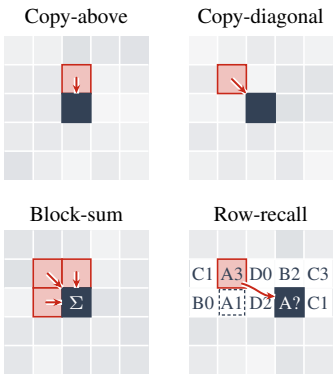
Call r the number of positions just before a query in scan order whose gradient to the query is at least a fraction ϵ of that of the immediately preceding position, and say the layer *resolves* r decay steps; by the measurement above, $r < 16$ at initialization for any $\epsilon > 10^{-4}$. The positions a layer resolves, directly or through a lag, are its *reach*, and the pixels that can influence a query within ℓ layers are all sums of ℓ offsets from the one-layer reach. For a *single* causal scan that set is the r positions before the query in scan order, so the reach after ℓ layers is exactly $\ell r + 1$ positions whatever order the scan visits the grid in. More generally, a scan order decides *which* offsets a layer reaches, never *how many*.

Proposition 2 (Reach is a count)**.** *If one layer resolves r decay steps along a route (a scan direction with lag set $\mathcal{P}_{route} \ni 0$), a query's one-layer reach has at most $\sum_{routes} \big((r+1)|\mathcal{P}_{route}| - 1\big)$ positions besides the query itself, for every choice of scan orders.*

An order can therefore make any *single* offset adjacent in scan order, but at $r = 2$ no order lets one route without lags reach the three neighbors needed by the block-sum task below, or a whole row as row-recall needs. An anti-diagonal zigzag, 4×4 windows, a Hilbert curve or a scan along the main diagonals costs nothing and adds no reach; the diagonal scan, for instance, solves copy-diagonal in one layer but none of the other three tasks (Figure D.1). Only additional routes, whether scan directions or lags, enlarge the reach. On a 16×16 grid at $r = 2$, one layer of four scan directions reaches 9 pixels, as does one scan with two lags, and the full vision mixer, with four scans and ± 1-row lags, reaches 21 and covers the grid in six layers (Figure C.2a). The one-layer set of VMamba's four-direction cross-scan is also a different shape, a plus, so every diagonal offset needs two layers, whereas a ± 1-row lag reaches into the previous and next rows in one. Trained effective receptive fields show the cross of the four scans, and the lags move some gradient mass into the adjacent rows (Figure C.3).

3.2 Synthetic routing tasks

The reach argument predicts that one SSD layer should fail any task whose source is not just before the query in one of its scan orders, whatever its state size, and that a lag should fix exactly those tasks. We test this prediction on four tasks over a 16×16 grid of tokens drawn independently from a small vocabulary. The models get no positional embedding and no convolution, so information can move between positions only through the recurrence. Each model embeds the tokens, applies ℓ blocks of a pre-norm mixer and a pre-norm FFN, and classifies every pixel; the model width is 64, $N = 16$, and each scan direction has four heads (Appendix D). The target at pixel (i, j) is:



	Copy-above	Copy-diag.	Block sum	Row recall	train MFLOPs
SSD, 1 scan	25	25	50	58	55
SSD, 1 scan, 2 layers	31	30	50	55	110
SSD, row + row-rev.	26	25	50	50	70
SSD, row + col.	**100**	28	50	49	70
SSD, row + col., 2 layers	**100**	**100**	**100**	**99**	139
SSD, 4 scans (VMamba2)	**100**	33	50	46	99
SSD, 4 scans, 2 layers	**100**	**100**	**100**	**100**	198
Delay-SSD, 1 scan + 1 lag	**100**	**100**	**100**	**100**	67
Delay-SSD, 4 scans + lags	**100**	**100**	**100**	**100**	191

Figure 3: Routing on 16×16 grids of i.i.d. tokens without positional embedding. Left: the four tasks on a 5×5 excerpt (query dark, source red, row-recall decoy dashed). Right: best accuracy (%, mean of three seeds; color runs from chance to solved). One Delay-SSD layer solves all four tasks; SSD needs two layers and scans along both rows and columns. §3.2, Table D.1.

- **Copy-above**: the token above, $x[i-1, j]$, which is one step back in a column scan and W steps back in a row scan.
- **Copy-diagonal**: the token above-left, $x[i-1, j-1]$, which is adjacent in no row or column scan.
- **Block-sum**: the sum of the 2×2 block ending at (i, j), which needs three neighbors at once.
- **Row-recall**: each token is a key–value pair, and the target is the value that the query's key carries in the previous row. The same keys also appear in the query's own row with decoy values, so the model has to read the row above rather than the most recent match.

The baseline is VMamba2, which is Delay-SSD without lags, run on the same kernels.

As predicted, one Delay-SSD layer solves all four tasks within 100–300 training steps (Figure 3), and so does a single scan with a single lag, which at 67 MFLOPs per sample is the cheapest configuration that solves any of them. One SSD layer solves only copy-above, and only when one of its scans runs down the columns, where a lag and a second scan direction tie at similar cost (70 MFLOPs for a row-and-column scan). One SSD layer stays at 25–33% on copy-diagonal and 46–58% on row-recall, where chance is 12.5% and 25%, and stops at 50% on block-sum, the accuracy of seeing three of the four pixels. A second SSD layer solves all three at twice the cost, but only when the model scans both rows and columns. With a single scan, even two layers fail every task, including copy-above (31%).

Controls. More heads or more state do not give one SSD layer this reach. A learned positional embedding can, but every construction we found needs a state that grows with W, whereas a lag needs $N = O(1)$ (Appendix D.1). With the 3×3 convolution of the full vision configuration in place, one lagged layer solves a copy from two rows and two columns away, which the convolution alone does not (Appendix D.2).

3.3 One dimension: what state cannot buy

By Theorem 1, a Mamba-2 head with input-independent parameters has one real decay rate whatever its N, so in one dimension a larger state buys no temporal resolution, and a long delay is out of reach of a few heads.

Theorem 2 (A delay needs τ states)**.** *No linear recurrence with input-independent parameters and fewer than τ states approximates the τ-step delay $y_t = x_{t-\tau}$ better than outputting zero. In the worst case over past inputs of unit energy, its error on the future outputs is at least* 1.

The past reaches the future only through the state, so with $n < \tau$ states some past input leaves the state at zero, and the system outputs nothing where the delay must reproduce that input; smooth inputs leave fewer such directions, which is why short delays are within reach. Appendix B gives the proof and a worked example. A lag $p = \tau$, in contrast, realizes the delay with a single head. On the delay task of Voelker et al. (2019); Gu et al. (2020), whose target at time t is $x_{t-\tau}$, one Delay-SSD layer with $\mathcal{P} = \{0, \tau\}$ reaches a relative RMSE of 0.013–0.014 for every τ from 8 to 1024, where 1

Table 1: ImageNet-1k top-1 accuracy as reported, ordered by parameters. Ours is the top-1 of the exponential moving average (EMA) of the weights after the last of 300 epochs, with no epoch selection on the validation set. FLOPs (G) at 224^2 count multiply-accumulates, following Swin and VMamba, and are as published; Vim reports none. Ours are analytic, with the scan counted in its recurrent form of one multiply-add per state entry for the decay, for each lag's write and for the read-out; published counts differ in how they treat the scan (Appendix F). Bold: no model with at most as many parameters is more accurate; shaded: ours. §4.1.

Model	Params	FLOPs	Top-1 (%)
Vim-Ti	7M	–	76.1
M2D-N	7M	1.4	79.6
DSSD-N	7M	1.6	**79.9**
VSSD-M	14M	2.3	**82.5**
VSSD-T	24M	4.5	**83.7**
Vim-S	26M	–	80.3
LocalVMamba-T	26M	5.7	82.7
M2D-T	27M	4.7	**84.0**
DSSD-T	27M	5.0	83.9
Swin-T	29M	4.5	81.3
VMamba-T	30M	4.9	82.6
MambaVision-T	32M	4.4	82.3
EffVMamba-B	33M	4.0	81.8
VSSD-S	40M	7.4	**84.1**
MambaVision-S	50M	7.5	83.3
VMamba-S	50M	8.7	83.6
2DVMamba-S	50M	8.8	83.8
V2M-B	50M	9.6	83.8
M2D-S	50M	13.7	**85.3**
DSSD-S	50M	14.1	**85.3**
Swin-B	88M	15.4	83.5
VMamba-B	89M	15.4	83.9
VSSD-B	89M	16.1	84.7
MambaVision-B	98M	15.0	84.2

is the error of predicting zero. No SSD configuration recovers the signal for $\tau \geq 32$, across 2–32 heads, 1–4 layers, $N \leq 128$ and up to $2.8\times$ the training FLOPs (Figure E.1). The effect is specific to the lag, which leaves a residual of 0.60 when placed four samples off.

3.4 What a lag costs

Table F.2 measures the token mixer alone, at the first-stage shape of the vision models (56×56 tokens of width 96, one RTX 5090), against the SSD-family alternatives that could provide comparable reach. With two lags the mixer takes $1.36\times$ the device time and $1.12\times$ the peak training memory of the four-scan SSD mixer, and 1.46–$1.48\times$ the time and 1.06–$1.07\times$ the memory at the two deeper stage shapes. A second SSD layer executes about the same FLOPs as the lags but takes $2.00\times$ the time and adds a layer of depth, so at equal FLOPs the lags run at 0.68–$0.74\times$ its latency. A $4\times$ larger state takes $1.90\times$ the time and $1.91\times$ the memory without providing the reach. One scan with one lag, which solves all four routing tasks, needs $0.34\times$ the time and $0.46\times$ the memory of four scans.

The lags multiply only the SSD core, which accounts for a third of the mixer's device time (Appendix F). Compared with global attention, the linear-time mixer is faster above about 1.3×10^4 tokens in training and, with our fused inference kernel, above about 7×10^3 tokens at inference (Figures F.1b and F.2).

4 Experiments

We evaluate Delay-SSD on ImageNet-1k (Deng et al., 2009), COCO detection and segmentation (Lin et al., 2014), ADE20K semantic segmentation (Zhou et al., 2017), a 100-class ImageNet subset for ablations, and long-term forecasting with the PEMS03/04/07/08 traffic-flow sets (Chen et al., 2001; Liu et al., 2022a) and the Traffic road-occupancy set (Lai et al., 2018; Wu et al., 2021) (Appendix H.1).

4.1 ImageNet-1k classification

Table 1 compares our three models, DSSD-N, DSSD-T and DSSD-S (nano, tiny, small), with 1-D and 2-D Mamba backbones, including M2D and MambaVision (Hatamizadeh & Kautz, 2025), and with Transformer backbones at matched parameter counts (Figure 4a). At the nano scale, DSSD-N reaches 79.9%, above M2D-N (79.6%) and Vim-Ti (76.1%). At the tiny scale, DSSD-T reaches 83.9%, 0.1 points below M2D-T and more than a point above the multi-scan VMamba-T and LocalVMamba-T. At the small scale, DSSD-S matches M2D-S at 85.3% and exceeds VSSD-B (Shi et al., 2025) (84.7%) with 39M fewer parameters and comparable FLOPs, 14.1G in our count and 16.3G as executed compared with its 16.1G. Delay-SSD thus matches Mamba2D's 2-D wavefront scan with Mamba-2's

Table 2: Dense prediction: Mask R-CNN on COCO (box and mask AP at the 1× and 3× schedules) and UperNet on ADE20K (single- and multi-scale mIoU after 160k iterations). Each size group shows Mamba2D, the strongest VSSD or VMamba model and ours; Tables I.3 and I.2 list every comparator and size. Bold: best of the size group in those full tables; shaded: ours.

	COCO 1×		COCO 3×		ADE20K	
Backbone	AP^b	AP^m	AP^b	AP^m	SS	MS
VSSD-M	**45.4**	**41.3**	**47.7**	**42.8**	45.6	46.0
M2D-N	42.8	39.6	47.2	42.4	43.1	43.1
DSSD-N	43.9	40.3	47.0	42.4	**46.1**	**46.2**
VMamba-T	47.3	42.7	48.8	43.7	47.9	48.8
M2D-T	48.5	**43.8**	**50.5**	**44.7**	48.9	49.3
DSSD-T	**48.6**	43.7	50.3	**44.7**	**50.7**	**51.2**

1-D chunked scan and $N = 32$, at 3–12% more FLOPs. Besides M2D-T, the one model ahead of Delay-SSD in its group is VSSD-M, which removes the decay altogether in favor of positional embeddings and has twice the parameters of DSSD-N. Model-level throughput and memory, with a fused inference kernel, are in Table F.5.

Ablations. Table G.1 ablates the vision design at a fixed parameter count on a 100-class ImageNet subset (three seeds, 100 epochs). The lags are worth 0.5 points on 15k held-out images (74.6 vs. 74.1), both against the no-lag model at equal parameters and against one that spends the lags' time on extra blocks. Sharing Δ across lags costs nothing, and sharing $\mathbf{k}$ costs 0.3 points. The trained ImageNet models use their lags in every block, and switching them all off costs 3–6 points (Appendix G.2).

4.2 Dense prediction: COCO and ADE20K

Table 2 gives the strongest comparators at each size, and Figure 4(b, c) places Delay-SSD among all of them; recipes are in Appendices I.4 and I.5. On COCO, DSSD-T matches M2D-T within 0.2 points in box and mask average precision (AP) under both schedules and is ahead of VMamba-T, VSSD-T and V2M-S, and DSSD-N is second to VSSD-M at 1×. On ADE20K, Delay-SSD is the best backbone in every size group, 3.0, 1.8 and 0.5 mIoU above Mamba2D, and DSSD-S reaches 52.2 mIoU single-scale and 52.8 multi-scale (Table I.2).

4.3 Long-term forecasting

Forecasting tests the lag in a second modality with everything but the mixer fixed. We use twenty (dataset, horizon) settings on PEMS03/04/07/08 and Traffic, a lookback window of $L = 96$ steps and three seeds, and choose the learning rate and the lag set on validation data (Figure H.2, Table H.3).

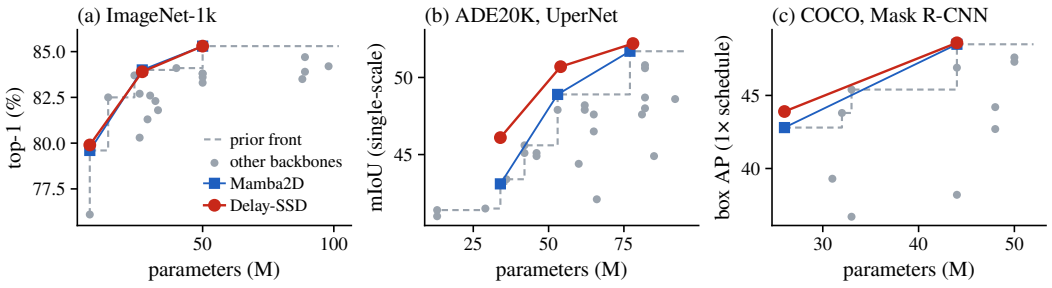


Figure 4: Accuracy against parameters for every model of **(a)** Table 1 (ImageNet-1k top-1), **(b)** Table I.2 (ADE20K, single-scale mIoU) and **(c)** Table I.3 (COCO, box AP at the 1× schedule): Delay-SSD N, T, S in red, Mamba2D in blue, others gray; dashed: best comparator at no more parameters.

What the comparison shows. T-DSSD lowers the mean squared error (MSE) by 2.2% relative to the no-lag control, in 18 of 20 settings ($p < 0.001$, one-sided Wilcoxon signed-rank test over the settings), and by 1.4% relative to T-Mamba-2 (18/20, $p = 0.004$). It is better than both on all five datasets ($p = 0.031$, one-sided test over datasets). The gain also holds at equal size, since with no more parameters or FLOPs than T-Mamba-2, T-DSSD still lowers its MSE by 1.1% (18/20, $p = 0.002$), whereas narrowing T-Mamba-2 to the same width raises its MSE by 0.7%. With lags $\{0, 4, 14\}$ on PEMS04 at horizon 48, the last token is $0.26\times$ and $0.77\times$ as sensitive to the inputs 4 and 14 steps back as to the previous one, compared with 0.12 and 0.06 without lags, so the model reads the lagged steps directly (Figure H.3). A lag in the mixer also differs from a seasonal prefilter, since differencing or an ARIMA fit removes one linear dependence from the raw series once, whereas a Delay-SSD lag acts on learned features in every layer (Appendix H.7).

Scope. Beyond this comparison of mixers within one forecaster, Appendix H compares the variate-token S-Mamba (Wang et al., 2024b) and a dual-path model that adds a time-token branch to it, and shows that the lags also lower the error of a causal forecaster, by 2.1% relative to T-Mamba-2 (24 of 28 settings). On the small channel-independent ETT (Zhou et al., 2021) and Weather (Wu et al., 2021) sets, time-token models mostly overfit within one epoch.

5 Related Work

Structured state space models. S4 (Gu et al., 2022) and Mamba (Gu & Dao, 2024) discretize a continuous-time transition, Mamba with input-dependent selection; Mamba-2 (Dao & Gu, 2024) restricts it to a scalar per head, which exposes the duality with linear attention (Katharopoulos et al., 2020; Yang et al., 2024), permits a chunked scan and makes the decay uniform. Non-scalar transitions, as in the Legendre Memory Unit (Voelker et al., 2019) and HiPPO (Gu et al., 2020), solve the delay task of §3.3, and negative or complex eigenvalues enrich a linear recurrence's modes (Grazzi et al., 2025); Delay-SSD keeps the scalar pole and adds a lag. Fixed-state recurrences struggle with copying at a distance (Jelassi et al., 2024) and with state tracking (Merrill et al., 2024). Delayed inputs to a recurrence go back to NARX networks (Lin et al., 1996) and dilated recurrent networks (Chang et al., 2017), which shorten gradient paths the same way (Appendix B).

SSMs for vision. Most vision SSMs restore vertical locality by adding scans: bidirectional (Zhu et al., 2024), cross-scan (Liu et al., 2024b), space-filling (Hu et al., 2024b; Yang et al., 2025), windowed (Huang et al., 2024) or atrous (Pei et al., 2024). Others change the recurrence: row and column states (Zhang et al., 2026; Wang et al., 2024a), a 2-D wavefront transition (Baty et al., 2026), no decay (Shi et al., 2025), or kernels over materialized states (Xiao et al., 2024). A fixed lag on a row-major sequence is also a shift, as in TSM and ShiftViT (Lin et al., 2019; Wang et al., 2022), here written into the state; MambaOut (Yu & Wang, 2025) asks whether an SSM is needed at 224^2 at all (Figure F.1b).

SSMs for forecasting. S-Mamba (Wang et al., 2024b) scans across variates as in iTransformer (Liu et al., 2024a); later work adds multi-scale hierarchies (Ahamed & Cheng, 2024; Karadag et al., 2025) or Fourier-split dual paths (An et al., 2026), and linear and patch models remain strong (Zeng et al., 2023; Nie et al., 2023). Attraos (Hu et al., 2024a) is closest to our use of lags: it feeds a time-delay embedding of each variate to a Mamba-style SSM as input coordinates, whereas Delay-SSD adds the lagged input to the state update. Our PACF lag selection follows the Box–Jenkins identification of autoregressive moving-average (ARMA) models (Box et al., 2015).

6 Conclusion

Delay-SSD lets a scalar-decay SSM address the past by lag as well as by distance, a reach no larger state or scan order provides, at 1.36–$1.48\times$ the mixer time compared with about $2\times$ for a second layer or a larger state. It is on par with the best 2-D Mamba backbone on ImageNet-1k and COCO, ahead on ADE20K at every scale, and lowers the forecasting error on traffic data by 2.2% relative to the same model without lags and by 1.1% relative to Mamba-2 at equal parameters and FLOPs.

Limitations. A lag is a prior on where structure lies, and PACF selects no lag on a random walk. On small forecasting sets such as ETT, time-token models mostly overfit within one epoch, with or without lags. The vision ablations use a 100-class subset, and the ImageNet-1k, COCO and ADE20K results are single runs.

AI USE STATEMENT

The idea, the model and its base implementation, and the design of the vision and forecasting experiments were developed by the authors over roughly eight months with limited use of AI tools. In the final phase of the project we used generative AI tools: to propose some of the synthetic tasks of Section 3; to implement other parts of the code, including analysis and plotting scripts, the migration of the time-series forecasting codebase, experiment grids, throughput benchmarks and the fused inference kernel of Appendix F; to run and monitor experiments; to give feedback on the experimental design and propose additional controls and statistical tests; to help formalize propositions from our intuitions and sketches and draft their proofs; to search for related work and published results and to critique drafts of the paper; and to write and edit the text, tables and figures. No data were generated by AI models, since the synthetic tasks are sampled by programs; translation and qualitative data analysis are not applicable to this work. We reviewed all AI-assisted work: we checked every written interpretation against our own reading of the underlying results; every number reported for our models is produced by a script from logged runs, and comparator numbers taken from the cited papers were human verified; the final configurations of all experiments are based on hand-written base configurations; AI-written kernels are tested against human written reference implementations; references found via automated literature review have been checked and all proofs were checked by hand by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The code supplement contains the models as a single file (the Delay-SSD mixer, the DSSD-N/T/S classifiers and the TileLang kernels of the multi-lag chunked scan, with a pure-PyTorch fallback) and a dependency-only install script pinned to the versions we used. The full repository, with the training and evaluation code, the configurations and the scripts that produce every table and figure, will be released under the MIT license. Seeds, hardware and software are in Appendix I.1, the vision training recipes in Appendix I and the forecasting protocol in Appendix H.

REFERENCES

Md Atiq Ahamed and Qiang Cheng. TimeMachine: A time series is worth 4 mambas for long-term forecasting. In *International Conference on Machine Learning*, pp. 525–541. PMLR, 2024.

Rui An, Haohao Qu, Wenqi Fan, Xuequn Shang, and Qing Li. DeMa: Dual-path delay-aware Mamba for efficient multivariate time series analysis. *Frontiers of Computer Science*, 2026. arXiv:2601.05527.

Enis Baty, Alejandro Hernández Díaz, Rebecca Davidson, Chris Bridges, and Simon Hadfield. Mamba2D: A natively multi-dimensional state-space model for vision tasks. 2026. URL `https://arxiv.org/abs/2412.16146`.

Bernhard Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numerische Mathematik*, 85(4):553–577, 2000.

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 5 edition, 2015.

Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. Springer, 2 edition, 1991.

Xiuding Cai, Xueyao Wang, Yaoyao Zhu, and Yu Yao. MambaTS: Improved selective state space models for long-term time series forecasting. *Pattern Recognition*, 2026. arXiv:2405.16440.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark Hasegawa-Johnson, and Thomas S. Huang. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2017.

Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: Mining loop detector data. *Transportation Research Record: Journal of the Transportation Research Board*, 1748:96–102, 01 2001. doi: 10.3141/1748-12.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. MMDetection: OpenMMLab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, pp. 10041–10071. PMLR, 2024.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. IEEE, 2009.

Walter Gautschi. On inverses of Vandermonde and confluent Vandermonde matrices. *Numerische Mathematik*, 4(1):117–123, 1962.

Keith Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds. *International Journal of Control*, 39(6):1115–1193, 1984.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear RNNs through negative eigenvalues. In *International Conference on Learning Representations*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

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 Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6185–6194, 2023.

Ali Hatamizadeh and Jan Kautz. MambaVision: A hybrid Mamba-transformer vision backbone. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25261–25270, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Jiaxi Hu, Yuehong Hu, Wei Chen, Ming Jin, Shirui Pan, Qingsong Wen, and Yuxuan Liang. Attractor memory for long-term time series forecasting: A chaos perspective. In *Advances in Neural Information Processing Systems*, 2024a.

Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes Schusterbauer, and Björn Ommer. ZigMa: A DiT-style zigzag Mamba diffusion model. In *European Conference on Computer Vision*, pp. 192–210. Springer, 2024b.

Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. LocalMamba: Visual state space model with windowed selective scan. In *European Conference on Computer Vision*, pp. 12–22. Springer, 2024.

Sukjun Hwang, Aakash Lahoti, Ratish Puduppully, Tri Dao, and Albert Gu. Hydra: Bidirectional state space models through generalized matrix mixers. *Advances in Neural Information Processing Systems*, 37:110876–110908, 2024.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *International Conference on Machine Learning*, 2024.

Yusuf Meric Karadag, Ismail Talaz, Ipek Gursel Dino, and Sinan Kalkan. ms-Mamba: Multi-scale Mamba for time-series forecasting. *Neurocomputing*, 2025. arXiv:2504.07654.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 95–104, 2018. doi: 10.1145/3209978.3210006.

Henry J. Landau and Henry O. Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty—III: The dimension of the space of essentially time- and band-limited signals. *Bell System Technical Journal*, 41(4):1295–1336, 1962.

Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Tsungnan Lin, Bill G. Horne, Peter Tiňo, and C. Lee Giles. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. In *Advances in Neural Information Processing Systems*, volume 35, 2022a.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024a. doi: 10.48550/arxiv.2310.06625.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. VMamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2024b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022b.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *International Conference on Machine Learning*, 2024.

MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark, 2020.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Xiaohuan Pei, Tao Huang, and Chang Xu. EfficientVMamba: Atrous selective scan for light weight visual Mamba. *arXiv preprint arXiv:2403.09977*, 2024.

Yuheng Shi, Minjing Dong, and Chang Xu. Multi-scale VMamba: Hierarchy in hierarchy visual state space model. *Advances in Neural Information Processing Systems*, 37:25687–25708, 2024.

Yuheng Shi, Mingjia Li, Minjing Dong, and Chang Xu. VSSD: Vision Mamba with non-causal state space duality. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10819–10829, 2025.

David Slepian. Prolate spheroidal wave functions, Fourier analysis, and uncertainty—V: The discrete case. *Bell System Technical Journal*, 57(5):1371–1430, 1978.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Aaron R. Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Chengkun Wang, Wenzhao Zheng, Yuanhui Huang, Jie Zhou, and Jiwen Lu. V2M: Visual 2-dimensional Mamba for image representation learning, 2024a. URL https://arxiv.org/abs/2410.10382.

Guangting Wang, Yucheng Zhao, Chuanxin Tang, Chong Luo, and Wenjun Zeng. When shift operation meets vision transformer: An extremely simple alternative to attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Lei Wang, Yu Cheng, Yining Shi, Zhengju Tang, Zhiwen Mo, Wenhao Xie, Lingxiao Ma, Yuqing Xia, Jilong Xue, Fan Yang, and Zhi Yang. TileLang: A composable tiled programming model for ai systems, 2025. URL https://arxiv.org/abs/2504.17577.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is Mamba effective for time series forecasting? *Neurocomputing*, 609:128448, 2024b.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021. doi: 10.48550/arxiv.2106.13008.

Chaodong Xiao, Minghan Li, Zhengqiang Zhang, Deyu Meng, and Lei Zhang. Spatial-Mamba: Effective visual state space models via structure-aware state fusion. *arXiv preprint arXiv:2410.15091*, 2024.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 418–434, 2018.

Xiongxiao Xu, Canyu Chen, Yueqing Liang, Baixiang Huang, Guangji Bai, Liang Zhao, and Kai Shu. SST: Multi-scale hybrid Mamba-transformer experts for time series forecasting. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, 2025.

Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J. Crowley. PlainMamba: Improving non-hierarchical Mamba in visual recognition. In *International Conference on Learning Representations*, 2025.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, pp. 56061–56081. PMLR, 2024.

Weihao Yu and Xinchao Wang. MambaOut: Do we really need Mamba for vision? In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 4484–4496, 2025.

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. MetaFormer is actually what you need for vision. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10809–10819. IEEE, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence*, 2023.

G. Peter Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175, 2003.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Jingwei Zhang, Anh Tien Nguyen, Xi Han, Vincent Quoc-Huy Trinh, Hong Qin, Dimitris Samaras, and Mahdi S. Hosseini. 2DMamba: Efficient state space model for image representation with applications on giga-pixel whole slide image classification, 2026. URL https://arxiv.org/abs/2412.00678.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

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. doi: 10.1609/aaai.v35i12.17325.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model. In *International Conference on Machine Learning*, pp. 62429–62441. PMLR, 2024.

A CHUNKED SCAN AND COMPLEXITY

The chunked scan of Mamba-2 has four parts: the intra-chunk output, the intra-chunk state, the inter-chunk state passing, and the final output. Let t_c be the left boundary of a chunk. For $p_i \in \mathcal{P}$ and $t_c \le s \le t \le t_{c+1} - 1$ the intra-chunk output weight of position s at target t is $\alpha_{s+1:t}\, \mathbf{q}_t^\top \Delta_{s-p_i}^{(i)} \mathbf{k}_{s-p_i}^{(i)}$, applied to $\mathbf{v}_{s-p_i}$ as in equation 6, and the intra-chunk state is $\sum_{s=t_c}^{t_{c+1}-1} \sum_i \alpha_{s+1:t_{c+1}-1} \Delta_{s-p_i}^{(i)} \mathbf{k}_{s-p_i}^{(i)} \mathbf{v}_{s-p_i}^\top$, which is the correct state at the end of the chunk for a zero boundary state. State passing then corrects each chunk's state with the true history, and the final output reduces $\mathbf{q}_t^\top \mathbf{S}_{t_c}$ against the boundary state, expands the intra-chunk outputs by $\mathbf{v}_{t-p_i}$, sums them across the chunk and adds the two. Of these steps, state passing and the state reduction are independent of $|\mathcal{P}|$, whereas the intra-chunk output, the intra-chunk state outer product and the expand-and-sum are computed $|\mathcal{P}|$ times, over inputs that are shifts of the same sequence.

Proposition 3 (Multi-lag complexity)**.** *For sequence length L, chunk length Q, state size N, head dimension C and lag set $\mathcal{P}$, the intra-chunk work of Delay-SSD is $\mathcal{O}\big(|\mathcal{P}|L(Q(N+C)+NC)\big)$, $|\mathcal{P}|$ times that of SSD, and the sequential inter-chunk state passing does not depend on $|\mathcal{P}|$.*

The $|\mathcal{P}|$ branches are evaluated in an independent loop over the shifted inputs and accumulated into one shared state, so the boundary states and the state-passing traffic are the same as for SSD. The step sizes require $|\mathcal{P}|$ softplus activations and sequence shifts before the sum-reduction, and the multiply-accumulates and the intermediate shared-memory allocations of the intra-chunk phases scale linearly in $|\mathcal{P}|$; no product $\mathbf{k}^{(i)}\mathbf{k}^{(j)}$ between lags is ever formed. The inter-chunk recurrence and the parts of the mixer outside the scan do not grow with $|\mathcal{P}|$, which is why the measured wall-clock penalty is below the FLOP multiplier (§3.4).

B PROOFS

Throughout, quantities of lag i are indexed by the *source* position: $\mathbf{k}_s^{(i)}$ and $\Delta_s^{(i)}$ are the key and step size computed for lag i from the input at s, which enters the state at time $s + p_i$.

Proposition 1 (structure). Unrolling equation 4 and collecting the coefficient of $\mathbf{v}_s$ in $\mathbf{o}_t$ gives

$$M_{ts} = \sum_{i=1}^{|\mathcal{P}|} \mathbf{1}[s + p_i \le t]\; \alpha_{s+p_i+1:t}\, \Delta_s^{(i)}\, \mathbf{q}_t^\top \mathbf{k}_s^{(i)}.$$

Let $t \ge m$ and $s \le m - 1 - p_{\max}$. Then $s + p_i \le m - 1 < t$ for every i, so every term is present, and because α_t is a scalar shared by all lags, $\alpha_{s+p_i+1:t} = \alpha_{m:t}\, \alpha_{s+p_i+1:m-1}$. Hence

$$M_{ts} = \big(\alpha_{m:t}\, \mathbf{q}_t\big)^\top \mathbf{x}_s, \qquad \mathbf{x}_s = \sum_{i=1}^{|\mathcal{P}|} \alpha_{s+p_i+1:m-1}\, \Delta_s^{(i)}\, \mathbf{k}_s^{(i)} \in \mathbb{R}^N,$$

and the block is the product of a matrix with N columns and one with N rows, so its rank is at most N (Figure B.1). The vectors $\mathbf{x}_s$ summed over the previous chunk are the state passed across the chunk boundary at m, which is why state passing is independent of $|\mathcal{P}|$. An entry with $s > t$ needs $s + p_i \le t$, i.e. $p_i \le t - s < 0$ and $s - t \le -p_i \le p_{\max}$. If each lag had its own state and decay $\alpha^{(i)}$, the row factors $\alpha_{m:t}^{(i)} \mathbf{q}_t$ would differ by lag and the block would be a sum of $|\mathcal{P}|$ rank-N matrices. Separate states with a shared decay and readout add up to the shared state. For a forward and a reversed scan the sum is quasiseparable outside the band, the class Hwang et al. (2024) identify for bidirectional SSMs. □

Theorem 1 (isolating a lag). With $\Delta_t \equiv \Delta$, $\mathbf{k}_t \equiv \mathbf{k}$ and $\mathbf{q}_t \equiv \mathbf{q}$, equation 1 gives $\mathbf{o}_t = \sum_{\tau \ge 0} c\, \alpha^\tau \mathbf{v}_{t-\tau} + D\mathbf{v}_t$ with $c = \Delta\, \mathbf{q}^\top \mathbf{k}$. The state size N enters only through the scalar c. Heads with equal decays merge, so a linear mixture of n_h heads has the kernel $h(\tau) = \sum_{j \le n_h} c_j \alpha_j^\tau$ with distinct α_j. If h vanishes on $\{0, \dots, W-1\}$ and $n_h \le W$, the first n_h of these equations form a square Vandermonde system $[\alpha_j^\tau]_{\tau < n_h,\, j \le n_h}\, \mathbf{c} = 0$. Its determinant $\prod_{j<k}(\alpha_k - \alpha_j)$ is nonzero, so $\mathbf{c} = 0$. The skip D adds only to $h(0)$, so with it only the equations at $\tau = 1, \dots, W-1$ constrain $\mathbf{c}$, and the same argument on $[\alpha_j^\tau]_{1 \le \tau \le n_h}$ (determinant $\prod_j \alpha_j \prod_{j<k}(\alpha_k - \alpha_j) \ne 0$) gives $\mathbf{c} = 0$

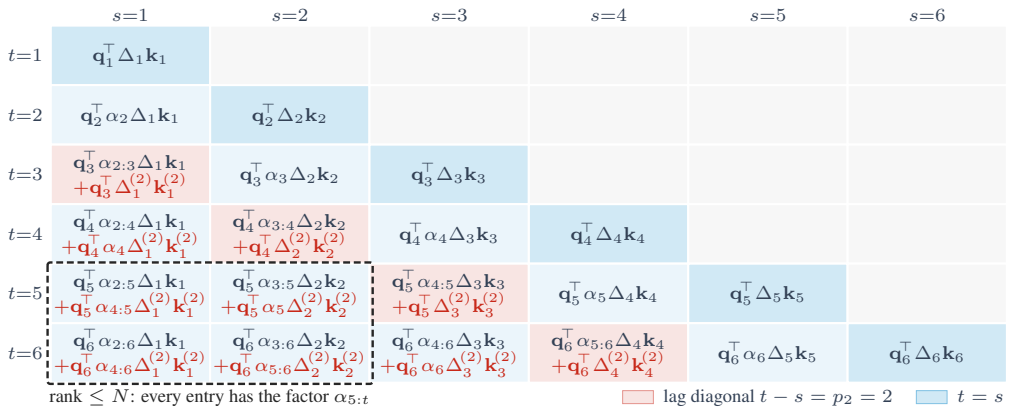


Figure B.1: Proposition 1 on a 1-D example ($L = 6$, $\mathcal{P} = \{0, 2\}$, one head). Each entry M_{ts} is the lag-0 term plus, on and below the lag diagonal $t - s = 2$, the lag-2 term (red), both indexed by the source s. Every entry of the dashed block ($t \geq 5$, $s \leq m - 1 - p_{\max} = 2$) carries the factor $\alpha_{5:t}$ of the shared decay, so the block factors through the N-dimensional state and has rank $\leq N$. Discussed in §2.3.

whenever $n_h \leq W - 1$. With a lag at W, one head has the kernel $g_0\alpha^\tau + g_W\alpha^{\tau-W}\mathbf{1}[\tau \geq W]$, where $g_i = \Delta^{(i)}\mathbf{q}^\top\mathbf{k}^{(i)}$. A key $\mathbf{k}^{(0)} \perp \mathbf{q}$ and a fast decay give $h(\tau) = g_W\,\mathbf{1}[\tau = W]$ in the limit. □

Isolation also has to hold after W, since a head that reads distance W reads the larger distances too unless its kernel vanishes there. If only $|h(\tau)| \leq \eta$ is asked for, the coefficients solve a least-squares problem in a real Vandermonde matrix, whose condition number grows at least exponentially in n_h for nodes in $(0, 1)$ (Gautschi, 1962; Beckermann, 2000). The one-sided target, 0 below W and 1 at W, can be approached with large coefficients of alternating sign (residual 0.11 at $W = n_h = 4$). The two-sided target, the indicator of lag W on $[0, 2W]$, cannot be approached in numerical least-squares fits, whose best relative residual stays at or above 0.77 for $n_h = W \leq 8$.

Corollary 1 (input-dependent heads). Write one head's output as $\mathbf{o}_t = \sum_{s<t} w_{ts}\mathbf{v}_s + \mathbf{u}_t$, where $w_{ts} = (\alpha_t\mathbf{q}_t)^\top(\Delta_s\mathbf{k}_s)\prod_{u=s+1}^{t-1}\alpha_u$ and $\mathbf{u}_t = (\Delta_t\mathbf{q}_t^\top\mathbf{k}_t + D)\mathbf{v}_t$ collects the terms of the target token itself. Every factor is a function of a single token, and the tokens are independent. Fix a source $s_0 = t - \tau$ with $\tau \geq 1$, write $\mathbb{E}_a$ for the expectation given $x_{s_0} = a$, and let $\mathbf{e} = \mathbb{E}[\alpha_t\mathbf{q}_t]$, $M(a) = \Delta(a)\,\mathbf{v}(a)\mathbf{k}(a)^\top$ and $\bar{M} = \mathbb{E}[M(x)]$. The token at s_0 enters $\mathbf{o}_t$ through its own write, the term $s = s_0$, and through its decay α_{s_0}, which multiplies the write of every earlier source. Independence factorizes the expectation of each term; the write of s_0 passes $\tau - 1$ intermediate decays and the write of $s < s_0$ passes $t - s - 2$ besides α_{s_0}. For a sequence that starts at $s = 1$,

$$\mathbb{E}_a[\mathbf{o}_t] - \mathbb{E}[\mathbf{o}_t] = \bar{\alpha}^{\,\tau-1}\,\mathbf{g}(a) - \mathbf{r}_t(a), \qquad \mathbf{g}(a) = \big(M(a) - \bar{M}\big)\mathbf{e} + \frac{\alpha(a) - \bar{\alpha}}{1 - \bar{\alpha}}\,\bar{M}\mathbf{e},$$

$$\mathbf{r}_t(a) = \frac{\alpha(a) - \bar{\alpha}}{1 - \bar{\alpha}}\,\bar{\alpha}^{\,t-2}\,\bar{M}\mathbf{e}.$$

Neither $\mathbf{g}$ nor $\mathbf{r}_t$ depends on τ, and $\mathbf{r}_t$ vanishes far from the start of the sequence. A linear mixture of heads has the expected change $\sum_j W_j\mathbf{g}_j(a)\,\bar{\alpha}_j^{\tau-1}$, where heads with equal mean decays merge. If it vanishes at $\tau = 1, \ldots, W - 1$ for every a and $n_h \leq W - 1$, the Vandermonde system $[\bar{\alpha}_j^{\tau-1}]_{1\leq\tau\leq n_h}$, with determinant $\prod_{j<k}(\bar{\alpha}_k - \bar{\alpha}_j) \neq 0$, gives $W_j\mathbf{g}_j(a) = 0$ for every head, so the expected change also vanishes at $\tau = W$. Near the start of the sequence, $\mathbf{r}_t$ adds one node at 1, and the same argument holds for $n_h \leq W - 2$. The corollary concerns the conditional mean of the output, which is what a linear readout of the layer can extract about the source; higher moments involve products of decays and are not covered. □

Theorem 2 (a delay needs τ states). Take a single-input single-output linear recurrence with input-independent parameters and $n < \tau$ states, $\mathbf{s}_{t+1} = A\mathbf{s}_t + \mathbf{b}\,u_t$, $y_t = \mathbf{c}^\top\mathbf{s}_t + d\,u_t$, and feed it a past window $\mathbf{x} = (u_{-\tau}, \ldots, u_{-1})$ of unit energy, with no input before or after it. The past reaches the

future only through the state at time 0,

$$\mathbf{s}_0 = C\mathbf{x}, \qquad C = \left[A^{\tau-1}\mathbf{b},\ \ldots,\ A\mathbf{b},\ \mathbf{b}\right] \in \mathbb{R}^{n\times\tau},$$

and, since no input arrives after time -1, the outputs $(y_0, \ldots, y_{\tau-1})$ are $O\mathbf{s}_0$ with $O = [\mathbf{c}^\top A^i]_{i<\tau} \in \mathbb{R}^{\tau\times n}$. Because C has fewer rows than columns, some unit $\mathbf{x}$ has $C\mathbf{x} = 0$, which leaves the state at zero and every future output of the system at 0. The delay's future outputs are $y_i = u_{i-\tau}$, the window $\mathbf{x}$ itself, of energy 1. For this input the error on the future outputs therefore has energy at least 1, and outputting zero, whose error is at most 1 for every unit-energy past input, attains the bound (Figure B.2). The argument uses only that $\mathbf{s}_0$ is a linear function of the past window, so it holds unchanged if A, $\mathbf{b}$, $\mathbf{c}$ and d vary with time but not with the input. For constant parameters, OC is the leading block of the system's Hankel operator, and the bound is the Hankel-norm bound of Glover (1984), obtained here without its approximation theory. By Theorem 1, an n_h-head SSD layer with input-independent parameters is such a system with $A = \mathrm{diag}(\alpha_j)$ and at most n_h states whatever N, so for $\tau > n_h$ its error is at least 1. The causal convolution in front of the mixer is a delay line one sample shorter than its width and adds at most that many states. If the inputs are band-limited to β, the input $\mathbf{x}$ above must be band-limited too, and only about $2\beta\tau$ directions of a τ-sample window carry appreciable energy (Landau & Pollak, 1962; Slepian, 1978), so the argument needs $n < 2\beta\tau$ rather than $n < \tau$, which is why short delays are within reach. □

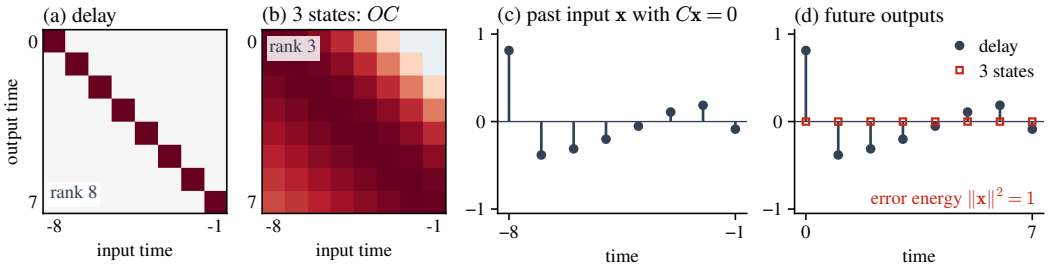


Figure B.2: Theorem 2 on an example: a delay of $\tau = 8$ and its best least-squares fit by three input-independent SSD heads, that is, by $n = 3$ states with three decay rates. **(a)** The delay maps the past window to the future window by the identity (rank 8). **(b)** The system maps it through its state, OC (rank 3), a blurred version of the identity. **(c)** A unit-energy past input with $C\mathbf{x} = 0$ leaves the state at zero. **(d)** The system's future outputs are then all zero, while the delay's are $\mathbf{x}$, so the error has energy 1 whatever the weights. Discussed in §3.3.

Proposition 2 (reach is a count). Along a route with scan order ρ (a permutation of the positions) and lag set $\mathcal{P}_{\text{route}}$, a token enters the state either at its own position or, through lag p, at decay distance zero at position $\rho^{-1}(\rho(s) + p)$, and then decays like any other entry. If a layer resolves r decay steps (§3.1), the query at t can single out by position only the sources $s \neq t$ with $\rho(t) - \rho(s) \in \{1, \ldots, r\}$ (through lag 0) or $\rho(t) - \rho(s) - p \in \{0, \ldots, r\}$ for some $p \in \mathcal{P}_{\text{route}} \setminus \{0\}$ (through lag p). That is at most $r + (r+1)(|\mathcal{P}_{\text{route}}| - 1) = (r+1)|\mathcal{P}_{\text{route}}| - 1$ positions per route, and summing over routes proves the bound. The set depends on ρ, but its size does not (Table C.1). □

A decay-free gradient path. For $p \in \mathcal{P}$, $\partial \mathbf{o}_t / \partial \mathbf{v}_{t-p}$ contains the term $\Delta^{(i)}_{t-p}\, \mathbf{q}_t^\top \mathbf{k}^{(i)}_{t-p}$ of the lagged write, which carries no decay factor, in addition to the path through the write of lag 0, which is attenuated by $\alpha_{t-p+1:t}$. The gradient to a lag-p input therefore does not vanish like α^p. This is the mechanism of NARX recurrent networks (Lin et al., 1996) and dilated RNNs (Chang et al., 2017), and the peaks of Figure C.2b and Figure H.3 measure it.

Proposition 4 (PACF lag selection)**.** *Let (x_t) be a stationary linear process with i.i.d. innovations of finite fourth moment and a positive definite autocorrelation matrix up to lag K, and let $\hat{\pi}_k$ be the sample partial autocorrelations from n_{obs} observations. Then $\sqrt{n_{obs}}(\hat{\pi}_k - \pi_k) \Rightarrow \mathcal{N}(0, \sigma_k^2)$ jointly in $k \leq K$, with $\sigma_k = 1$ for $k > m$ if (x_t) is an autoregression of finite order m. For $n_{obs} \geq \left(\sigma_{\max}\, z_{1-\gamma/(2K)}/\eta\right)^2$ with $\sigma_{\max} = \max_{k\leq K} \sigma_k$ and z_q the standard normal q-quantile, asymptotically $\Pr(\max_{k\leq K} |\hat{\pi}_k - \pi_k| \leq \eta) \geq 1 - \gamma$. On that event, the threshold θ selects every lag with $|\pi_k| > \theta + \eta$, and every selected lag has $|\pi_k| > \theta - \eta$.*

Proof. Bartlett's formula gives the joint normal limit of the sample autocorrelations (Brockwell & Davis, 1991, Thm. 7.2.1). The map from autocorrelations to π_k (Durbin–Levinson) is smooth where

the autocorrelation matrix is positive definite, so the delta method gives the limit of $\hat{\pi}_k$. For an autoregression of order m and $k > m$, $\sigma_k = 1$ (Quenouille; Brockwell & Davis, 1991, §8.1). Each lag exceeds η with limiting probability at most γ/K, and a union bound over K lags finishes the uniform statement. The selection statements are the triangle inequality. □

With the threshold $z_{1-0.05/(2K)}/\sqrt{n_{\text{obs}}}$ of Appendix H.2 and white noise, the family-wise rate of spurious lags is asymptotically at most 0.05, matching the 5.0% measured there.

C POSITIONAL REACH

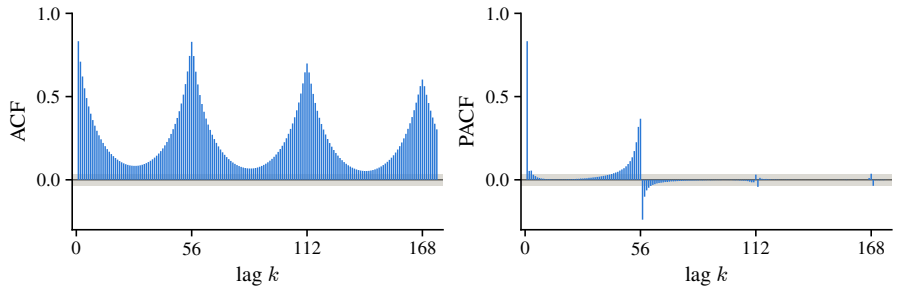
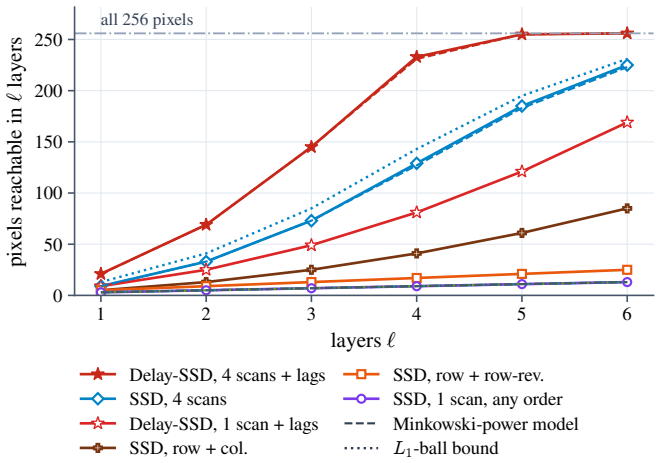
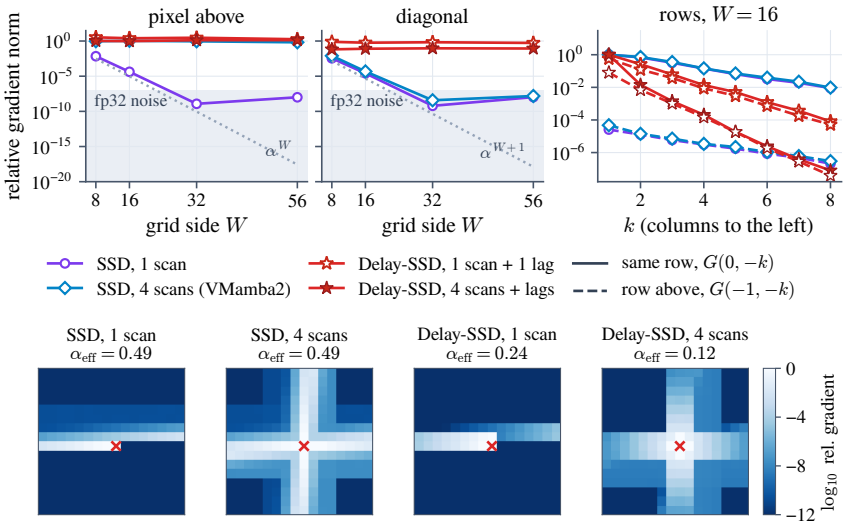


Figure C.1: Autocorrelation (left) and partial autocorrelation (right) of validation images of the 100-class ImageNet subset, resized to the 56×56 resolution of our first stage, flattened in row-major order and averaged over images, with RGB treated as three channels; shaded: the $\pm 1.96/\sqrt{3136}$ band. The autocorrelation peaks at every multiple of the row stride $W = 56$. The partial autocorrelation is concentrated at lag 1 and on the lags leading up to W, peaking at W itself, and is negligible beyond one row, because once the previous row is known, rows further back add almost nothing. The column-major scan gives the same picture with H in place of W. Discussed in §1 and §2.4.

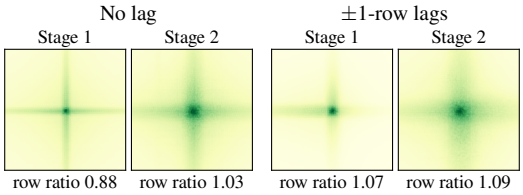


(a) Combinatorial reach against depth.

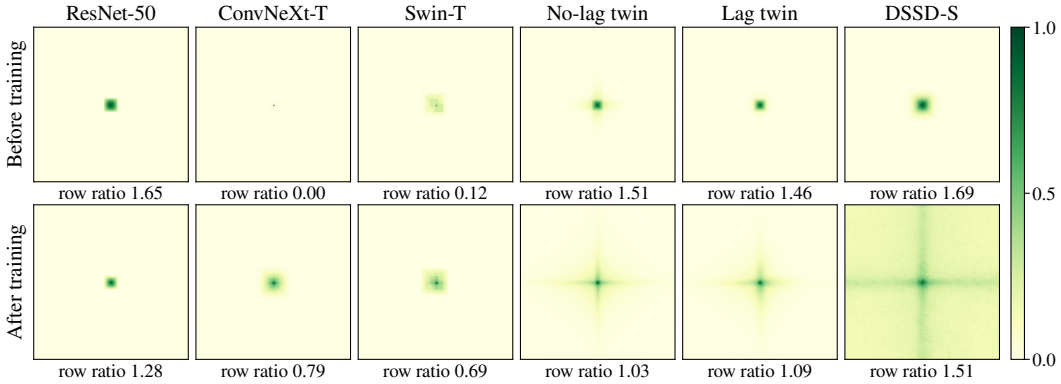


(b) Gradient reach of one untrained layer.

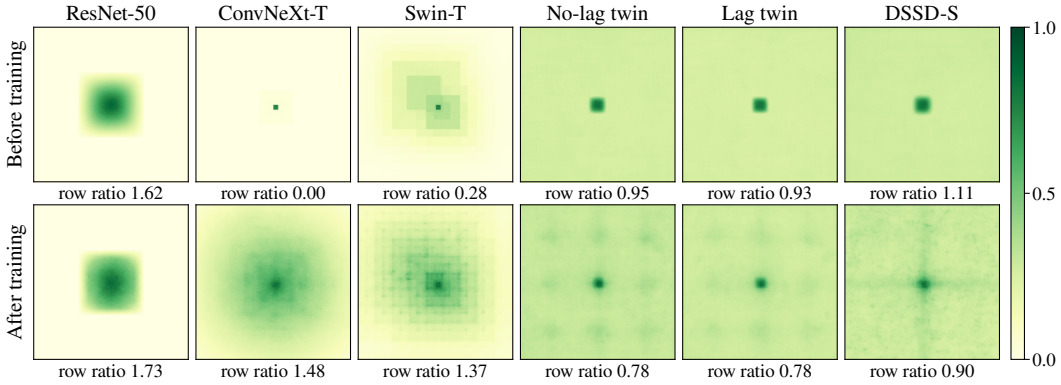
Figure C.2: Positional reach in two views. **(a)** The number of pixels of a 16×16 grid that can influence a query within ℓ layers when one layer resolves $r = 2$ decay steps along each scan, by breadth-first search over the one-layer sets; dashed and dotted lines are the sums-of-offsets and L_1-ball models. Row-major, anti-diagonal zigzag, 4×4-window and Hilbert-curve scans give the same curve, since a single causal scan reaches $2\ell + 1$ pixels in any order, whereas scan directions and lags multiply the one-layer reach. The model bounds *positional* reach only; content addressing can match any earlier token in one hop, which is why the tasks of §3.2 use i.i.d. tokens. **(b)** Relative Jacobian norm at initialization from the query to the pixel above (left), to the diagonal (middle) and along the current and previous rows at $W = 16$ (right), with the fp32 reference path, no convolution and VMamba2's decay initialization $\alpha \approx 0.49$; shaded: the fp32 noise floor. Bottom: $\log_{10}$ maps, annotated with the effective per-step decay $\alpha_{\text{eff}} = \alpha^{|\mathcal{P}|}$. With lags the previous-row profile coincides with the same-row profile, so the state sees a band, not a line. Discussed in §3.1.



(a) The parameter-matched DSSD-N twins without and with ±1-row lags after training, stages 1–2.



(b) Stage 2, before (top) and after (bottom) training.



(c) Stage 4 (the backbone output), before and after training.

Figure C.3: Effective receptive fields in the style of Liu et al. (2024b): the gradient of the center feature of a stage's output with respect to the input image, rectified, summed over channels, averaged over 50 images at 1024×1024 and shown max-normalized as $x^{0.2}$. The number under each map is the row ratio: the gradient mass in the token rows directly above and below the center row divided by the mass in the center row, where a row is one token of that stage, the unit a ±1-row lag moves by. The DSSD-N twins are the parameter-matched pair of Appendix G trained on the 100-class subset, with images from its test split; ResNet-50, ConvNeXt-T, Swin-T (`timm` weights) and the reported DSSD-S checkpoint use ImageNet-1k validation images. With lags, the trained stage-1 row ratio rises from 0.88 to 1.07 and the stage-2 ratio from 1.03 to 1.09, so the lags move some gradient mass into the adjacent rows, most at stage 1, where the lag is one pixel row of an image patch. Both twins and DSSD-S form a cross of the four scan directions. DSSD-S spreads further because its stage 2 has 12 blocks instead of 3 and its decays are slower, taking a median of 32–34 steps at the step size set by the bias to fall to 1% in stages 1–2, against 8–17 for the twins. At stage 4, after the attention stages, the fields of all three Delay-SSD models are diffuse, with row ratios of 0.78–0.90. The hot spots about 300 px from the center of the trained twins' stage-4 maps come from RoPE at a resolution the models were not trained at. At 1024^2 stage 3 has 64×64 tokens against 14×14 in training, and at an offset of about 19 tokens the two fastest RoPE frequencies return nearly to their phase at offset 0. At 1536^2 the hot spots stay at the same offset, with a second ring at twice it. Discussed in §3.1.

Table C.1: Minimum number of layers for the pixel at offset $(\Delta i, \Delta j)$ to influence the query at the center of a 16×16 grid, one layer resolving $r = 2$ decay steps per scan (∞: not within 6 layers). Offsets that no scan has just before the query, every diagonal included, need two cross-scan layers, whereas a ± 1-row lag reaches into the previous and next rows in one. The permuted scan orders reach the same *number* of pixels as the row-major scan, in a different place. Discussed in §3.1.

architecture	$(-1, 0)$	$(-1, -1)$	$(-1, -2)$	$(-1, -4)$	$(0, -1)$	$(-2, 0)$	$(1, 1)$
SSD, 1 scan	∞	∞	∞	∞	1	∞	∞
SSD, bidirectional row scan	∞	∞	∞	∞	1	∞	∞
SSD, row + column scan	1	2	2	3	1	1	∞
SSD, 4 scans (VMamba2)	1	2	2	3	1	1	2
Delay-SSD, 1 scan + lags	1	1	1	2	1	2	∞
Delay-SSD, 4 scans + lags	1	1	1	2	1	1	1
SSD, 1 scan (anti-diag. zigzag)	∞	∞	∞	∞	∞	∞	∞
SSD, 1 scan (4×4 windows)	∞	∞	∞	∞	∞	∞	∞
SSD, 1 scan (Hilbert curve)	∞	∞	∞	∞	1	∞	∞

D SYNTHETIC ROUTING TASKS: PROTOCOL, TABLE AND CONTROLS

Protocol. Models are an embedding, ℓ blocks of a pre-norm mixer and a pre-norm FFN, and a per-pixel head, trained on 16×16 grids of i.i.d. tokens (a vocabulary of 8 tokens, binary tokens for block-sum, or 32 key–value tokens for row-recall) with a per-pixel target that depends only on the pixels the task names. By default, models use $d_{\text{model}} = 64$, $N = 16$, four heads per scan direction with $\mathbf{k}$ and $\mathbf{q}$ shared within a direction and Δ per head, no convolution, no positional embedding and the fp32 reference scan, and train with AdamW at learning rate 3×10^{-3} for 2000 steps at batch 128. Accuracy is measured on 2048 fresh grids at each of 20 evaluations and we report the best of them, so "steps to 99%" has a resolution of 100 steps. We run three seeds per configuration on the grid tasks and the one-layer configurations of the bridge experiment, which adds the convolution back (Appendix D.2), and two on the positional-embedding sweeps and the deeper bridge configurations. The baseline with no lags is the SSD recurrence. Our kernels are checked against `mamba_ssm`'s Triton kernels forward and backward, and VMamba2's own Triton implementation runs at 0.95–$0.99\times$ the device time of our four-scan mixer at the vision models' stage shapes. Sharing $\mathbf{k}$ across lags costs no accuracy on the two tasks that load the state hardest (Table D.2).

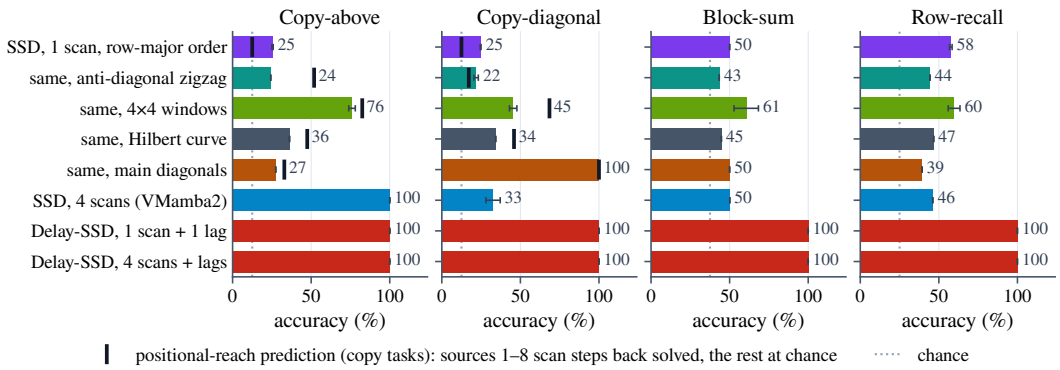


Figure D.1: A scan order moves the reach but does not enlarge it. Each permuted order is run as a single-scan SSD with exactly the parameters and FLOPs of the row-major single scan (3 seeds). The anti-diagonal zigzag, 4×4 windows and a Hilbert curve solve none of the four tasks in one layer, the best being 4×4 windows on copy-above at 76.0%. An order along the main diagonals, which makes the copy-diagonal source the previous token, solves copy-diagonal (100%) and stays at 27–50% on the other three tasks, while the row-major scan with one lag solves all four. Vertical ticks: the positional-reach prediction for the copy tasks. Discussed in §3.1.

Table D.1: Synthetic routing tasks without positional embedding (16×16 grids, $N = 16$, no convolution, 3 seeds). FLOPs are training FLOPs per sample on the reference scan path; "steps to 99%" is the median over the seeds that got there, and "–" marks a configuration no seed brought to 99%. The "bidirectional row scan" runs the row scan forward and reversed. Like a single scan it does not reach the pixel above, which a single scan does not reach with two layers either. Discussed in §3.2.

task	chance (%)	mixer	ℓ	acc. (%)	params (K)	train GFLOPs/sample	steps to 99%
Copy-above	12.5	SSD, 1 scan	1	25.4 ± 0.5	29	0.055	–
		SSD, 1 scan	2	30.9 ± 0.8	56	0.110	–
		SSD, bidirectional row scan	1	25.7 ± 0.3	31	0.070	–
		SSD, row + column scan	1	100.0 ± 0.0	31	0.070	100
		SSD, row + column scan	2	100.0 ± 0.0	61	0.139	100
		SSD, 4 scans (VMamba2)	1	100.0 ± 0.0	36	0.100	100
		SSD, 4 scans (VMamba2)	2	100.0 ± 0.0	70	0.198	100
		Delay-SSD, 1 scan + 1 lag	1	100.0 ± 0.0	30	0.067	100
		Delay-SSD, 4 scans + lags	1	100.0 ± 0.0	46	0.191	100
Copy-diagonal	12.5	SSD, 1 scan	1	24.7 ± 0.3	29	0.055	–
		SSD, 1 scan	2	30.0 ± 1.0	56	0.110	–
		SSD, bidirectional row scan	1	25.2 ± 0.3	31	0.070	–
		SSD, row + column scan	1	28.0 ± 3.1	31	0.070	–
		SSD, row + column scan	2	100.0 ± 0.0	61	0.139	200
		SSD, 4 scans (VMamba2)	1	32.6 ± 4.6	36	0.100	–
		SSD, 4 scans (VMamba2)	2	100.0 ± 0.0	70	0.198	200
		Delay-SSD, 1 scan + 1 lag	1	100.0 ± 0.0	30	0.067	200
		Delay-SSD, 4 scans + lags	1	100.0 ± 0.0	46	0.191	100
		Delay-SSD, 4 scans + lags	2	100.0 ± 0.0	91	0.381	100
Block-sum	37.5	SSD, 1 scan	1	50.0 ± 0.1	28	0.055	–
		SSD, 1 scan	2	50.0 ± 0.1	56	0.110	–
		SSD, bidirectional row scan	1	50.0 ± 0.1	30	0.070	–
		SSD, row + column scan	1	50.1 ± 0.0	30	0.070	–
		SSD, row + column scan	2	100.0 ± 0.0	60	0.139	300
		SSD, 4 scans (VMamba2)	1	50.1 ± 0.0	35	0.099	–
		SSD, 4 scans (VMamba2)	2	100.0 ± 0.0	70	0.198	200
		Delay-SSD, 1 scan + 1 lag	1	100.0 ± 0.0	29	0.067	300
		Delay-SSD, 4 scans + lags	1	100.0 ± 0.0	45	0.190	100
		Delay-SSD, 4 scans + lags	2	100.0 ± 0.0	90	0.380	100
Row-recall	25.0	SSD, 1 scan	1	57.6 ± 0.8	30	0.055	–
		SSD, 1 scan	2	54.9 ± 7.1	58	0.110	–
		SSD, bidirectional row scan	1	50.2 ± 14.3	32	0.070	–
		SSD, row + column scan	1	49.3 ± 4.4	32	0.070	–
		SSD, row + column scan	2	99.4 ± 0.4	62	0.139	1800
		SSD, 4 scans (VMamba2)	1	46.1 ± 0.1	37	0.099	–
		SSD, 4 scans (VMamba2)	2	100.0 ± 0.0	72	0.198	800
		Delay-SSD, 1 scan + 1 lag	1	100.0 ± 0.0	31	0.066	300
		Delay-SSD, 4 scans + lags	1	100.0 ± 0.0	47	0.190	300
		Delay-SSD, 4 scans + lags	2	100.0 ± 0.0	92	0.380	200

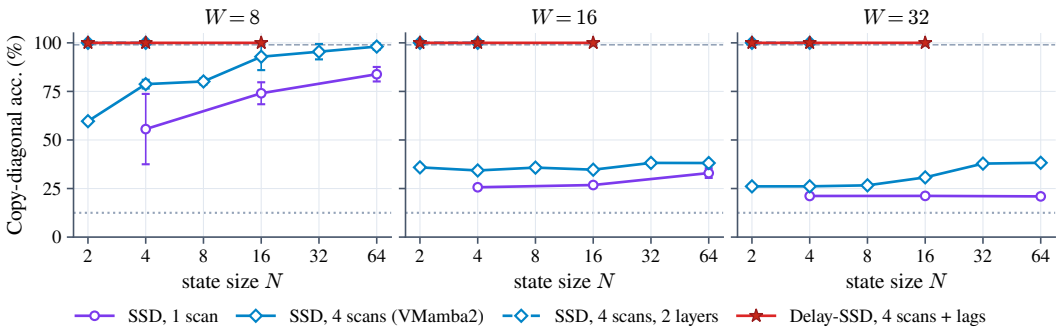
Table D.2: Sharing $\mathbf{k}$ across lags costs no accuracy on the two tasks that load the state hardest and removes the per-lag parameter overhead. Discussed in Appendix D.

task	$\mathbf{k}$ across lags	acc. (%)	params (K)	steps to 99%
Copy-diagonal	separate (default)	100.0 ± 0.0	46	100
	shared	100.0 ± 0.0	38	100
Row-recall	separate (default)	100.0 ± 0.0	47	300
	shared	100.0 ± 0.0	39	300

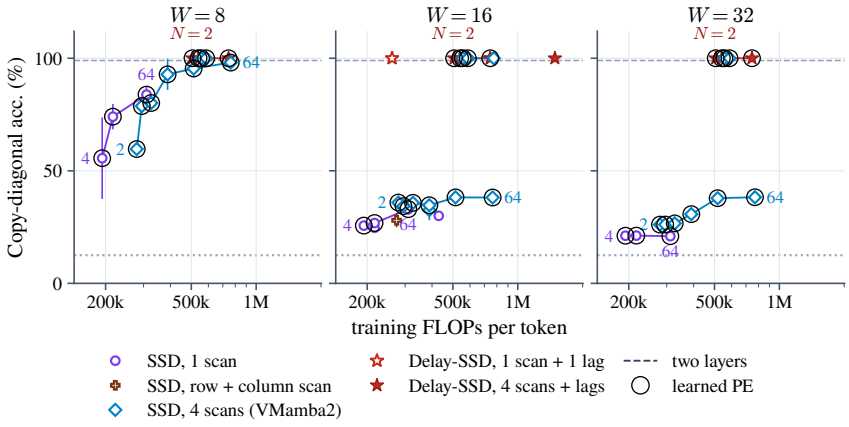
D.1 With a learned positional embedding: a training wall, not a state wall

If $\mathbf{k}_s$, $\mathbf{q}_t$ and Δ_s may depend on position, a single SSD layer can solve copy-diagonal by position, but not with the obvious construction. With $e_{\mathrm{col}(s)}$ the standard basis vector indexed by the column of s, the choice $\mathbf{k}_s = e_{\mathrm{col}(s)}$, $\mathbf{q}_t = e_{\mathrm{col}(t)-1}$ selects *every* earlier token of column $j-1$, and the most recent of them is the current row's left neighbor at weight α^1, not the previous row's at α^{W+1}. A correct construction gates Δ_s by the row parity so that a head writes only in the rows the query does not read from; under one $(\mathbf{k}, \mathbf{q})$ pair per scan direction this costs $N = W$ and three gated heads, and folding the parity into the state gives a one-head solution at $N = 2W$. Every construction we found needs a state that grows with W and a decay with α^W neither negligible nor saturating, i.e. one that no longer separates rows. We verify the constructions by setting the trained architecture's weights by hand. In fp32, $N = 2W$ reaches 100% at $W \in \{8, 16, 32\}$ and $N = W$ does so at $W \in \{8, 16\}$, reaching 85.6% at $W = 32$ for the numerical reason just given. A lag needs $N = O(1)$ and leaves the decay unconstrained.

At the budget used for every other configuration, the baseline plateaus at $34.70 \pm 0.10\%$ with $N = 16$ and $38.16 \pm 0.54\%$ with $N = 64$ at $W = 16$, flat in N (Figure D.2a, Table D.3). The same configuration reaches $99.97 \pm 0.02\%$ within 2000 steps at learning rate 10^{-2}, and at 3×10^{-3} with 8000 steps it leaves the plateau at step 2000–2500 and reaches 100%; at 10^{-2} and 8000 steps even $W = 32$ is solved with $N = W/2 = 16$. The plateau is a saddle the budget ends on, not a capacity limit. The initialization explains the ordering, since the construction wants $\alpha^{W+1} \approx 0.1$–0.3 and VMamba2's initialization starts five orders of magnitude below it. State is not the binding constraint in the other direction either, because a two-layer SSD solves copy-diagonal for every width at $N = 2$. One layer with a lag also solves it at $N = 2$, at the first budget tried and without a decay that couples rows, for $0.91\times$ the training FLOPs of the least expensive two-layer SSD that solves it (Figure D.2b).



(a) Accuracy against state size at the fixed budget.



(b) Accuracy against training compute.

Figure D.2: Copy-diagonal with a learned positional embedding. **(a)** At learning rate 3×10^{-3} and 2000 steps (2 seeds), the one-layer SSD baseline is limited by its budget, not its state. It reaches 93–98% at $W = 8$ for $N \geq 16$ and 26–39% at $W = 16$ and 32 for every $N \leq 64$, flat in N, whereas one Delay-SSD layer and a two-layer SSD both solve every width at $N = 2$. **(b)** Every configuration at the same budget against its training compute, averaged over seeds. Open rings mark a learned positional embedding, dashed lines two layers and numbers the N at the ends of each sweep, with $N = 2$ at the Delay-SSD point. Discussed in §3.2 and Appendix D.1.

Table D.3: Smallest state N^* reaching 95% on copy-diagonal with a learned positional embedding, and accuracy at $N = 16$, at the fixed budget of Figure D.2a. Here N^* is a property of that budget, since the $W = 16$, $N = 16$ configuration reaches 99.97% at learning rate 10^{-2} and 100% with 8000 steps at 3×10^{-3}. A two-layer SSD reaches 95% at $N = 2$ for every W (Figure D.2a). Discussed in §3.2.

mixer	N^* (W=8)	N^* (W=16)	N^* (W=32)	acc. N=16 (W=8)	acc. N=16 (W=16)	acc. N=16 (W=32)
SSD, 1 scan	> 64	> 64	> 64	74.1 ± 5.7	26.8 ± 0.6	21.2 ± 0.0
SSD, 4 scans (VMamba2)	32	> 64	> 64	92.8 ± 6.9	34.7 ± 0.1	30.7 ± 0.3
Delay-SSD, 4 scans + lags	2	2	2	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0

D.2 Bridge to the full vision configuration

The mixer of the full vision configuration carries a 3×3 depthwise convolution, which routes the distance-one neighbors by itself, so the question is what a lag adds *on top of* it. We re-run the copy tasks at distance 2 and 4 with the convolution in place, at depths 1, 2 and 4, with three lag sets (± 1 row, ± 2 rows, both) or a wider convolution (Figure D.3, Table D.4).

At distance 2, one SSD layer with the 3×3 convolution reaches $93.3 \pm 3.5\%$ on $y = x[i-2, j-2]$ and never crosses 99%, while every lag set is at 100% within 100–200 steps. A 5×5 convolution, whose window covers that source, also closes this gap, at $+2.8\%$ parameters and $1.30\times$ training FLOPs against $+28\%$ and $1.78\times$ for the ± 1-row lags. No single layer suffices at distance 4, where SSD reaches 23–58% depending on the convolution width and all three lag sets sit at chance. A lag adds a band around the positions just before the query in scan order, not distance. At depth 2 the lag adds reliability and speed. SSD with the 3×3 convolution is at $98.7 \pm 1.0\%$ with one of two seeds past 99% (median 1300 steps), while every lag variant is at 100% in 200–650 steps, and ± 2 rows is the better default at distance 4 (200 vs. 650 steps). That the lagged models sit at chance at distance 4 and depth 1 is consistent with the shared decay, since the three terms of $\mathcal{P} = \{0, \pm W\}$ at the same Δ give an effective per-step decay of $\alpha^3 \approx 0.12$ at VMamba2's initialization (Figure C.2b), a shorter horizontal horizon.

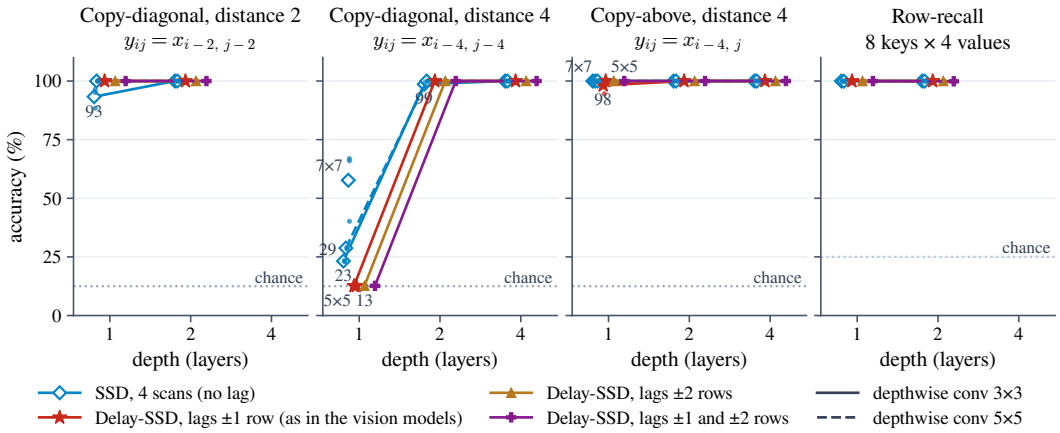


Figure D.3: Accuracy against depth with the depthwise convolution of the full vision configuration in place, one panel per task, on 16×16 grids with $N = 16$ and no positional embedding (3 seeds at one layer, 2 deeper). Seed points sit beside each mean; dotted: chance. Color is the lag set; the dashed line and in-place labels are the wider-convolution controls. At distance 2 one SSD layer with the 3×3 convolution reaches 93%, whereas every lag set and a 5×5 convolution reach 100%. At distance 4 no single layer solves the task, lags included. At two layers the lagged models reach 100% in 200–650 steps, against 1300 for the convolution alone. Discussed in §3.2.

Table D.4: Bridge to the full vision configuration: copies at distance 2 and 4 and row-recall with its 3×3 depthwise convolution, against a wider convolution and three lag sets, at depths 1, 2 and 4. †Only some seeds reached 99%; –: none did. Discussed in §3.2.

task	mixer	conv	acc. ℓ=1	acc. ℓ=2	acc. ℓ=4	99% ℓ=1	99% ℓ=2	99% ℓ=4
Copy-diagonal, distance 2	SSD, 4 scans (no lag)	3×3	93.3 ± 3.5	100.0 ± 0.0	–	–	100	–
	SSD, 4 scans (no lag)	5×5	100.0 ± 0.0	100.0 ± 0.0	–	100	100	–
	Delay-SSD, lags ±1 row	3×3	100.0 ± 0.0	100.0 ± 0.0	–	100	100	–
	Delay-SSD, lags ±2 rows	3×3	100.0 ± 0.0	100.0 ± 0.0	–	200	100	–
	Delay-SSD, lags ±1, ±2 rows	3×3	100.0 ± 0.0	100.0 ± 0.0	–	200	100	–
Copy-diagonal, distance 4	SSD, 4 scans (no lag)	3×3	23.2 ± 0.1	98.7 ± 1.0	100.0 ± 0.0	–	1300†	450
	SSD, 4 scans (no lag)	5×5	28.8 ± 0.4	100.0 ± 0.0	100.0 ± 0.0	–	100	100
	SSD, 4 scans (no lag)	7×7	57.7 ± 12.4	–	–	–	–	–
	Delay-SSD, lags ±1 row	3×3	12.6 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	–	650	550
	Delay-SSD, lags ±1 row	5×5	12.6 ± 0.0	–	–	–	–	–
	Delay-SSD, lags ±2 rows	3×3	12.6 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	–	200	200
	Delay-SSD, lags ±1, ±2 rows	3×3	12.6 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	–	200	200
Copy-above, distance 4	SSD, 4 scans (no lag)	3×3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	1100	250	200
	SSD, 4 scans (no lag)	5×5	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	300	100	100
	SSD, 4 scans (no lag)	7×7	100.0 ± 0.0	–	–	200	–	–
	Delay-SSD, lags ±1 row	3×3	98.2 ± 2.6	100.0 ± 0.0	100.0 ± 0.0	750†	200	150
	Delay-SSD, lags ±1 row	5×5	100.0 ± 0.0	–	–	300	–	–
	Delay-SSD, lags ±2 rows	3×3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	500	100	100
	Delay-SSD, lags ±1, ±2 rows	3×3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	900	100	100
Row-recall	SSD, 4 scans (no lag)	3×3	100.0 ± 0.0	100.0 ± 0.0	–	500	500	–
	SSD, 4 scans (no lag)	5×5	99.9 ± 0.0	100.0 ± 0.0	–	800	700	–
	Delay-SSD, lags ±1 row	3×3	100.0 ± 0.0	100.0 ± 0.0	–	500	600	–
	Delay-SSD, lags ±2 rows	3×3	100.0 ± 0.0	100.0 ± 0.0	–	500	550	–
	Delay-SSD, lags ±1, ±2 rows	3×3	100.0 ± 0.0	99.9 ± 0.1	–	600	850	–

E ONE-DIMENSIONAL DELAY TASK

The task is to recover $x_{t-\tau}$, the input at target-to-source distance τ (§2.1), from a band-limited white signal of 4096 steps with bandwidth $\beta = 0.25$ cycles per sample unless stated. A relative RMSE of 1 is that of the zero predictor, and because the generator has no per-sequence offset, the baseline's asymptote is the true chance level. SSD is swept over 2, 8 or 32 heads in 1, 2 or 4 layers with $N \in [4, 128]$, and Delay-SSD is one layer of 8 heads with a lag at τ (Table E.1).

Sharpness of the lag. The residual with the lag placed at $p \neq \tau$ is the SSD's own error at distance $|p - \tau|$, which at $\tau = 512$ is already 0.60 four samples early and 0.99 four samples late.

Learned lags. With two-tap linear interpolation and their own optimizer group, continuous lags recover the true lag by gradient descent. For a true delay of 64 samples, a lag initialized at 48 converges to 63.9 and reaches a relative RMSE of 0.0049, compared with 0.4352 fixed at 48 and 0.0047 fixed at the truth. The useful knob is how far the lag may move per step, and about 0.3 samples per step worked across a $10\times$ range of multipliers. The gradient is local, with a convergence radius of about one autocorrelation width of the signal.

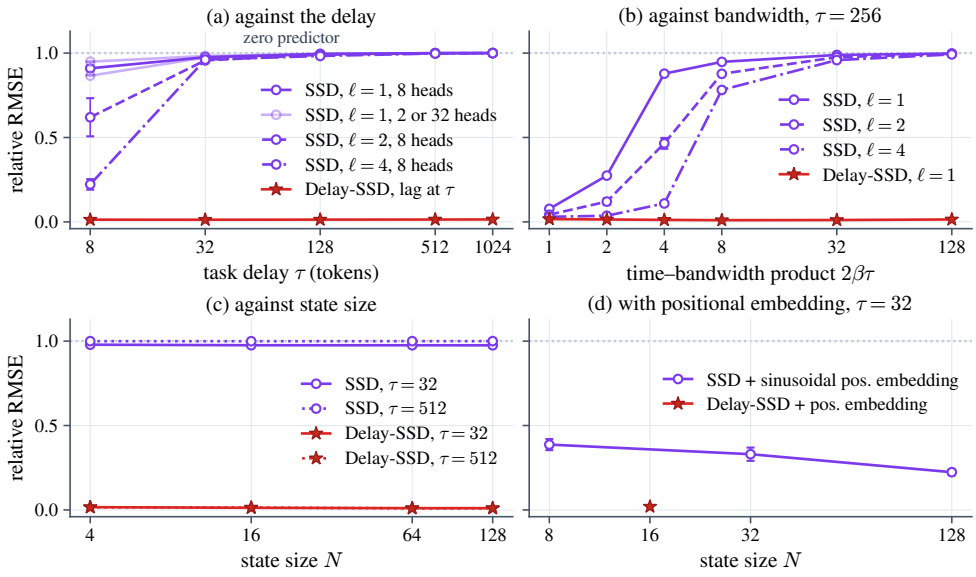


Figure E.1: 1-D delay task, relative RMSE against the zero predictor (2 seeds). (a) Against τ; (b) against the time–bandwidth product at $\tau = 256$; (c) against state size, which does not help the baseline; (d) with a sinusoidal positional embedding, where state helps slowly, from 0.39 at $N = 8$ to 0.22 at $N = 128$ for $\tau = 32$. Discussed in §3.3.

Table E.1: 1-D delay task: best SSD configuration per delay τ against one Delay-SSD layer with the matching lag $p = \tau$. Discussed in §3.3.

τ	best SSD rel-RMSE	best SSD config	its train FLOPs vs DSSD	DSSD rel-RMSE
8	0.222	8 heads, ℓ=4	2.6×	0.013
32	0.918	32 heads, ℓ=4	2.8×	0.013
128	0.983	8 heads, ℓ=4	2.6×	0.013
512	0.997	8 heads, ℓ=4	2.6×	0.014
1024	1.000	8 heads, ℓ=4	2.6×	0.014

F Cost

FLOPs conventions. Table 1 counts our scan in its recurrent form: one multiply-add per state entry and token for the decay, one for each lag's write $\Delta\,\mathbf{k}\mathbf{v}^\top$ and one for the read-out $\mathbf{q}^\top\mathbf{S}$. This gives 1.57, 4.98 and 14.09G for DSSD-N, T and S, and 1.33, 4.48 and 12.49G without lags. Counting the chunked kernels as executed, with their dense 64×64 intra-chunk products, gives 2.15, 5.85 and 16.30. VMamba's fvcore handle would give 2.15, 6.14 and 17.57 because it charges a flat 9 per state entry, Mamba-1's cost, where every entry has its own $\exp(\Delta A)$; this overcounts SSD, whose decay is one scalar per head. The published comparator FLOPs follow no single convention, so we recounted them with each repository's own tool on CPU. VMamba, whose scan is 1–2% of its total at $N = 1$, and Swin reproduce within 1%. VSSD's count includes only $1/n_h$ of its non-causal SSD products and is about 4–5% low. MambaVision's counter misses the selective scan and the fused attention (0.1–0.2G). LocalVMamba applies VMamba's 9 per state entry at $N = 16$, which is 34% of its 5.7G. Mamba2D gives no FLOPs code and its published numbers exceed every count of its released model by 0.2–1.0G. For ADE20K, the standard UperNet decode head that Mamba2D's configurations inherit costs 840–845G alone at 512×2048, more than every Mamba2D total in Table I.2, and the same count reproduces VMamba-T's 948.8G exactly. Tables 1 and I.2 keep the published values, even where we could not reproduce them.

Mixer layout and input size. Stages 3–4 of every Delay-SSD backbone use global attention (§2.5), and its $\mathbf{Q}\mathbf{K}^\top$ and attention-times-value products grow with the square of the token count. At 224^2 stage 3 has 196 tokens and these products are small, but at the 512×2048 input at which segmentation

FLOPs are reported it has $4{,}096$, and they add 50, 100 and 184G to DSSD-N, T and S, about as much as the rest of the backbone. This is why our ADE20K totals in Table I.2 exceed VMamba's and Swin's, whose backbones grow in proportion to the number of pixels; in our models and in VMamba's the UperNet decode head (840–845G) is still the larger part. As Table F.1 shows, with attention in all four stages the same backbone costs 1.4–$2.0\times$ as much as ours at 224^2 and 9–$12\times$ as much at the detection and segmentation inputs. With Delay-SSD in all four stages it costs 0.46–$0.64\times$ as much as ours at 512×2048, but 2–12% more at 224^2, where the last two stages have only 196 and 49 tokens. This saving is in FLOPs and carries over to time only at the dense inputs, because below about 1.3×10^4 tokens in training, and about 7×10^3 at inference with the fused kernel, attention runs faster than our mixer on the GPU (Table F.2, Figure F.2), and stage 1 has $3{,}136$ tokens at 224^2 but $65{,}536$ at 512×2048.

Table F.1: Backbone FLOPs (G, multiply-accumulates) of the Delay-SSD layout with the token mixer of stages 1–2 and 3–4 swapped. Widths, depths and the rest of the block are those of Table 1, and FLOPs are counted as there (the scan in its recurrent form); attention is global, with 2-D RoPE and head dimension 32. Detection and segmentation FLOPs are conventionally reported at 1280×800 and 512×2048. Params: backbone only. Shaded: the layout we train.

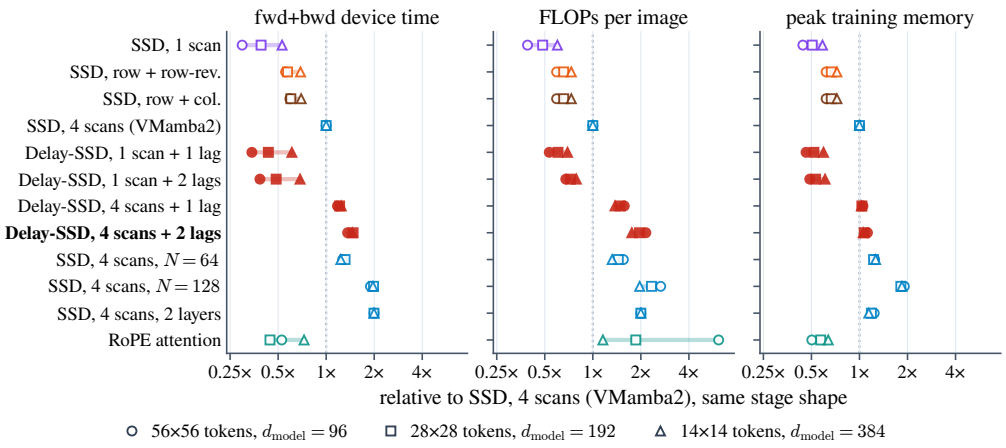
Model	Stages 1–2 / 3–4	Params	224^2	1280×800	512×2048
DSSD-N	Attention / Attention	5.8M	3.2	952	998
DSSD-N	Delay-SSD / Attention	6.0M	1.6	77	80
DSSD-N	Delay-SSD / Delay-SSD	6.6M	1.8	36	37
DSSD-T	Attention / Attention	23.2M	8.3	1,942	2,035
DSSD-T	Delay-SSD / Attention	23.4M	5.0	192	199
DSSD-T	Delay-SSD / Delay-SSD	23.6M	5.2	106	109
DSSD-S	Attention / Attention	45.3M	20.0	3,940	4,126
DSSD-S	Delay-SSD / Attention	46.1M	14.1	454	470
DSSD-S	Delay-SSD / Delay-SSD	46.0M	14.4	294	301

Why twice the FLOPs cost $1.36\times$ the time. The lags multiply only the SSD core. In a per-kernel profile, the six kernels that carry per-lag work grow 1.7–$2.7\times$ but take 28% of the mixer's device time, while the cross-scan gather, the projections and the pointwise traffic do not move. An Amdahl fit, with device time as a fixed part plus a part proportional to the number of lags, gives a lag-proportional fraction of 0.32. The mixer is far from compute bound, reaching 12 TFLOP/s without lags and 18 with them, whereas FlashAttention reaches 106 on the same shape. The extra lags *improve* arithmetic intensity because they reuse tiles already resident in shared memory.

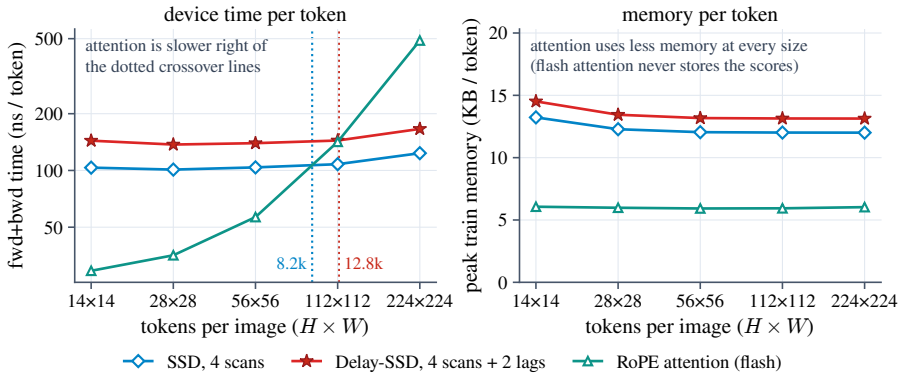
Table F.2: Token-mixer cost at the vision models' first stage shape (56×56 tokens, $d_{\text{model}} = 96$, batch 16, bf16, one RTX 5090), relative to the four-scan SSD mixer of VMamba2; shaded: ours. Device time is the profiler's forward+backward device time, and FLOPs are the analytic count of the kernels as executed, including the dense intra-chunk products; the recurrent count of Table 1 gives $1.65\times$ for the lags. A $4\times$ state and a second layer, the alternatives that could provide comparable reach, cost about $2\times$ the time, and attention is cheaper than all of them at this size but not above $\approx 1.3 \times 10^4$ tokens in training or $\approx 7 \times 10^3$ at inference with the fused kernel (Figures F.1b and F.2). The parameter overhead of the lags comes from per-lag $\mathbf{k}$ and Δ projections and disappears when $\mathbf{k}$ is shared. Discussed in §3.4; full table in Table F.3, other stage shapes in Figure F.1a.

Mixer	Params (K)	FLOPs	Device time	Peak memory
SSD, 4 scans (VMamba2)	47	$1.00\times$	$1.00\times$	$1.00\times$
SSD, 4 scans, $N = 64$	71	$1.55\times$	$1.27\times$	$1.24\times$
SSD, 4 scans, $N = 128$	120	$2.66\times$	$1.90\times$	$1.91\times$
SSD, 4 scans, 2 layers	93	$2.00\times$	$2.00\times$	$1.24\times$
Delay-SSD, 4 scans, lags $\pm W$	76	$2.15\times$	$1.36\times$	$1.12\times$
Delay-SSD, 1 scan, 1 lag	30	$0.53\times$	$0.34\times$	$0.46\times$
RoPE attention (flash)	37	$6.14\times$	$0.53\times$	$0.50\times$

Fused inference kernel. We also wrote a fused, inference-only forward kernel for the lagged scan in bf16 with lags $\{0, \pm W\}$ and $\{0, \pm H\}$, adapted from the persistent TileLang forward kernel



(a) Cost relative to the four-scan SSD mixer at the vision models' three stage shapes.



(b) Cost per token against tokens per image in training: where attention stops being the faster mixer.

Figure F.1: Token-mixer cost. **(a)** Forward+backward device time (left), FLOPs per image as executed (middle) and peak training memory (right) of each mixer relative to the four-scan SSD mixer at the same stage shape (log scale), one marker per stage shape. **(b)** Forward+backward device time per token (left) and peak training memory per token (right) at $d_{\text{model}} = 128$, bf16, with batch $\times$ tokens held at 50 176. The SSD mixers are flat in token count and attention is quadratic, so attention overtakes the four-scan SSD at 8.2×10^3 tokens and the two-lag mixer at $\approx 1.28 \times 10^4$, which is $\approx 113^2$ tokens or a ≈ 450 px input at this stem stride. Memory never crosses below 224^2. Discussed in §3.4.

Table F.3: Mixer cost at the vision models' first stage shape (56×56 tokens, $d_{\text{model}} = 96$; batch 16, bf16, one RTX 5090). "fwd+bwd device" is profiler device time, the column the claims use; the eager CUDA-event columns are launch-bound at this size. Ratios are relative to the four-scan SSD mixer. Discussed in §3.4.

mixer	params (K)	GFLOPs/img	fwd (ms)	fwd+bwd (ms)	fwd+bwd device (ms)	peak mem (MB)	train GFLOPs/img (counted)
SSD, 1 scan	26	0.25 (0.39×)	1.24 (0.76×)	4.17 (0.83×)	1.26 (0.30×)	205 (0.44×)	1.30 (0.47×)
SSD, bidir. row scan	33	0.39 (0.59×)	1.38 (0.85×)	4.44 (0.88×)	2.36 (0.56×)	287 (0.62×)	1.79 (0.65×)
SSD, row + column scan	33	0.39 (0.59×)	1.38 (0.85×)	4.65 (0.92×)	2.52 (0.59×)	287 (0.62×)	1.79 (0.65×)
SSD, 4 scans (VMamba2)	47	0.65 (1.00×)	1.62 (1.00×)	5.03 (1.00×)	4.24 (1.00×)	465 (1.00×)	2.76 (1.00×)
Delay-SSD, 1 scan, +1 lag	30	0.35 (0.53×)	1.39 (0.86×)	4.25 (0.85×)	1.45 (0.34×)	215 (0.46×)	1.67 (0.61×)
Delay-SSD, 1 scan, +2 lags	34	0.44 (0.68×)	1.33 (0.82×)	4.01 (0.80×)	1.63 (0.38×)	227 (0.49×)	2.04 (0.74×)
Delay-SSD, 4 scans, +1 lag	61	1.03 (1.57×)	1.76 (1.09×)	5.93 (1.18×)	5.00 (1.18×)	489 (1.05×)	4.22 (1.53×)
Delay-SSD, 4 scans, +2 lags (full vision config.)	76	1.40 (2.15×)	1.77 (1.09×)	6.50 (1.29×)	5.79 (1.36×)	522 (1.12×)	5.68 (2.06×)
SSD, 4 scans, N=64	71	1.01 (1.55×)	1.77 (1.09×)	6.04 (1.20×)	5.39 (1.27×)	578 (1.24×)	4.03 (1.46×)
SSD, 4 scans, N=128	120	1.74 (2.66×)	1.95 (1.21×)	8.45 (1.68×)	8.05 (1.90×)	888 (1.91×)	6.58 (2.39×)
SSD, 4 scans, 2 layers	93	1.30 (2.00×)	2.96 (1.83×)	9.74 (1.94×)	8.47 (2.00×)	575 (1.24×)	5.57 (2.02×)
RoPE attention	37	4.01 (6.14×)	1.14 (0.71×)	2.73 (0.54×)	2.24 (0.53×)	234 (0.50×)	13.74 (4.98×)

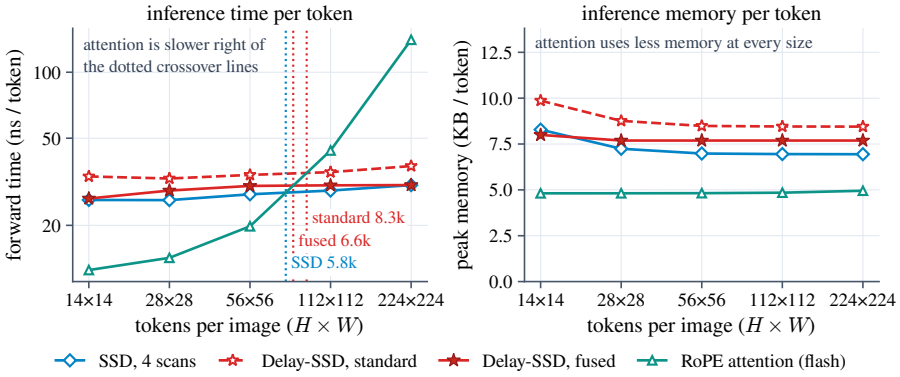


Figure F.2: Token-mixer cost per token at inference, the sweep of Figure F.1b without gradients: forward device time per token (left) and peak inference memory per token (right), with the two-lag mixer on the standard kernels and on the fused inference kernel (Appendix F, "Fused inference kernel"), all four mixers timed in one session. The fused kernel runs the mixer 1.12–1.26× faster, and attention overtakes it at $\approx 6.6 \times 10^3$ tokens, compared with $\approx 8.3 \times 10^3$ on the standard kernels and $\approx 5.8 \times 10^3$ for the four-scan SSD. Timed directly at intermediate sizes, the fused mixer and attention are equal at 7,168 tokens, about 85^2 or a ≈ 340 px input. Attention uses less memory at every size. Discussed in §3.4.

Table F.4: Full-model throughput and memory with and without lags for the parameter-matched DSSD-N pair (224×224, bf16, one RTX 5090; median of 30 timed calls). The CUDA-graph rows capture the six lag mixers only and their memory is not comparable. With the fused inference kernel (Appendix F, "Fused inference kernel") the lagged model runs eagerly at 4,176 and 4,073 img/s at batch 64 and 128, 0.95× and 0.91× the no-lag model; in the same session the standard kernels give 3,933 and 3,786 with lags and 4,396 and 4,471 without. Discussed in §3.4.

mode	batch	CUDA graph	no lags (img/s)	lags (img/s)	ratio	no lags (MB)	lags (MB)	ratio
inference	64	no	4143	3945	0.952×	500	720	1.44×
inference	64	yes	4387	3933	0.897×	558	556	1.00×
inference	128	no	4461	3779	0.847×	1147	1588	1.38×
inference	128	yes	4450	3771	0.848×	1041	1036	1.00×
train step	32	no	663	573	0.865×	1696	1888	1.11×
train step	32	yes	747	655	0.876×	1813	1808	1.00×
train step	64	no	899	793	0.882×	3302	3687	1.12×
train step	64	yes	1068	914	0.856×	3088	3078	1.00×

of Mamba-3 in `mamba_ssm`. It splits each scan into 512-token segments: one kernel computes each segment's state summary, with the step-size softplus, the lag roll and the in-chunk cumulative decay computed inline; the existing state-passing kernel carries the state across segments; and a second kernel walks each segment with its state in registers and computes the inter-chunk term, the

intra-chunk term of every lag and the skip. It uses internal chunks of 16 tokens with two heads per thread block, which halves the tensor-core work per token of the chunk-64 scan, and it was autotuned once on the RTX 5090. It matches the standard forward to within 0.6% of the output range. On this GPU it runs the scan 1.7–$2.2\times$ faster at detection and segmentation input sizes, and it raises compiled model throughput at 224×224 by 1.07–$1.19\times$ across DSSD-N, T and S and eager throughput of the parameter-matched pair by 1.06–$1.08\times$ (Table F.4). These numbers give a clearer view of what an optimized kernel can do than the general-purpose kernels, but there is still room for further optimization, since the cross-scan, cross-merge and layout copies around the scan, about half of the remaining mixer time at 1280×800, are not fused, and the backward pass is unchanged. The kernel is tuned for this GPU, whereas the comparators run their released kernels.

Against other backbones. Table F.5 and Figure F.3 measure every model of Table 1 whose code runs on our stack, with our mixers in the faster of the eager and CUDA-graphed settings and our inference on the fused kernel. At inference DSSD-T reaches 2.9k images/s and DSSD-S 1.1k, below VMamba, VSSD and Swin at each size. With the fused kernel the lagged models run at 0.83–$0.97\times$ the inference throughput of the same models without lags on the standard kernels; with the standard kernels on both, the lags cost 0.78–$0.85\times$ at inference and 0.76–$0.79\times$ in training. The rest of the gap comes from the backbone itself, since without lags DSSD-T runs at 3.1k images/s, still below VMamba-T. DSSD-S is nonetheless the most accurate model measured.

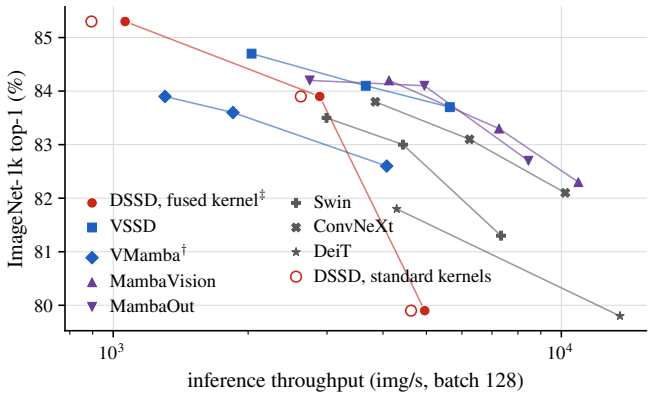


Figure F.3: ImageNet-1k top-1 against inference throughput (log scale) for the models of Table F.5; one line per family, from the largest to the smallest model. Filled Delay-SSD markers use the fused inference kernel, hollow ones the standard kernels. †Lower bound (see the table).

Table F.5: Model throughput and training memory on one RTX 5090 (450 W, bf16, `torch.compile`, synthetic 224×224 input; inference at batch 128, one training step at batch 64; median of 30 calls). Top-1 as published (ours: last-epoch EMA, Table 1). For Delay-SSD, ‡marks inference with the fused kernel (Appendix F, "Fused inference kernel"), tuned on this GPU; the "standard kernels" sub-row gives the same model on the standard kernels, and each parenthesis gives the ratio to the same model without lags ("no lags", not trained) on the same kernels. Training uses the standard kernels. †VMamba runs on the selective-scan kernel of `mamba_ssm` instead of its own compiled kernel, so its throughput is a lower bound. VSSD and MambaVision run from their official code and the other comparators from `timm`.

Model	Params	Top-1 (%)	Inference (img/s)	Training (img/s)	Train memory (GB)
Nano					
DSSD-N	7M	79.9	4,955‡	1,141 (0.77×)	3.0
standard kernels			4,620 (0.78×)		
no lags	7M	–	5,954	1,474	2.4
Tiny					
DSSD-T	27M	83.9	2,890‡	672 (0.79×)	5.1
standard kernels			2,622 (0.85×)		
no lags	26M	–	3,081	856	4.7
VSSD-T	24M	83.7	5,639	1,208	7.0
VMamba-T†	30M	82.6	4,077	1,120	5.2
MambaVision-T	32M	82.3	10,913	2,524	2.0
MambaOut-T	27M	82.7	8,443	2,359	3.7
Swin-T	28M	81.3	7,334	2,553	3.3
ConvNeXt-T	29M	82.1	10,202	2,633	3.5
DeiT-S	22M	79.8	13,512	3,798	1.9
Small					
DSSD-S	50M	85.3	1,063‡	244 (0.76×)	11.7
standard kernels			894 (0.82×)		
no lags	49M	–	1,091	318	10.9
VSSD-S	40M	84.1	3,660	778	11.0
VMamba-S†	50M	83.6	1,850	504	9.9
MambaVision-S	50M	83.3	7,259	1,769	2.7
MambaOut-S	48M	84.1	4,947	1,362	6.0
Swin-S	50M	83.0	4,432	1,485	5.3
ConvNeXt-S	50M	83.1	6,239	1,562	5.5
Base					
VSSD-B	89M	84.7	2,036	481	16.1
VMamba-B†	89M	83.9	1,305	368	13.2
MambaVision-B	98M	84.2	4,125	1,047	4.5
MambaOut-B	85M	84.2	2,745	828	8.1
Swin-B	88M	83.5	2,992	1,040	7.1
ConvNeXt-B	89M	83.8	3,846	1,095	7.4
DeiT-B	87M	81.8	4,292	1,368	4.0

G VISION ABLATIONS

All ablations use `timm/mini-imagenet` from the Hugging Face Hub, a 100-class subset of ImageNet-1k with 500 training images per class. Its 50,000 training and 10,000 validation images are drawn from the ImageNet-1k training split and its 5,000 test images from the validation split, all at their original resolution. We train at 224×224 on one RTX 4090 or RTX 5090 with the full-ImageNet recipe of Appendix I, except for an effective batch size of 64 (32 images per step with two accumulation steps) and a learning rate of 0.001. The smaller batch brings the number of steps close to that of full ImageNet-1k training, and the smaller learning rate matches the reduced data complexity. Every variant has the parameter count of the reported configuration, with the FFN expansion re-solved as described below.

Parameter balancing. Each variant changes the parameter count of the mixer: a per-lag $\mathbf{k}^{(i)}$ or $\Delta^{(i)}$ adds a projection per lag, removing the lags or the convolution removes parameters, and the rotary mixer has no convolution weights. To compare at one budget we hold the total count at the base's $6{,}323{,}100$ by solving for the FFN expansion factor of every block by bisection; since the hidden width is rounded to an integer, the counts agree to within 0.03%. The lag and no-lag DSSD-N twins of Figure C.3 come from an earlier single-seed, 300-epoch run of this parameter-matched family.

G.1 SEEDED ABLATION

Table G.1: Seeded, parameter-matched vision ablation on the 100-class subset: 100 epochs, seeds 0–2, final EMA weights. Test: the 5,000 held-out images; Val.+test: the union with the 10,000 validation images; mean $\pm$ s.d. over seeds. Δ: change of the Val.+test mean relative to the reported configuration. Throughput- and FLOP-matched: the no-lag model with extra blocks in the first two stages. Discussed in §4.1.

Configuration	Seeds	Test top-1 (%)	Val.+test top-1 (%)	Δ
Reported: lags $\pm W$, per-lag $\mathbf{k}$ and Δ	3	73.31 $\pm$0.39	74.62 $\pm$0.07	–
shared Δ across lags	3	73.57 $\pm$0.15	74.70 $\pm$0.19	$+0.09$
shared $\mathbf{k}$ across lags	3	72.94 $\pm$0.46	74.36 $\pm$0.43	-0.26
no lags, parameter-matched	3	72.96 $\pm$0.39	74.10 $\pm$0.07	-0.52
no lags, throughput-matched (blocks 4-4-9-3)	3	73.13 $\pm$0.50	74.15 $\pm$0.55	-0.46
no lags, FLOP-matched (blocks 5-4-9-3)	3	72.64 $\pm$0.50	73.81 $\pm$0.26	-0.80
no convolution (2-D RoPE), lags	3	70.51 $\pm$0.69	71.72 $\pm$0.63	-2.89
no convolution (2-D RoPE), no lags	3	70.24 $\pm$0.28	71.41 $\pm$0.50	-3.20

With the 3×3 convolution in place, the lags are worth 0.5 points at equal parameters, 74.62 ± 0.07 compared with 74.10 ± 0.07 on the 15,000 images, and at equal throughput, where the no-lag model spends the time the lags cost on extra blocks and reaches 74.15 ± 0.55, a gap within its seed spread. Sharing Δ across lags costs nothing ($+0.09$) and sharing $\mathbf{k}$ costs 0.26, within the seed spread of either. Without the convolution both models lose about three points and the lags are worth 0.3.

G.2 WHERE THE TRAINED MODELS USE THE LAGS

To see whether the trained ImageNet-1k models of Table 1 use their lags, and where, we switch the lags off after training by zeroing a lag's key rows in the mixer's projection, which removes what the lag writes into the state while its step size still enters the shared decay. Top-1 is measured on a fixed, seeded subset of 10,000 ImageNet-1k validation images, so every comparison is paired; the step sizes are measured on 512 of them (Figure G.1).

The lags are used in every block. Training has switched no lag off, since in every block the lags' step sizes are 0.5–$1.4\times$ those of the no-lag term, their keys have 0.7–$1.1\times$ its norm, and at most 12% of (head, scan direction) pairs have a lag step size below a tenth of the no-lag term's. Switching every lag off costs 5.7, 3.3 and 6.4 points for DSSD-N, DSSD-T and DSSD-S.

Reliance grows with depth. Switching the lags off in the first block costs at most 0.1 points in every model. DSSD-N and DSSD-T rely most on the last block of each stage, whereas DSSD-S spreads its

reliance over the twelve blocks of stage 2 (Figure G.1a). DSSD-N's last block alone costs 6.6 points, more than switching every lag off. Keeping the lags in only part of the network recovers the least when they are kept in the first blocks. DSSD-N is 6.3 points down with lags in the first half of its blocks but 3.3 down with lags in the last half (b). The first block writes through its lags at 30–50% of the no-lag term's strength (c), and uses them unevenly across scan directions. In two of the four directions of DSSD-T and DSSD-S, the lag that reaches ahead in scan order has 1.2–1.8× the no-lag step size, whereas the lag that reaches behind, whose token the scan's own state already carries, has 0.3–0.5× (d). This is the imbalance the decay predicts, because the lag behind is redundant only while the scan's state still holds its token. Over all blocks, scan directions and models, the decay, taken as the median over heads of $\exp(-A\sum_i \Delta^{(i)}W)$, keeps at least 1% of a token over one row in 15 of 132 cases. These favor the lag ahead by 1.7× on average, whereas the 117 in which the decay erases the row use the two lags within 15% of each other. The step sizes vary little with position in the image.

Reliance and training. These tests measure how much a network trained with lags depends on them, whereas a network trained without lags adapts to their absence (Table G.1). They also suggest that the lags of the first block of each stage could be removed at little cost.

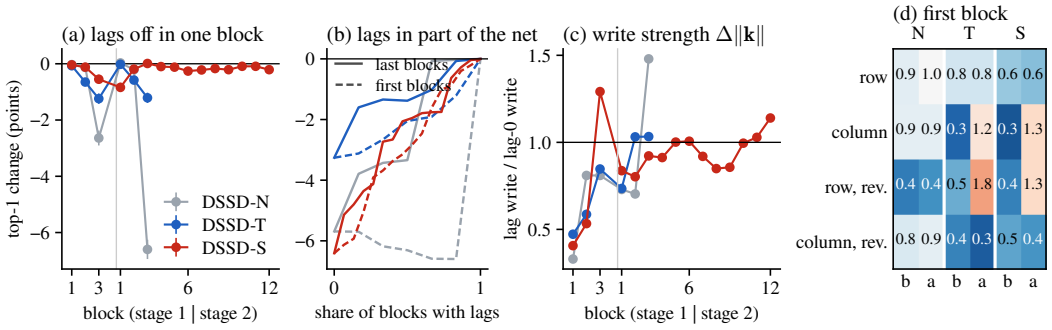


Figure G.1: Where the trained ImageNet-1k models use their lags (EMA weights after the last epoch; 10,000 validation images, paired standard errors in (a)). **(a)** Top-1 change when the lags of one Delay-SSD block are switched off. **(b)** Top-1 change with the lags kept only in the last (solid) or the first (dashed) blocks, against the share of blocks that keep them. **(c)** Write strength of the lags, the mean of $\Delta\|\mathbf{k}\|$ relative to the no-lag term. **(d)** First block: step size of the lag that reaches behind (b) and ahead (a) in scan order, relative to the no-lag term, per scan direction (color on a log scale).

H TIME SERIES

H.1 DATASETS

We use the standard long-term forecasting versions of the datasets as distributed with S-Mamba (Wang et al., 2024b), with their chronological train/validation/test splits (Table H.1); every series is standardized with the mean and variance of its training split, and errors are reported on that scale. **ETTh1, ETTh2, ETTm1, ETTm2** (Zhou et al., 2021) record the oil temperature and six power-load features of two electricity transformers from July 2016 to June 2018, hourly (ETTh) and every 15 minutes (ETTm). **Weather** (Wu et al., 2021) holds 21 meteorological indicators, such as air temperature and humidity, recorded every 10 minutes during 2020 at the weather station of the Max Planck Institute for Biogeochemistry. **Exchange** (Lai et al., 2018) holds the daily exchange rates of eight countries. **Traffic** (Lai et al., 2018; Wu et al., 2021) holds two years of hourly road-occupancy rates from 862 sensors on San Francisco Bay Area freeways, published by the California Department of Transportation. **PEMS03, PEMS04, PEMS07, PEMS08** are traffic-flow counts aggregated into 5-minute windows from the Caltrans Performance Measurement System (Chen et al., 2001), one road network per set, in the versions of Liu et al. (2022a).

Table H.1: Forecasting datasets as loaded: number of variates (series), length in time steps, sampling interval and the chronological split. The 5-minute PEMS sets have a daily period of 288 steps, longer than the lookback of 96 (§4.3).

Dataset	Source	Variates	Steps	Sampling	Train / val. / test
ETTh1, ETTh2	Zhou et al., 2021	7	17,420	1 hour	12 / 4 / 4 months
ETTm1, ETTm2	Zhou et al., 2021	7	69,680	15 min	12 / 4 / 4 months
Weather	Wu et al., 2021	21	52,696	10 min	7 : 1 : 2
Exchange	Lai et al., 2018	8	7,588	1 day	7 : 1 : 2
Traffic	Lai et al., 2018	862	17,544	1 hour	7 : 1 : 2
PEMS03	Liu et al., 2022a	358	26,208	5 min	6 : 2 : 2
PEMS04	Liu et al., 2022a	307	16,992	5 min	6 : 2 : 2
PEMS07	Liu et al., 2022a	883	28,224	5 min	6 : 2 : 2
PEMS08	Liu et al., 2022a	170	17,856	5 min	6 : 2 : 2

H.2 Lag selection

With input-independent parameters, a Delay-SSD head with lags $\mathcal{P}$ predicts from the inputs $\{\mathbf{v}_{t-p} : p \in \mathcal{P}\}$, i.e. from past observations of the series at the lags $\mathcal{P}$, filtered by one shared pole. Seen from the series these are autoregressive terms, which Box–Jenkins identification reads off the partial autocorrelation function (Box et al., 2015). The PACF ranks a lag by the variance it explains given the shorter lags, which a state carrying the recent past already has, whereas the autocorrelation function (ACF) of a near-unit-root series decays monotonically and ranks the shortest lags first (Figure H.1). Proposition 4 (Appendix B) gives the sampling guarantee behind the threshold below. The procedure is: per variate, compute the PACF of the standardized training split by Durbin–Levinson up to lag $K = L$, the lookback, and mark the lags whose coefficient clears the Bonferroni-corrected threshold $z_{1-0.05/(2K)}/\sqrt{n_{\text{obs}}}$ (family-wise level 0.05); keep the lags significant on at least half the variates; rank them by the mean squared coefficient across variates; and take them greedily with a minimum separation equal to the mixer's convolution width, up to $|\mathcal{P}| = 3$. A PACF lag k is read as the lag $p = k$ in equation 4, the token k steps back; $p = 0$ is always included and counts toward $|\mathcal{P}|$, so lags within the convolution width of it are skipped. On a sparse autoregressive (AR) process $x_t = 0.5x_{t-1} + 0.3x_{t-24} + \varepsilon_t$ the procedure selects $\mathcal{P} = \{0, 24\}$ in 18 of 20 seeds from $n_{\text{obs}} = 1000$ samples, and on white noise it selects a spurious lag in 5.0% of seeds, the nominal family-wise level.

Selected sets. On the 5-minute PEMS sets at $L = 96$ the detected period is 12 samples (one hour), and the selected sets are $\{0, 4, 13\}$ on PEMS03 and PEMS07 and $\{0, 4, 14\}$ on PEMS04 and PEMS08; Traffic selects $\{0, 19, 23\}$. The daily period of 288 samples does not fit in a 96-sample window. Exchange, a random walk, selects $\{0\}$, as no lag beyond 1 clears the threshold. The channel-independent ETT and Weather runs, the input-independent variants and the dual-path model (Appendix H.8) use the sets of an earlier reading of PACF lag k as $p = k - 1$: $\{0, 24, 48\}$ on ETTh1 and ETTh2 at both lookbacks ($\{0, 25, 49\}$ under $p = k$); at $L = 96$ and $L = 336$, $\{0, 35, 79\}$ and $\{0, 96, 192\}$ on ETTm1 and $\{0, 4, 44\}$ and $\{0, 4, 98\}$ on ETTm2.

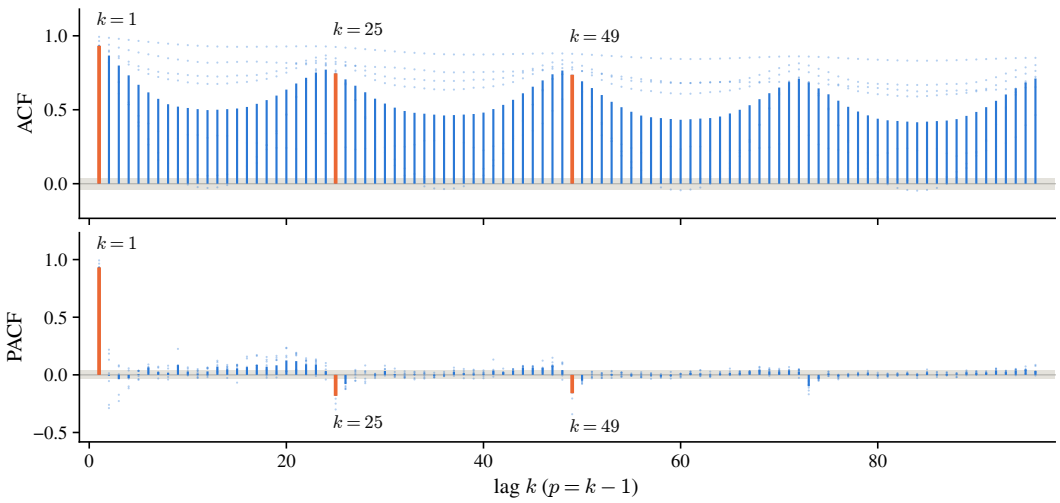


Figure H.1: Lag selection on ETTh1 at $L = 96$: autocorrelation (top, variate mean with per-variate dots) and partial autocorrelation (bottom) with the Bonferroni band; selected lags in orange. The PACF is significant at lags 1, 25 and 49, next to the daily period and its first multiple, while the ACF is above 0.4 everywhere and would rank the shortest lags first. Discussed in §2.4.

H.3 Time-token backbone and protocol

Why time tokens. In the published (*inverted*) S-Mamba a token is one variate's whole input window and the SSM scans across variates, so a lag would shift along the sensor ordering. In the *time-token* model a token is one time step, a linear embedding of all variates at that step and its time-of-day marks, so the SSM scans along time and a lag p reads the series p steps back. The mixer is Mamba, Mamba-2 or Delay-SSD at the same width and depth, giving T-Mamba, T-Mamba-2 and T-DSSD. T-DSSD is reported with the lag set chosen on validation; with $\mathcal{P} = \{0\}$, written T-DSSD $\{0\}$ in the tables, it is the no-lag control, which is Mamba-2 up to kernels, activation and step-size initialization at the same parameter count. Two variants test robustness: a causal model and, for the small ETT sets, a channel-independent model in which each variate is its own sequence.

Model. The time-token backbone of §2.5 is S-Mamba's encoder with time steps as tokens: instance normalization, a linear embedding of each step's V variates plus a linear embedding of its time marks, ℓ layers of [bidirectional mixer, two-convolution FFN, post-norm] identical to the inverted model's, layer normalization, and a linear head $d_{\text{model}} \to V$ per token followed by a linear map from the lookback to the horizon along time. For a given d_{model}, d_{ff}, ℓ and N the encoder parameter count equals the inverted model's to the parameter (checked by a unit test); only the embedding ($L \times d_{\text{model}}$ against $V \times d_{\text{model}}$) and the head differ. Two lags ($|\mathcal{P}| = 3$) add about 5% parameters (the per-lag Δ projection) and 7–14% FLOPs, depending on how the scan is counted (Appendix H.4). The channel-independent form folds the V variates into the batch, with $V\times$ the tokens and $68\times$ S-Mamba's FLOPs on ETTh1 ($105\times$ on Traffic), so it is run on ETT only.

Widths and schedules. Traffic and PEMS use $d_{\text{model}} = d_{\text{ff}} = 512$, four layers, $N = 16$, head dimension 8, batch 16 and five epochs, with early stopping on validation loss (patience 3). The ETT sets use $d_{\text{model}} = d_{\text{ff}} = 256$, two layers and ten epochs, and Weather uses 512, three layers and five epochs. ETT and Weather use lookback 336, so that the daily lag (24 on ETTh, 96 on ETTm, 144 on Weather) lies inside the window, and channel independence, because the channel-mixing form with all variates in one token overfits on these small sets. The iso-cost model keeps every hyper-parameter and reduces $d_{\text{model}} = d_{\text{ff}}$ to 480 (from 512) and 232 (from 256), the largest widths at which T-DSSD with two lags has no more parameters and no more analytic FLOPs than T-Mamba-2. The iso-width runs of Table H.8 use T-Mamba-2 and the control at the same reduced width. The causal backbone (Table H.5) drops the reversed mixer and reads the forecast from the last token through a linear map from d_{model} to all V variates over the horizon. Its references, control and PACF-lag model are swept, and its periodic-lag model runs at the bidirectional control's rate.

Selection protocol. Every setting of Tables H.3, H.4 and H.5 is run at seeds 2023, 2024 and 2025 on PEMS and Traffic and reported as mean ± s.d. The channel-independent ETT settings at lookback 336 carry three seeds for the reference models and T-DSSD and one for the rest; Weather has one, as a channel-independent run at lookback 336 costs about one GPU-hour. The learning rates of S-Mamba, T-Mamba, T-Mamba-2, the no-lag Delay-SSD control and the input-independent models are swept over $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ at seed 2023. The other Delay-SSD variants with lags are not swept, so they get one rate where the control gets three. Each runs at the rate its control chose, since the control and T-Mamba-2 differ in kernels, conv activation and step-size initialization and do not share an optimum. Seeds 2024 and 2025 run at the chosen rate, for three controls also at a second rate, and every table reports a model at the rate with the lowest validation loss averaged over the seeds run at it. The control's reported rate therefore differs from its lagged variants' in 2 of the 20 PEMS/Traffic settings. On PEMS08 at horizon 12 the single seed at 10^{-3} edges the three-seed mean at 10^{-4}, the better rate on seed 2023. On Traffic at horizon 96 the control overfits after its second or third epoch at both 3×10^{-4} and 10^{-3}, and its mean validation losses at the two rates differ by 0.03%. On the channel-independent ETT settings 82% of runs reach their best validation loss in the first epoch, so the rate chosen there matters little. The lag set is also chosen on validation, between the PACF set of Appendix H.2 and a periodic set, and Table H.7 reports both. The periodic set holds multiples of the daily period on hourly data, such as $\{0, 24, 48\}$ on Traffic, and is $\{0, 12, 24, 36\}$ on 5-minute PEMS. Wins and Wilcoxon signed-rank p-values are computed over settings on the per-setting means (one-sided, "better than the reference"), as recommended by Demšar (2006).

Patched channel-independent geometry. The published Mamba forecasters that mix along time (MambaTS (Cai et al., 2026), SST (Xu et al., 2025)) scan PatchTST-style patches. We ran the same geometry on the ETT/Weather settings, with patches of 16 samples, stride 8 and lags in patch units, at $d_{\text{model}} = 256$ and at $d_{\text{model}} = 128$ with dropout 0.2. It is worse than the unpatched channel-independent form for every mixer and every setting, for example 0.547 against 0.422 for T-Mamba-2 on ETTh1 at horizon 96, so the unpatched rows are the ones reported.

H.4 Cost of the time-token models

Encoder FLOPs per token are equal across the mixers, and the token count changes to L from the inverted model's $V + n_{\text{mark}}$, which includes the n_{mark} time-mark features it embeds as extra tokens. A time-token model thus costs $8.4\times$ the inverted one on ETTh1 ($V = 7$), $0.32\times$ on PEMS04 ($V = 307$) and $0.12\times$ on Traffic ($V = 862$). Table H.2 measures the variants at the widths of Table H.3 on one otherwise idle RTX 5090 (batch 16, fp32, CUDA-event medians after a warm-up). Two lags ($|\mathcal{P}| = 3$) add 5% parameters to T-Mamba-2 and 9–14% forward FLOPs with the chunked SSD counted as executed in both models, or 7–8% with the scan counted in its recurrent form, as in Table 1. Under the two counts the iso-cost model is at 0.95–$0.97\times$ and 0.90–$0.95\times$ T-Mamba-2's FLOPs. The measured training step is 0.86–$1.01\times$ T-Mamba-2's, since our kernels run the no-lag control at 0.81–$0.95\times$ and the lags add 0–25% on top of it. Inference throughput is 0.82–$0.88\times$ and peak memory is 1.4–$1.65\times$, of which the lags account for 3–7 points, as the no-lag control already stores 1.33–$1.55\times$ Mamba-2's intermediates. Recomputing the lagged operands in the backward pass brings the memory to 1.3–$1.5\times$ at the same step time. At the iso-cost widths $d_{\text{model}} = 232$ and 480, with no more parameters or FLOPs than T-Mamba-2, the step is 0.92–$0.95\times$ T-Mamba-2's. In streaming inference the chunked scan returns its final state, after which each token is one step. Under a CUDA graph that step takes 0.057 ms for a two-lag mixer at $d_{\text{model}} = 256$, 0.042 ms for the no-lag mixer and 0.013 ms for `mamba_ssm`'s fused Mamba-2 step. The lag ring buffer holds $\max \mathcal{P} + 2$ tokens of $\mathbf{v}$, $\mathbf{k}$ and δ per sequence (1.4 MB at batch 16).

Table H.2: Measured cost of the time-token variants at the widths of Table H.3 ($d_{\text{model}} = 256$, two layers on ETTh1; 512, four layers on Traffic), horizon 96, batch 16, fp32, one RTX 5090. Top: GFLOPs are forward per sample from the analytic count; "step" is one training step (forward and backward), with the ratio to T-Mamba-2 for the Delay-SSD rows; samples/s is training throughput; MB is peak memory; "CI patched": channel-independent with patches of 16 samples, stride 8. Bottom: streaming decode of one mixer layer, a prefill over L and then one token per step, eager and under a CUDA graph, with the size of the recurrent cache.

	ETTh1, $L = 96$					Traffic, $L = 96$				
Model	params	GFLOPs	step ms	samples/s	MB	params	GFLOPs	step ms	samples/s	MB
S-Mamba (inv.)	1.19M	0.03	5.53	2893	85	8.98M	15.79	23.64	677	2031
iTransformer (inv.)	0.84M	0.02	3.15	5082	79	6.41M	17.21	39.71	403	5284
T-Mamba	1.15M	0.23	6.46	2477	127	9.78M	1.92	10.79	1483	556
T-Mamba-2	1.14M	0.23	10.75	1488	126	9.71M	1.93	20.58	778	551
T-DSSD $\{0\}$	1.14M	0.23	10.25 (0.95×)	1562	173	9.71M	1.93	18.73 (0.91×)	854	731
T-DSSD, 2 lags	1.20M	0.26	9.39 (0.87×)	1704	178	10.23M	2.10	17.99 (0.87×)	890	768
gated	1.20M	0.26	10.42	1535	182	10.23M	2.10	19.16	835	789
recompute	1.20M	0.26	10.82	1478	165	10.23M	2.10	22.50	711	713
T-DSSD, 2 lags, iso-cost width	0.99M	0.22	9.90 (0.92×)	1615	167	9.06M	1.87	19.47 (0.95×)	822	732
T-DSSD, 2 lags, input-indep.	1.07M	0.23	10.29 (0.96×)	1555	166	9.32M	1.93	19.13 (0.93×)	836	708
T-DSSD, 3 lags	1.24M	0.28	9.35 (0.87×)	1711	182	–	–	–	–	–
T-Mamba-2, CI patched	1.13M	0.18	10.56	1515	142	–	–	–	–	–
T-DSSD, 2 lags, CI patched	1.19M	0.21	9.74 (0.91×)	1642	199	–	–	–	–	–

	ETTh1, $L = 96$				Traffic, $L = 96$			
	prefill	step ms		cache	prefill	step ms		cache
Mixer layer	ms	eager	graph	MB	ms	eager	graph	MB
Delay-SSD $\{0\}$ mixer	0.74	0.421	0.042	0.3	0.80	0.423	0.047	0.6
Delay-SSD mixer, 2 lags	0.74	0.556	0.057	1.4	0.89	0.564	0.062	1.6
Mamba-2 `step` (mamba_ssm)	–	0.312	0.013	0.3	–	0.283	0.018	0.6

H.5 Full results

The result tables carry test errors only, and Table H.6 collects the paired tests behind every comparison.

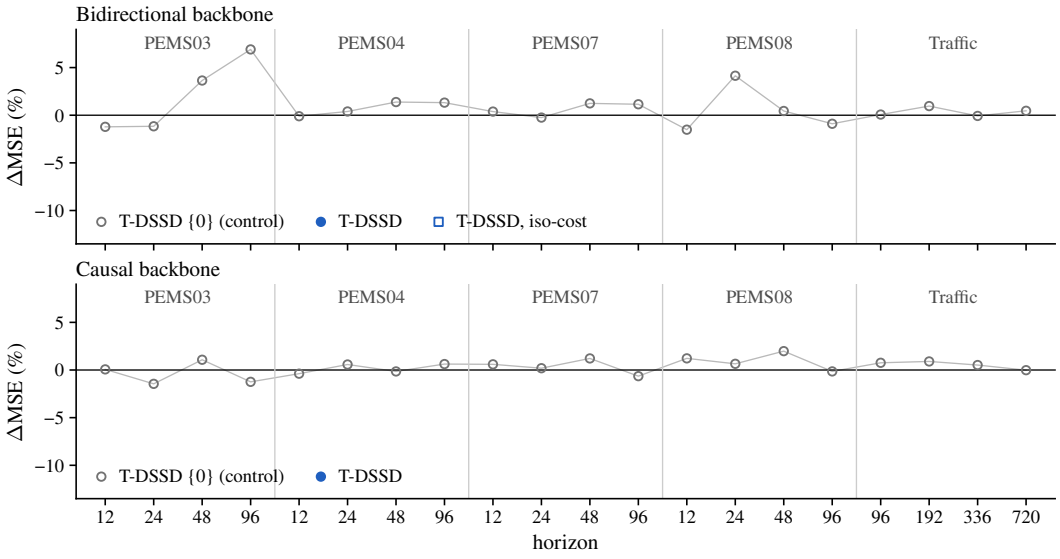


Figure H.2: Relative change in test MSE against T-Mamba-2 per (dataset, horizon) setting for the no-lag control, T-DSSD and T-DSSD at iso-cost width, in the bidirectional backbone of Table H.3 (top) and the causal backbone of Table H.5 (bottom). Paired statistics: Table H.6. Discussed in §4.3 and Appendix H.5.

Table H.3: Test MSE / MAE on PEMS and Traffic, lookback 96, bidirectional backbone (mean over three seeds; s.d. in Table H.7). Only the mixer differs between columns (§4.3); "iso-cost" is T-DSSD at no more parameters or FLOPs than T-Mamba-2. Bold: best in the row; underline: second. Paired statistics: Table H.6(a).

		T-Mamba		T-Mamba-2		T-DSSD $\{0\}$		T-DSSD		T-DSSD, iso-cost	
Dataset	Horizon	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PEMS03	12	0.087	0.195	0.088	0.196	0.087	0.196	**0.084**	**0.190**	0.084	0.190
	24	0.111	0.218	0.110	0.219	0.108	0.217	**0.107**	**0.213**	0.108	0.213
	48	0.161	0.261	**0.155**	0.258	0.161	0.263	0.162	0.258	0.158	**0.256**
	96	0.244	0.322	0.242	0.323	0.259	0.330	**0.236**	**0.314**	0.239	0.322
	Avg	0.151	0.249	0.149	0.249	0.154	0.252	**0.147**	**0.244**	0.147	0.245
PEMS04	12	0.085	0.191	0.086	0.192	0.085	0.192	0.085	0.190	**0.085**	**0.190**
	24	0.100	0.209	0.098	0.207	0.099	0.208	**0.098**	0.206	0.098	**0.206**
	48	0.126	0.236	0.124	0.235	0.125	0.236	**0.121**	**0.230**	0.123	0.235
	96	0.166	0.269	0.168	0.274	0.171	0.276	**0.164**	**0.269**	0.165	0.270
	Avg	0.119	0.226	0.119	0.227	0.120	0.228	**0.117**	**0.224**	0.118	0.225
PEMS07	12	0.083	0.181	0.083	0.182	0.083	0.182	**0.082**	**0.179**	0.083	0.180
	24	0.102	0.201	0.102	0.204	0.102	0.200	0.101	0.200	**0.100**	**0.198**
	48	0.136	0.232	0.137	0.236	0.139	0.237	0.135	**0.230**	**0.133**	0.230
	96	0.185	**0.270**	0.185	0.277	0.188	0.279	**0.181**	0.272	0.182	0.272
	Avg	0.127	0.221	0.127	0.225	0.128	0.225	0.125	0.220	**0.124**	**0.220**
PEMS08	12	0.114	0.213	0.116	0.217	0.115	0.216	**0.113**	**0.212**	0.115	0.214
	24	0.148	0.244	0.145	0.244	0.151	0.249	0.143	**0.239**	**0.143**	0.239
	48	0.205	0.285	0.206	0.286	0.207	0.290	**0.203**	**0.283**	0.205	0.286
	96	0.335	0.354	0.330	0.356	**0.328**	**0.346**	0.338	0.353	0.343	0.357
	Avg	0.200	0.274	0.199	0.276	0.200	0.275	**0.199**	**0.272**	0.202	0.274
Traffic	96	0.623	0.354	0.621	0.357	0.621	0.356	**0.598**	**0.336**	0.602	0.337
	192	0.645	0.366	0.639	0.362	0.645	0.359	0.634	**0.354**	**0.631**	0.356
	336	0.655	0.370	0.649	0.368	0.649	0.366	0.643	0.364	**0.634**	**0.362**
	720	0.691	0.390	0.693	0.389	0.697	0.391	**0.681**	0.383	0.689	**0.381**
	Avg	0.653	0.370	0.651	0.369	0.653	0.368	0.639	0.359	**0.639**	**0.359**

Table H.4: Test MSE / MAE on the channel-independent ETT and Weather settings at lookback 336, where the daily lag lies inside the window (columns and protocol as in Table H.3). Three seeds for the reference models and T-DSSD, one for the rest and for Weather. Most runs reach their best validation loss in the first epoch here (82% on ETT), and the seed spread is 2–$6\times$ that of the PEMS settings (§4.3). Bold: best in the row; underline: second. Paired statistics: Table H.6(b).

		T-Mamba		T-Mamba-2		T-DSSD $\{0\}$		T-DSSD		T-DSSD, iso-cost	
Dataset	Horizon	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.423**	**0.438**	0.445	0.447	0.465	0.462	0.449	0.452	0.478	0.460
	336	0.519	0.490	0.524	0.502	**0.503**	**0.490**	0.561	0.519	0.604	0.538
	Avg	**0.471**	**0.464**	0.485	0.474	0.484	0.476	0.505	0.486	0.541	0.499
ETTh2	96	**0.311**	**0.357**	0.331	0.384	0.329	0.378	0.327	0.377	0.339	0.378
	336	**0.398**	**0.418**	0.414	0.433	0.437	0.443	0.441	0.455	0.437	0.448
	Avg	**0.354**	**0.388**	0.373	0.408	0.383	0.410	0.384	0.416	0.388	0.413
ETTm1	96	0.430	0.415	**0.377**	0.399	0.378	0.398	0.441	0.410	0.397	**0.392**
	336	0.557	0.486	0.514	0.482	0.574	0.500	0.536	0.478	**0.494**	**0.468**
	Avg	0.493	0.451	0.446	0.440	0.476	0.449	0.489	0.444	**0.445**	**0.430**
ETTm2	96	**0.176**	0.269	0.184	0.272	0.177	**0.267**	0.186	0.275	0.190	0.276
	336	**0.296**	**0.345**	0.306	0.349	0.312	0.353	0.325	0.358	0.319	0.359
	Avg	**0.236**	**0.307**	0.245	0.311	0.244	0.310	0.256	0.317	0.254	0.317
Weather	96	0.163	0.217	0.162	0.212	0.156	**0.205**	0.169	0.221	**0.156**	0.206

Input-independent variants. With selection disabled the mixer's step size, $\mathbf{k}$ and $\mathbf{q}$ are learned constants (per head and, for Δ, per lag) instead of projections of the input; lags, decay, skip and gated norm are unchanged and the same kernels run. On PEMS/Traffic the input-independent control is $+4.7\%$ against T-Mamba-2 and the input-independent T-DSSD is $+4.2\%$ (Table H.8), so selection matters for this parameterization, and the lags recover part of what it loses.

Table H.5: Test MSE / MAE in the causal backbone: the settings and protocol of Tables H.3 and H.4 with the reversed mixer removed and the forecast read from the last token, i.e. a plain multi-layer causal sequence model; T-DSSD uses the PACF or periodic set chosen on validation. Three seeds per setting (Weather omitted). Bold: best in the row; underline: second. Paired statistics: Table H.6(c). Discussed in §4.3.

		T-Mamba		T-Mamba-2		T-DSSD $\{0\}$		T-DSSD	
Dataset	Horizon	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PEMS03	12	0.079	0.184	0.078	0.184	0.078	0.185	**0.078**	**0.183**
	24	0.101	0.209	0.103	0.210	0.101	0.209	**0.100**	**0.207**
	48	**0.159**	0.256	0.162	**0.255**	0.164	0.257	0.164	0.256
	96	0.260	0.323	0.270	0.328	0.267	0.330	**0.256**	**0.320**
	Avg	0.150	0.243	0.153	0.244	0.152	0.245	**0.150**	**0.242**
PEMS04	12	0.083	0.189	0.083	0.189	0.083	0.189	**0.083**	**0.189**
	24	0.096	0.205	0.096	0.205	0.096	0.206	**0.095**	**0.204**
	48	0.121	0.233	0.121	0.233	0.121	0.233	**0.120**	**0.232**
	96	0.165	0.274	0.163	0.271	0.164	0.273	**0.161**	**0.268**
	Avg	0.116	0.225	0.116	0.225	0.116	0.225	**0.115**	**0.223**
PEMS07	12	0.081	0.179	0.080	0.179	0.081	0.179	**0.080**	**0.177**
	24	0.099	0.196	0.099	0.197	0.099	0.196	**0.098**	**0.195**
	48	0.132	0.228	0.131	0.228	0.133	0.229	**0.130**	**0.226**
	96	0.179	0.268	0.180	0.268	0.178	0.268	**0.177**	**0.265**
	Avg	0.123	0.218	0.122	0.218	0.123	0.218	**0.121**	**0.216**
PEMS08	12	0.105	0.205	0.105	0.204	0.106	0.205	**0.104**	**0.202**
	24	0.136	0.232	0.134	0.229	0.135	0.231	**0.134**	**0.228**
	48	0.199	0.277	0.190	0.270	0.193	0.273	**0.189**	**0.269**
	96	0.334	0.346	0.338	0.347	0.338	0.350	**0.331**	**0.340**
	Avg	0.194	0.265	0.192	0.262	0.193	0.265	**0.189**	**0.260**
Traffic	96	0.596	0.328	0.593	0.325	0.597	0.328	**0.592**	**0.324**
	192	0.623	0.340	0.621	0.338	0.627	0.341	**0.618**	**0.335**
	336	0.636	0.346	0.636	0.344	0.639	0.344	**0.632**	**0.340**
	720	0.667	0.357	0.671	0.359	0.671	0.359	**0.662**	**0.353**
	Avg	0.630	0.343	0.630	0.342	0.634	0.343	**0.626**	**0.338**
ETTh1	96	0.465	0.470	0.492	0.479	0.479	0.470	**0.427**	**0.445**
	336	**0.541**	**0.508**	0.575	0.526	0.567	0.523	0.549	0.520
	Avg	0.503	0.489	0.534	0.502	0.523	0.497	**0.488**	**0.482**
ETTh2	96	0.312	0.363	**0.307**	**0.358**	0.334	0.378	0.312	0.362
	336	0.402	0.430	**0.397**	**0.425**	0.397	0.428	0.410	0.437
	Avg	0.357	0.396	**0.352**	**0.392**	0.366	0.403	0.361	0.399
ETTm1	96	0.436	0.427	0.425	0.430	0.412	0.425	**0.353**	**0.393**
	336	0.446	0.454	0.440	0.456	0.472	0.462	**0.410**	**0.435**
	Avg	0.441	0.441	0.433	0.443	0.442	0.444	**0.382**	**0.414**
ETTm2	96	**0.188**	0.278	0.188	0.278	0.190	**0.277**	0.190	0.277
	336	**0.317**	**0.361**	0.340	0.372	0.336	0.369	0.325	0.361
	Avg	**0.253**	0.320	0.264	0.325	0.263	0.323	0.257	**0.319**

Table H.6: Paired statistics for the three forecasting tables, over the settings of each (per-setting means over seeds): the mean relative change in test MSE against the reference, the number of settings on which the model is better, and the one-sided Wilcoxon signed-rank p for "better than the reference" (Demšar, 2006). Bold: $p < 0.05$; "–": the reference itself. The iso-cost model is compared in the bidirectional and channel-independent backbones. Discussed in §4.3.

Against	Statistic	T-Mamba	T-Mamba-2	T-DSSD $\{0\}$	T-DSSD	T-DSSD, iso-cost
(a) Bidirectional, PEMS and Traffic, lookback 96 (Table H.3), 20 settings						
T-Mamba-2	Mean ΔMSE	$+0.4\%$	–	$+0.9\%$	-1.4%	-1.1%
	Wins	9/20	–	7/20	18/20	18/20
	Wilcoxon p	0.923	–	0.976	**0.004**	**0.002**
No-lag control	Mean ΔMSE	-0.4%	-0.8%	–	-2.2%	-1.9%
	Wins	9/20	13/20	–	18/20	19/20
	Wilcoxon p	0.298	**0.027**	–	**<0.001**	**<0.001**
(b) Channel-independent, ETT and Weather, lookback 336 (Table H.4), 9 settings						
T-Mamba-2	Mean ΔMSE	-0.1%	–	$+1.3\%$	$+5.2\%$	$+4.0\%$
	Wins	6/9	–	4/9	1/9	2/9
	Wilcoxon p	0.326	–	0.752	0.994	0.973
No-lag control	Mean ΔMSE	-1.2%	-1.0%	–	$+4.1\%$	$+3.0\%$
	Wins	6/9	5/9	–	3/9	2/9
	Wilcoxon p	0.213	0.285	–	0.850	0.951

Against	Statistic	T-Mamba	T-Mamba-2	T-DSSD $\{0\}$	T-DSSD
(c) Causal, every setting of (a) and (b) but Weather (Table H.5), 28 settings					
T-Mamba-2	Mean ΔMSE	-0.3%	–	$+0.5\%$	-2.1%
	Wins	12/28	–	11/28	24/28
	Wilcoxon p	0.619	–	0.836	**0.001**
No-lag control	Mean ΔMSE	-0.8%	-0.5%	–	-2.6%
	Wins	17/28	17/28	–	25/28
	Wilcoxon p	**0.021**	0.169	–	**<0.001**

Table H.7: Test MSE, mean ± s.d. over seeds (no s.d.: one seed), of the reference mixers and both lag sets on every setting of Tables H.3 and H.4 but Weather, plus the published inverted S-Mamba at the same width and depth (one seed; not ranked). ETT rows are channel-independent at lookback 336. "PACF" and "periodic" are the two candidate lag sets of Appendix H.3; T-DSSD in the other tables uses the one chosen on validation. Bold: best time-token model in the row; underline: second.

Dataset	Horizon	S-Mamba (inverted)	T-Mamba	T-Mamba-2	T-DSSD $\{0\}$	T-DSSD (PACF)	T-DSSD (periodic)
PEMS03	12	0.065 ±0.000	0.087 ±0.002	0.088 ±0.001	0.087 ±0.003	0.085 ±0.001	**0.084 ±0.001**
	24	0.090	0.111 ±0.002	0.110 ±0.002	0.108 ±0.001	0.109 ±0.001	**0.107 ±0.001**
	48	0.136 ±0.002	0.161 ±0.008	**0.155 ±0.007**	0.161 ±0.004	0.162 ±0.005	0.162 ±0.010
	96	0.211	0.244 ±0.006	0.242 ±0.014	0.259 ±0.018	0.244 ±0.003	**0.236 ±0.020**
PEMS04	12	0.075	0.085 ±0.001	0.086 ±0.001	0.085 ±0.001	**0.085 ±0.001**	0.085 ±0.000
	24	0.094	0.100 ±0.001	0.098 ±0.000	0.099 ±0.000	0.099 ±0.001	**0.098 ±0.001**
	48	0.130	0.126 ±0.001	0.124 ±0.002	0.125 ±0.001	0.123 ±0.001	**0.121 ±0.001**
	96	0.193	0.166 ±0.001	0.168 ±0.001	0.171 ±0.003	0.166 ±0.003	**0.164 ±0.001**
PEMS07	12	0.060	0.083 ±0.001	0.083 ±0.001	0.083 ±0.000	0.083 ±0.001	**0.082 ±0.001**
	24	0.078	0.102 ±0.001	0.102 ±0.001	0.102 ±0.001	0.102 ±0.001	**0.101 ±0.000**
	48	0.108	0.136 ±0.000	0.137 ±0.001	0.139 ±0.000	**0.135 ±0.002**	0.135 ±0.002
	96	0.187	0.185 ±0.001	0.185 ±0.001	0.188 ±0.002	0.184 ±0.001	**0.181 ±0.000**
PEMS08	12	0.075	0.114 ±0.001	0.116 ±0.001	0.115	0.114 ±0.002	**0.113 ±0.001**
	24	0.099	0.148 ±0.002	0.145 ±0.002	0.151 ±0.003	0.147 ±0.001	**0.143 ±0.002**
	48	0.162 ±0.004	0.205 ±0.004	0.206 ±0.004	0.207 ±0.008	0.206 ±0.007	**0.203 ±0.004**
	96	0.241	0.335 ±0.014	0.330 ±0.017	**0.328 ±0.010**	0.338 ±0.014	0.338 ±0.006
Traffic	96	0.385	0.623 ±0.001	0.621 ±0.005	0.621 ±0.005	0.618 ±0.012	**0.598 ±0.007**
	192	0.416	0.645 ±0.011	0.639 ±0.008	0.645 ±0.005	**0.634 ±0.007**	0.635 ±0.006
	336	0.421	0.655 ±0.014	0.649 ±0.007	0.649 ±0.011	0.645 ±0.008	**0.643 ±0.008**
	720	0.465	0.691 ±0.007	0.693 ±0.003	0.697 ±0.017	0.688 ±0.006	**0.681 ±0.012**
ETTh1	96	0.397	**0.423**	0.445 ±0.025	0.465 ±0.057	0.466 ±0.041	0.449 ±0.026
	336	0.456	0.519 ±0.011	0.524 ±0.035	**0.503**	0.575 ±0.010	0.561 ±0.057
ETTh2	96	0.305	**0.311**	0.331	0.329	0.327 ±0.007	0.333 ±0.008
	336	0.415	**0.398**	0.414 ±0.016	0.437 ±0.020	0.443 ±0.017	0.441 ±0.024
ETTm1	96	0.309	0.430 ±0.040	**0.377 ±0.023**	0.378 ±0.027	0.417 ±0.064	0.441 ±0.073
	336	0.392	0.557 ±0.084	**0.514**	0.574 ±0.036	0.536 ±0.005	0.555 ±0.068
ETTm2	96	0.189	**0.176**	0.184 ±0.012	0.177 ±0.009	0.184 ±0.008	0.186 ±0.010
	336	0.295	**0.296 ±0.007**	0.306 ±0.003	0.312 ±0.010	0.325 ±0.018	0.324 ±0.013

Table H.8: Test MSE, mean ± s.d. over seeds (no s.d.: one seed), of the ablation variants next to T-Mamba-2, on the PEMS and Traffic settings of Table H.3. "input-indep.": input-independent Δ, $\mathbf{k}$ and $\mathbf{q}$, learned as constants; "iso width": T-Mamba-2 and the no-lag control at the reduced width $d_{\text{model}} = d_{\text{ff}} = 480$; "iso-cost": T-DSSD at that width. Bold: best in the row; underline: second. Discussed in §4.3 and Appendix H.3.

Dataset	Horizon	T-Mamba-2	Input-indep. $\{0\}$	Input-indep., lags	T-Mamba-2, iso width	$\{0\}$, iso width	T-DSSD, iso-cost
PEMS03	12	0.088 ±0.001	0.093 ±0.001	0.093 ±0.001	0.089 ±0.001	0.088 ±0.001	**0.084 ±0.001**
	24	0.110 ±0.002	0.122 ±0.003	0.120 ±0.004	0.110 ±0.001	0.110 ±0.001	**0.108 ±0.003**
	48	**0.155 ±0.007**	0.165 ±0.009	0.161 ±0.005	0.165 ±0.006	0.163 ±0.005	0.158 ±0.002
	96	0.242 ±0.014	0.248 ±0.005	0.244 ±0.007	**0.234 ±0.005**	0.240 ±0.005	0.239 ±0.016
PEMS04	12	0.086 ±0.001	0.085	0.085	0.087 ±0.001	0.086 ±0.001	**0.085 ±0.000**
	24	0.098 ±0.000	0.101 ±0.001	0.100 ±0.001	0.099 ±0.001	0.100 ±0.002	**0.098 ±0.001**
	48	0.124 ±0.002	0.128	0.127	0.124 ±0.002	0.125 ±0.002	**0.123 ±0.001**
	96	0.168 ±0.001	0.182 ±0.004	0.179 ±0.006	0.168 ±0.004	0.169 ±0.001	**0.165 ±0.001**
PEMS07	12	0.083 ±0.001	0.084 ±0.001	0.084 ±0.001	0.083 ±0.000	0.084 ±0.000	**0.083 ±0.001**
	24	0.102 ±0.001	0.102 ±0.001	0.103 ±0.000	0.102 ±0.001	0.102 ±0.001	**0.100 ±0.001**
	48	0.137 ±0.001	0.138	0.139	0.136 ±0.001	0.137 ±0.003	**0.133 ±0.002**
	96	0.185 ±0.001	0.192 ±0.002	0.189 ±0.002	0.187 ±0.001	0.188 ±0.002	**0.182 ±0.001**
PEMS08	12	0.116 ±0.001	0.117 ±0.002	0.118	0.117 ±0.002	0.118 ±0.001	**0.115 ±0.002**
	24	0.145 ±0.002	0.157 ±0.005	0.157 ±0.005	0.150 ±0.002	0.154 ±0.001	**0.143 ±0.005**
	48	0.206 ±0.004	0.251	0.253	0.208 ±0.003	0.208 ±0.003	**0.205 ±0.001**
	96	0.330 ±0.017	0.340	0.340	0.339 ±0.014	**0.330 ±0.006**	0.343 ±0.012
Traffic	96	0.621 ±0.005	0.643 ±0.012	0.652	0.626 ±0.007	0.628 ±0.016	**0.602 ±0.008**
	192	0.639 ±0.008	0.664 ±0.005	0.651 ±0.010	0.639 ±0.008	0.637 ±0.010	**0.631 ±0.010**
	336	0.649 ±0.007	0.682 ±0.013	0.669 ±0.018	0.651 ±0.002	0.667 ±0.010	**0.634 ±0.004**
	720	0.693 ±0.003	0.710 ±0.016	0.699 ±0.012	0.682 ±0.009	**0.682 ±0.006**	0.689 ±0.006

H.6 Evidence that the lag acts in time

Two measurements test that the lag is used as a lag: the gradient of the trained forecast with respect to past inputs (Figure H.3), and synthetic series whose seasonal structure is known.

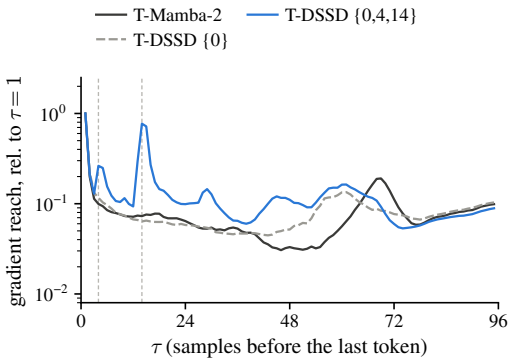


Figure H.3: Gradient reach of the trained forecasters, $\|\partial(\mathbf{u} \cdot \mathbf{h}_L)/\partial x_{L-\tau}\|$ against τ (geometric mean over 8 random probes $\mathbf{u}$ and 32 test windows, normalized by $\tau = 1$, read at the encoder's last token). PEMS04 at horizon 48 ($L = 96$, channel-mixing, the models of Table H.3). T-Mamba-2 and the no-lag control show no peak at the lags (0.07 and 0.06 at $\tau = 14$), while Delay-SSD with lags $\{0, 4, 14\}$ peaks at 0.26 and 0.77 at exactly $\tau = 4$ and 14.

Synthetic seasonal processes. We run three hourly synthetic sets with seven variates, the target variate last as in ETT, through the same pipeline at lookback 336: a persistent seasonal autoregression $x_t = 0.05x_{t-1} + 0.9x_{t-24} + \varepsilon_t$, a deterministic daily-plus-weekly sinusoid with AR(1) noise at unit signal-to-noise ratio, and the AR(1) null $x_t = 0.8x_{t-1} + \varepsilon_t$. Sparse AR processes such as that

of Appendix H.2, with coefficients well inside the unit circle, carry no signal 96 steps ahead, and every model scores the process variance on them. On the seasonal autoregression the daily lag set $\{0, 24, 168\}$ is at -11.5% and -9.8% against T-Mamba-2 at horizons 24 and 96, and -18% and -3% against the control, while a set one step short of the period, $\{0, 17, 23\}$, gains nothing. The lag has to land on the period, which is why a PACF lag k is read as $p = k$ (Appendix H.2). On the sinusoid every model is at the noise floor (0.53–0.54 at every horizon) and on the null every T-DSSD lag set is within 1% of the control, as they should be.

H.7 Lags versus seasonal preprocessing

A classical alternative to lags in the mixer is to remove the seasonal dependence before the network sees the series, by seasonal differencing, by an ARIMA model whose residuals the network then forecasts (Zhang, 2003), or by a trend–seasonal decomposition (Wu et al., 2021; Zeng et al., 2023), and to let an unmodified Mamba-2 forecast the rest. Three differences separate this from a Delay-SSD lag.

Where the lag acts. A prefilter is applied once, to the raw series, with fixed linear coefficients. A Delay-SSD lag acts in every layer, on learned features, and its write strength $\Delta_t^{(i)}$ and key $\mathbf{k}_t^{(i)}$ depend on the input, so the model can use the lagged value only when the content calls for it. Removing that input dependence from the whole model (the input-independent variants of Table H.8) moves T-DSSD from -1.4% to $+4.2\%$ against T-Mamba-2.

What the window contains. At lookback 96 the daily period of the 5-minute PEMS sets (288 samples) is outside the window, so a seasonal ARIMA fitted on the full history would use data the forecaster never sees, while the lags we compare are hourly and inside the window.

Vision has no prefilter. The row stride is a property of flattening a 2-D grid, and the dependence it serves is between learned features of vertically adjacent patches, not an autocorrelation of one series.

H.8 Longer lookback: lags in a dual-path model

At lookback 96 the daily period of the 5-minute PEMS sets (288 samples) lies outside the window, so the lags of Table H.3 are hourly. To test the lag where the daily period fits, we run every model at lookback 336 and add a time-token branch to the strongest model on these sets, the inverted S-Mamba2. The dual model sums the forecasts of two branches in the instance-normalized space: the inverted S-Mamba2 over variate tokens ($d_{\text{model}} = d_{\text{ff}} = 512$, four layers), and a channel-mixing time-token branch (a linear embedding of each step's variates to $d_{\text{model}} = 256$, two layers of the bidirectional mixer and FFN block). The time-token mixer is Mamba-2, the no-lag control, or Delay-SSD, with the periodic set $\{0, 288\}$ or the PACF set at lookback 336 (of the earlier $p = k - 1$ reading), chosen on validation. The learning rate is chosen from $\{3 \times 10^{-4}, 10^{-3}\}$ on the seed-2023 validation loss, and seeds 2024 and 2025 are run at that rate; the inverted references are selected the same way from three rates. Everything else follows Appendix H.3.

In all 16 PEMS settings of Table H.9, Delay-SSD lags in the time-token branch beat the same branch with Mamba-2 (-2.45% mean, -2.37% median, one-sided Wilcoxon $p = 1.5 \times 10^{-5}$). Against the inverted references at the same lookback, the best dual variant chosen on validation is ahead on PEMS04, by about 9% at horizons 48 and 96, and behind on PEMS03 and PEMS08. The inverted models at lookback 336 are already below the published lookback-96 numbers of Mamba forecasters, so that gap is a lookback effect and not a result of the lag. Two further variants gave no lag effect: a channel-independent patched time-token branch with patches of 16 and stride 8, at $+0.6\%$ over seven settings at one seed and 10–$20\times$ the cost per run, and the channel-mixing branch on Traffic, where it overfits ($+25$–45% against S-Mamba2).

Table H.9: Lags in a dual-path model at lookback 336 on PEMS: test MSE, mean ± s.d. over seeds 2023–2025 at the learning rate chosen on validation (entries without s.d. are one seed). "Dual" is the inverted S-Mamba2 plus a channel-mixing time-token branch whose mixer is Mamba-2 or Delay-SSD with lags; the last column compares the two time-token mixers. Bold: best in the row. Discussed in §4.3 and Appendix H.8.

Dataset	Horizon	S-Mamba	S-Mamba2	Dual, Mamba-2	Dual, Delay-SSD lags	Lags vs. Mamba-2
PEMS03	12	0.0619 ±0.0003	**0.0590 ±0.0003**	0.0629 ±0.0018	0.0616 ±0.0008	−2.1%
	24	0.0781 ±0.0051	**0.0723 ±0.0004**	0.0761 ±0.0008	0.0749 ±0.0019	−1.6%
	48	0.0933 ±0.0011	**0.0918 ±0.0006**	0.1000 ±0.0052	0.0956 ±0.0016	−4.4%
	96	**0.1160 ±0.0021**	0.1188 ±0.0009	0.1214 ±0.0005	0.1174 ±0.0034	−3.3%
PEMS04	12	0.0700 ±0.0007	0.0709 ±0.0002	0.0692 ±0.0004	**0.0688 ±0.0004**	−0.6%
	24	0.0789 ±0.0002	0.0815 ±0.0002	0.0806 ±0.0019	**0.0784 ±0.0015**	−2.7%
	48	0.0987 ±0.0074	0.1003 ±0.0020	0.0916 ±0.0025	**0.0895 ±0.0004**	−2.3%
	96	0.1124 ±0.0005	0.1223 ±0.0033	0.1025 ±0.0021	**0.1024 ±0.0042**	−0.1%
PEMS07	12	0.0502 ±0.0002	0.0505 ±0.0001	0.0503 ±0.0004	**0.0498 ±0.0001**	−1.0%
	24	**0.0577 ±0.0004**	0.0594 ±0.0001	0.0594 ±0.0008	0.0579 ±0.0010	−2.5%
	48	**0.0672 ±0.0008**	0.0694 ±0.0003	0.0707 ±0.0009	0.0685 ±0.0003	−3.1%
	96	**0.0766 ±0.0010**	0.0787 ±0.0003	0.0820 ±0.0009	0.0810 ±0.0009	−1.2%
PEMS08	12	0.0681 ±0.0002	**0.0671 ±0.0003**	0.0784 ±0.0018	0.0782 ±0.0051	−0.3%
	24	**0.0828 ±0.0002**	0.0848 ±0.0015	0.1017 ±0.0010	0.0959 ±0.0031	−5.7%
	48	0.1092 ±0.0017	**0.1080 ±0.0024**	0.1391 ±0.0047	0.1317 ±0.0068	−5.3%
	96	0.1573 ±0.0122	**0.1565 ±0.0025**	0.1875 ±0.0084	0.1814 ±0.0079	−3.3%

I Architectural and training details

I.1 Runs, seeds and compute

Every ImageNet-1k, COCO and ADE20K number is from a single training run. The ImageNet runs use seed 0; the COCO and ADE20K runs use the random seed MMEngine draws for each run and records in its log. The routing tasks, the one-dimensional delay task and the vision ablations use seeds 0–2, and the forecasting runs use seeds 2023–2025 (Appendices D, E, G and H.3). As recorded in each run's training log, the ImageNet runs used four NVIDIA RTX 5090 GPUs (32 GB), DSSD-N used four RTX 4090 GPUs (24 GB) on COCO and two on ADE20K, DSSD-S two RTX 4090s on ADE20K, and DSSD-T four RTX 5090s on both. The vision ablations, the routing and delay tasks and the forecasting runs used one GPU each, an RTX 5090 or an RTX 4090. Timing measurements use one RTX 5090 power-limited to 450 W unless a table states otherwise. The code supplement, a single-file implementation of the models and kernels with a dependency-only installer, pins the software used for the paper: Python 3.11, PyTorch 2.13 with CUDA 13.0, TileLang 0.1.12 and timm 1.0.28; detection and segmentation used MMDetection 3.3.0 and MMSegmentation 1.2.2.

I.2 Vision configurations

The three sizes share one design and differ in depth and width (Table I.1). Stages 1–2 use Delay-SSD blocks and stages 3–4 global attention (§2.5); every Delay-SSD block has state size $N = 32$, head dimension $C = 16$, lags $\pm W$ and $\pm H$, and a per-lag $\mathbf{k}$ and Δ.

Table I.1: Depth and width of the vision models; parameters and FLOPs at 224^2 as in Table 1.

Model	Blocks per stage	Width per stage	Params	FLOPs (G)
DSSD-N	3, 3, 9, 3	32, 64, 160, 256	7M	1.6
DSSD-T	3, 3, 9, 3	64, 128, 320, 512	27M	5.0
DSSD-S	3, 12, 14, 3	96, 192, 384, 576	50M	14.1

I.3 IMAGENET-1K CLASSIFICATION

We train for 300 epochs on 224×224 inputs in bfloat16 with AdamW, a global batch size of 4096 (4 GPUs $\times$ 64 images $\times$ 16 accumulation steps) and a base learning rate of 4×10^{-3} with a 5% linear warm-up and cosine annealing, and evaluate an exponential moving average of the weights with decay 0.999. Regularization grows with model size: stochastic depth is 0, 0.175 and 0.4 and classifier dropout 0, 0.15 and 0.4 for DSSD-N, T and S, and weight decay 0.025 for DSSD-N and 0.05 for the others. Gradients are clipped at norm 5.0 except for DSSD-T. Each training image is augmented in four steps, with no separate color jitter.

1. *Crop and flip.* A random crop covering 8–100% of the image area with aspect ratio between $3/4$ and $4/3$ is resized to 224×224 with bicubic interpolation and flipped horizontally with probability 0.5.
2. *RandAugment* (Cubuk et al., 2020). Two operations are drawn uniformly, with replacement, from fifteen: auto-contrast, histogram equalization, color inversion, rotation, posterization, solarization, solarize-add, saturation, contrast, brightness and sharpness adjustment, and horizontal and vertical shear and translation. Each is applied with probability 0.5, at a strength m drawn for every application from a normal distribution with mean 9 and standard deviation 0.5, clipped to $[0, 10]$. The strength scales each operation linearly: rotation by up to $\pm 3m$ degrees; shear by up to $\pm 0.03m$; translation by up to $\pm 4.5m\%$ of the image side; saturation, contrast, brightness and sharpness factors between $1 - 0.09m$ and $1 + 0.09m$; posterization to $4 - \lfloor 0.4m \rfloor$ bits per channel; solarization of the pixels above $256 - \lfloor 25.6m \rfloor$; and solarize-add of $\lfloor 11m \rfloor$ to the pixels below 128. Signed parameters take a random sign, and pixels a geometric operation uncovers are filled with the mean color $(124, 116, 104)$. At the mean strength this is up to $\pm 27^\circ$ of rotation and $\pm 40.5\%$ of translation.
3. *Random erasing* (Zhong et al., 2020). After normalization, with probability 0.25, one rectangle covering 2–33% of the image with aspect ratio between 0.3 and 3.3 is replaced by standard-normal noise drawn per pixel.
4. *Mixing.* Every batch is mixed by Mixup (Zhang et al., 2018) with a mixing weight drawn from $\mathrm{Beta}(0.8, 0.8)$ or by CutMix (Yun et al., 2019) with a pasted-box area drawn from $\mathrm{Beta}(1, 1)$, each chosen with probability 0.5, and the mixed targets use label smoothing 0.1 (Szegedy et al., 2016).

The pipeline is built with `timm`'s `create_transform` and `Mixup`; RandAugment corresponds to its configuration string `rand-m9-mstd0.5-inc1`.

I.4 MS-COCO DETECTION AND INSTANCE SEGMENTATION

Backbones are initialized from the last-epoch ImageNet-1k weights and fine-tuned in MMDetection (Chen et al., 2019) with a Mask R-CNN head (He et al., 2017) and an FPN with group normalization. Fine-tuning uses AdamW at 10^{-4} with weight decay 0.05, gradient clipping at norm 1.0, fp16 mixed precision and a batch size of 16 (4 GPUs $\times$ 2 images $\times$ 2 accumulation steps) under the standard $1\times$ and $3\times$ schedules, whose multi-step decay by 0.1 at epochs 8 and 11, or 27 and 33, follows 1000 warm-up iterations. The $1\times$ schedule trains at 1333×800 with horizontal flips; the $3\times$ schedule adds Swin's multi-scale training (short side resized to 480–800, with a random crop). We report the last epoch, with no selection on val2017; three of the four runs peaked there, and DSSD-T at $3\times$ peaked at epoch 34 with 50.4 box AP. Every run uses the same code, in which the column-major lag is the stride H on non-square inputs.

I.5 ADE20K SEMANTIC SEGMENTATION

Backbones are initialized from the last-epoch ImageNet-1k weights and fine-tuned in MMSegmentation (MMSegmentation Contributors, 2020) with UperNet (Xiao et al., 2018) and an auxiliary FCN head on ADE20K (Zhou et al., 2017) for 160k iterations. Fine-tuning uses AdamW at 6×10^{-5} with weight decay 0.05, 1500 warm-up iterations and a polynomial decay, 512×512 crops, gradient clipping at norm 1.0 and fp16 mixed precision. The standard recipe is batch 16 for 160k optimizer updates, and the footnotes of Table I.2 mark where our runs differ. An earlier DSSD-T run with a quarter of the updates, batch 16 through 4-step accumulation over 160k iterations, reached 47.2, and the table reports the re-run on the standard recipe. We report single-scale mIoU at the last

evaluation, as the comparators do, and multi-scale mIoU with whole-image inference at 0.5–1.75 times the 512×2048 single-scale test size and horizontal flips, the Swin and VMamba protocol. MMSegmentation 1.x applies these ratios to the original image instead, which runs half the validation images below the standard scale, and its test-time augmentation keeps every view's full-resolution logits on the GPU. We evaluate at the standard scales with a running average over the views, which gives the same merged prediction. Each model gains 0.1–0.6 points over its single-scale result.

Table I.2: Semantic segmentation on ADE20K with UperNet, 160k iterations, grouped as in Baty et al. (2026), from which the other rows are taken; an asterisk marks V2M's pyramid variants, as in Wang et al. (2024a). FLOPs are at 512×2048. Ours are the analytic FLOPs of the backbone, with the scan counted in its recurrent form as in Table 1, plus the decode head counted by mmengine without the auxiliary head, which is dropped at inference. The standard UperNet decode head accounts for 840–845G of our totals and the backbones for 80, 199 and 470G. The other rows' FLOPs are as published, and Mamba2D's are below the standard head alone (Appendix F). Our backbones cost more than VMamba's and Swin's at this input because their stages 3–4 use global attention, which is quadratic in the token count (Table F.1). mIoU (SS) / (MS): single- / multi-scale evaluation. Bold: best in group; underline: second; shaded: ours. Discussed in §4.2.

Backbone	#Param.	FLOPs	mIoU (SS)	mIoU (MS)
Nano				
Vim-Ti (Zhu et al., 2024)	13M	–	41.0	–
V2M-T (Wang et al., 2024a)	13M	–	41.4	42.0
EffVMamba-S (Pei et al., 2024)	29M	505G	41.5	42.1
M2D-N (Baty et al., 2026)	34M	355G	43.1	43.1
LocalVim-T (Huang et al., 2024)	36M	260G	<u>43.4</u>	<u>44.4</u>
DSSD-N	34M	921G	**46.1**	**46.2**
Tiny				
ResNet-50 (He et al., 2016)	66M	953G	42.1	42.8
Swin-T (Liu et al., 2021)	60M	945G	44.4	45.8
Vim-S (Zhu et al., 2024)	46M	–	44.9	–
V2M-S (Wang et al., 2024a)	46M	482G	45.1	46.1
MSVMamba-M (Shi et al., 2024)	42M	875G	45.1	45.4
VSSD-M (Shi et al., 2025)	42M	893G	45.6	46.0
EffVMamba-B (Pei et al., 2024)	65M	930G	46.5	47.3
MSVMamba-T (Shi et al., 2024)	65M	942G	47.6	48.5
VSSD-T (Shi et al., 2025)	53M	941G	47.9	48.7
VMamba-T (Liu et al., 2024b)	62M	948G	47.9	48.8
V2M-S* (Wang et al., 2024a)	62M	482G	48.2	49.0
M2D-T (Baty et al., 2026)	53M	485G	<u>48.9</u>	<u>49.3</u>
DSSD-T	54M	1042G	**50.7**	**51.2**
Small				
ResNet-101 (He et al., 2016)	85M	1030G	44.9	45.9
Swin-S (Liu et al., 2021)	81M	1039G	47.6	49.5
NAT-S (Hassani et al., 2023)	82M	1010G	48.0	49.5
2DMamba (Zhang et al., 2026)	92M	950G	48.6	49.3
ConvNeXt-S (Liu et al., 2022b)	82M	1028G	48.7	49.6
VMamba-S (Liu et al., 2024b)	82M	1028G	50.6	51.2
V2M-B* (Wang et al., 2024a)	82M	655G	50.8	51.3
M2D-S (Baty et al., 2026)	77M	768G	<u>51.7</u>	<u>51.8</u>
DSSD-S[†]	78M	1315G	**52.2**	**52.8**

[†]2 GPUs $\times$ 4 images: batch 8, 160k optimizer updates. Our MS column uses whole-image inference at 0.5–1.75$\times$ the 512×2048 test size with flips, the protocol of the comparators.

Table I.3: Object detection and instance segmentation on MS-COCO with Mask R-CNN under $1\times$ and $3\times$ schedules. Comparator rows as published: PVT (Wang et al., 2021), EffVMamba (Pei et al., 2024), M2D (Baty et al., 2026), MSVMamba (Shi et al., 2024), VSSD (Shi et al., 2025), ResNet (He et al., 2016), Swin (Liu et al., 2021), ConvNeXt (Liu et al., 2022b), VMamba (Liu et al., 2024b) and V2M (Wang et al., 2024a), whose asterisk marks its pyramid variant, as in its paper. AP^{b} / AP^{m}: box / mask average precision, subscripts: IoU threshold. Bold: best in group; underline: second. Discussed in §4.2; training details in Appendix I.4.

		Object Detection						**Instance Segmentation**					
		1× Schedule			**3× Schedule**			**1× Schedule**			**3× Schedule**		
Backbone	**#param.**	AP^{b}	$\mathrm{AP}^{\mathrm{b}}_{50}$	$\mathrm{AP}^{\mathrm{b}}_{75}$	AP^{b}	$\mathrm{AP}^{\mathrm{b}}_{50}$	$\mathrm{AP}^{\mathrm{b}}_{75}$	AP^{m}	$\mathrm{AP}^{\mathrm{m}}_{50}$	$\mathrm{AP}^{\mathrm{m}}_{75}$	AP^{m}	$\mathrm{AP}^{\mathrm{m}}_{50}$	$\mathrm{AP}^{\mathrm{m}}_{75}$
Micro/Nano													
PVT-T	33 M	36.7	59.2	39.3	39.8	62.2	43.0	35.1	56.7	37.3	37.4	59.3	39.9
EffVMamba-S	31 M	39.3	61.8	42.8	41.6	63.9	45.6	36.7	58.9	39.2	38.2	60.8	40.7
M2D-N	26 M	42.8	65.3	46.7	47.2	69.2	51.7	39.6	62.5	42.5	42.4	65.9	45.6
MSVMamba-M	32 M	43.8	65.8	47.7	46.3	68.1	50.8	39.9	62.9	42.9	41.8	65.1	44.9
VSSD-M	33 M	**45.4**	**67.5**	**49.8**	**47.7**	**69.7**	**52.1**	**41.3**	**64.5**	**44.6**	**42.8**	**66.5**	**46.0**
DSSD-N	26 M	43.9	66.2	48.0	47.0	69.3	51.5	40.3	63.1	43.4	42.4	66.1	45.8
Tiny													
ResNet-50	44 M	38.2	58.8	41.4	41.0	61.7	44.9	34.7	55.7	37.2	37.1	58.4	40.1
Swin-T	48 M	42.7	65.2	46.8	46.0	68.1	50.3	39.3	62.2	42.2	41.6	65.1	44.9
ConvNeXt-T	48 M	44.2	66.6	48.3	46.2	67.9	50.8	40.1	63.3	42.8	41.7	65.0	44.9
VSSD-T	44 M	46.9	69.4	51.4	48.8	70.4	53.6	42.6	66.4	45.9	43.6	67.6	46.9
VMamba-T	50 M	47.3	69.3	52.0	48.8	70.4	53.5	42.7	66.4	45.9	43.7	67.4	47.0
V2M-S*	50 M	47.6	69.4	52.2	49.0	70.6	53.5	42.9	66.5	46.3	43.8	67.5	47.2
M2D-T	44 M	48.5	**71.0**	53.1	**50.5**	**72.1**	**55.4**	**43.8**	**67.8**	47.1	**44.7**	68.9	**48.3**
DSSD-T	44 M	**48.6**	70.8	**53.5**	50.3	71.8	55.3	43.7	**67.8**	**47.2**	**44.7**	**69.0**	48.1