SHIFTED-WINDOW DIFFUSION WITH TEMPORAL-DISTANCE ATTENTION FOR SELF-SUPERVISED TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling drives modern machine learning, but its usual currency is parameters, data, and GPU memory. Holding the model and data fixed, we instead scale the supervision, namely the number W of prediction targets per input. Existing self-supervised pretexts for time series reconstruct targets within a single window or span at most two, so none of them can scale this axis. We propose SDTA (Shifted-window Diffusion with Temporal-distance Attention), a self-supervised pretraining framework whose encoder is fine-tuned for downstream forecasting. SDTA encodes the input window once and denoises W diffusion-noised target windows, each shifted a sampled distance into the future, so that one encoding must support prediction at many temporal distances. Temporal-distance attention injects each target's distance only into the attention keys, steering where the decoder attends rather than what it copies. Because the input is encoded once and the targets exist only in pretraining, raising W multiplies supervision at linear activation memory, fits one consumer GPU, and adds no deployment cost. Experiments on twelve standard forecasting benchmarks spanning energy, weather, finance, and traffic compare eight self-supervised methods under identical encoder capacity, one fixed configuration, no per-dataset tuning, and five-seed averaging. We further scale each baseline's own supervision axis: most possess no memory-feasible counterpart, and the two that do gain nothing from scaling it, whereas SDTA improves consistently along this axis, with the gain widening beyond the standard horizons. SDTA also achieves the lowest average MSE and MAE among the eight methods, and ranks top-two on ten of the twelve benchmarks by MSE and nine by MAE.

1 INTRODUCTION

Progress in machine learning is largely a story of scaling, and its currency is parameters, data, and GPU memory. Time-series forecasting has followed the same route, from larger backbones to web-scale foundation models (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024), which demand memory and compute far beyond the single consumer card most practitioners use. Self-supervised pretraining on the target dataset itself (Nie et al., 2023; Dong et al., 2023; 2024) is the memory-friendly alternative. That leaves a question: once the encoder and data are fixed, *what is left to scale?*

Our answer is the supervision itself. A pretext defines a set of prediction problems over the same data, and some pretexts let that set be enlarged, adding targets per input, without enlarging the model or the batch; each added target is an unlabeled slice of the same series, so the enlargement multiplies self-supervision, not annotation. We call such a control a *supervision axis* and require it to be *memory-feasible*: at fixed batch size its activation memory must grow at most linearly in that control. It trades a little pretraining time for stronger representations inside a fixed VRAM budget, and the cost disappears at deployment because only the encoder transfers.

Two gaps motivate our design. First, diffusion-based pretexts for time series denoise a *single* target window (Wang et al., 2024); their supervision is never scaled across many distance-controlled viewpoints, and the one pretext that does place such a target in another window is masking-based

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.

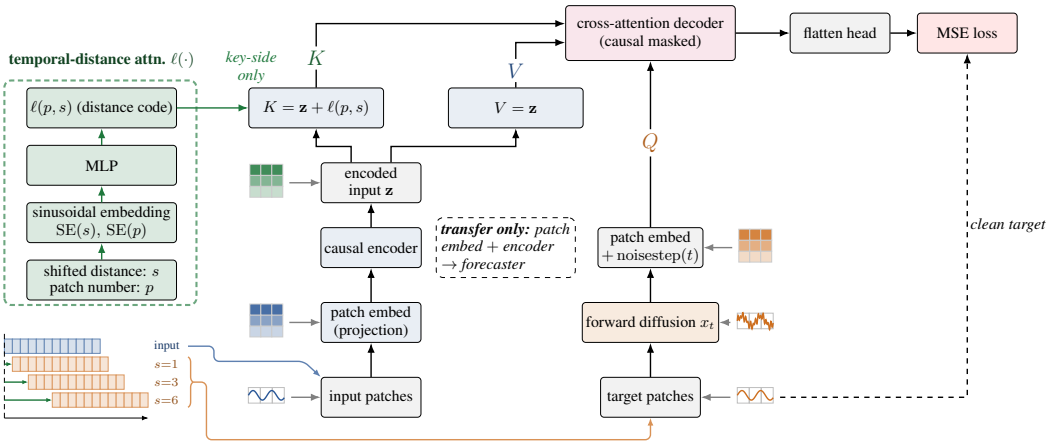

Figure 1: SDTA at a glance. *Bottom left:* shifted-window sampling with $W = 3$. One window is 12 patches (96 steps) and s is counted in patches, $s \in [1, 12]$; the first target is forced to $s = 1$, the rest random. *Main panel* (read bottom-up): the input is encoded *once* into $\mathbf{z}$, giving $V = \mathbf{z}$ and $K = \mathbf{z} + \ell(p, s)$. Each target is corrupted by forward diffusion, embedded into Q, and denoised by a causal cross-attention decoder under an MSE loss. The distance code enters the *key side only*, so distance routes *where* the decoder attends, not *what* it transports. Only the patch embedding and encoder transfer downstream.

and spans just two (Dong et al., 2024). Second, published comparisons are confounded by capacity: a reported win can reflect a bigger backbone rather than a better objective.

We address both gaps with **SDTA** (*Shifted-window Diffusion with Temporal-distance Attention*), a self-supervised pretraining pretext for forecasting (Figure 1). It extends diffusion denoising from same-window reconstruction to W target windows, each shifted a sampled distance s ahead of the input, which is encoded once and shared by all W problems. A *temporal-distance attention* code $\ell(p, s)$, added on the key side only ($K \neq V$), shapes *where* the decoder attends, not *what* it transports. The window count W is the supervision axis: supervision grows W-fold, activation memory at most linearly at fixed batch, and the axis exists only during pretraining.

To evaluate fairly, we build a capacity-matched benchmark spanning eight self-supervised methods across the masking, contrastive, predictive, and diffusion families (Appendix A.2, Table 7). All eight share the same encoder capacity, the same training budget, and *one* fixed configuration across all twelve datasets, with no per-dataset search for any method. Results therefore measure robustness of the pretext, not tuning effort.

Empirically, this axis pays off. Growing W lowers error at every capacity and on all twelve datasets, while the same sweep on each baseline's own axis yields nothing, so the scaling is unique to SDTA. Under the single fixed configuration SDTA is also the strongest of the eight overall: lowest twelve-dataset average error on both metrics, significantly ahead of six of the seven baselines on MSE and five on MAE, tied with the runner-up, and never among the worst anywhere. A stress test at $1.5\times$ the longest standard horizon widens the gain from scaling W.

Our contributions are:

- **A shifted-window diffusion pretext.** Diffusion denoising extended from same-window reconstruction to many distance-controlled, cross-window targets.
- **A memory-bounded scaling axis.** The window count W scales supervision at linear, fixed-batch memory. The gain grows on harder tasks, costs nothing at deployment, and no baseline reproduces it.
- **Continuous key-side distance routing.** A refinement rather than a pillar: a continuous distance code on the attention keys only, replacing the discrete lineage embeddings of prior work. Removing it costs $+0.0037$ MSE against $+0.0221$ for the core pretext.

- **A fair, capacity-matched benchmark.** One fixed configuration across twelve datasets and eight methods, with every method's own supervision axis swept, an extended-horizon stress test at $1.5\times$ the longest standard horizon, and Friedman/Wilcoxon tests for every headline comparison, including the one we do not win.

2 Related Work

Time-series forecasting backbones. PatchTST (Nie et al., 2023) established the current recipe, patch tokens with channel independence, which our encoder and every baseline's follows. Linear baselines (Zeng et al., 2023) showed capacity alone does not win, motivating our capacity-matched protocol.

Self-supervised pretraining for time series. Vision-style augmentation invariance transfers poorly to time series, whose semantics live in temporal evolution (Dong et al., 2023); the workable corruptions are patch masking and diffusion noising. Masking pretexts include PatchTST's own masked variant (Nie et al., 2023), SimMTM (Dong et al., 2023), TimeMAE (Cheng et al., 2023), and TimeSiam's two-window siamese reconstruction with discrete lineage embeddings (Dong et al., 2024); CPC (van den Oord et al., 2018) and SimTS (Zheng et al., 2023) instead predict future latents, contrastively or under a positive-only loss; TimeDART (Wang et al., 2024) brought diffusion denoising (Ho et al., 2020; Nichol & Dhariwal, 2021) to time-series pretraining, but within a single window. Only TimeSiam places a distance-controlled target in another window, and it spans exactly two; none scales supervision across *many* windows. Table 7 positions SDTA. Raising the view count is familiar in vision, where multi-crop (Caron et al., 2021) adds crops of the *same* instant under an invariance objective, so each view asks the encoder to discard more; our W targets are prediction problems at temporal distances under a recovery objective, so each asks it to model more.

Foundation models. Zero-shot forecasters pretrained at web scale (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024) sit at the opposite end of the memory spectrum; we target the regime they exclude, one modest GPU and pretraining on the target data itself.

3 Method

Problem and notation. Given a multivariate series of C channels, we forecast H future steps per channel. Each channel is an independent univariate series sharing one patch embedding and one encoder (Nie et al., 2023). A length-L window is split into non-overlapping patches of length P with stride $S=P$, giving $N = \lfloor (L-P)/S \rfloor + 1$ tokens. Each window is instance-normalized per channel, its mean and standard deviation removed before patching and restored at the head.

3.1 Pretext: Shifted-Window Diffusion

Each pretraining example pairs one clean *input* window (the anchor) with W *target* windows shifted forward in time. The distance s, counted in patches, is drawn from $s \in [1, s_{\max}]$: the first target is a *forced* $s=1$ *target*, a near-zero-cost stabilizer, and the remaining $W-1$ are independent draws.

Each target window is corrupted by forward diffusion (Ho et al., 2020),

$$q(x_{t_i} \mid x_0) = \mathcal{N}\big(x_{t_i};\ \sqrt{\gamma_{t_i}}\, x_0,\ (1-\gamma_{t_i})\mathbb{I}\big), \tag{1}$$

where $\gamma_{t_i} \in (0, 1]$ is the cumulative signal-retention coefficient of a cosine schedule (Nichol & Dhariwal, 2021), decreasing in t_i, and the step t_i is drawn independently for each patch i so a single window mixes difficulty levels. Unlike jitter-style augmentation, which enforces invariance and thereby asks the encoder to *discard* fine-grained dynamics, diffusion demands *recovery*. The model must reconstruct the clean signal, and this forces it to model temporal structure across a controlled, easy-to-hard curriculum.

More targets are not more labels. Every target window is an unlabeled slice of the same raw series that any single-window pretext already reads, produced by shifting an index into the pretraining split, so raising W adds no annotation and no new data. The contrast with supervised

Algorithm 1 SDTA pretraining step for one input window (the anchor) and its W target windows

Input: series X; anchor start a; window length L, split into N patches of length P
Parameter: number of target windows W, max shift $s_{\max}$, diffusion steps T
Output: pretraining loss $\mathcal{L}$. All windows instance-normalized.

$\mathbf{z} \leftarrow \mathrm{Enc}(\mathrm{Embed}(X[a{:}a{+}L]))$, $V \leftarrow \mathbf{z}$ {anchor: encoded once, shared by all W targets}
for $w = 1$ **to** W **do** {one denoising problem per target window}
  $s_w \leftarrow 1$ if $w{=}1$ else $s_w \sim \mathcal{U}\{1, \dots, s_{\max}\}$ {shift in patches, sampled independently}
  $x_0^{(w)} \leftarrow X[a{+}s_wP : a{+}s_wP{+}L]$ {target window, s_w patches ahead of the anchor}
  $t_i \sim \mathcal{U}\{1, \dots, T\}$ per patch i; $\varepsilon \sim \mathcal{N}(0, \mathbb{I})$ {fresh noise for every target}
  $x_t^{(w)} \leftarrow \sqrt{\gamma_{t_i}}\, x_0^{(w)} + \sqrt{1-\gamma_{t_i}}\, \varepsilon$ {forward diffusion}
  $Q^{(w)} \leftarrow \mathrm{Embed}(x_t^{(w)}) + \mathrm{SE}(t_i)$
  $K_p^{(w)} \leftarrow \mathbf{z}_p + \ell(p, s_w)$ for key positions $p = 1, \dots, N$ {key side only}
  $\hat{x}_0^{(w)} \leftarrow \mathrm{Head}(\mathrm{Dec}(Q^{(w)}, K^{(w)}, V))$ {causal masked}
end for
return $\mathcal{L} = \frac{1}{W} \sum_{w=1}^{W} \|\hat{x}_0^{(w)} - x_0^{(w)}\|_2^2$

multi-horizon forecasting is structural: that trains the deployed predictor on input–label pairs at the evaluation horizons, whereas SDTA's targets live only inside the pretext, at shifts unrelated to the downstream prediction lengths, and feed a decoder discarded before fine-tuning.

3.2 Architecture

Figure 1 shows the full pretext. The input window passes through a shared linear patch embedding and a *causal* Transformer encoder, producing $\mathbf{z}$, computed *once* and reused by all W views. Causality is needed because patch-aligned sampling makes target patch i at distance s the *same* raw patch as anchor patch $i{+}s$, so a bidirectional encoder would fold the answer into the keys. Each noisy target passes through the same embedding plus a sinusoidal embedding of its per-patch noise step, forming the decoder queries. A causal masked cross-attention decoder reads them against $K = \mathbf{z} + \ell(p, s)$ and $V = \mathbf{z}$; independently for self- and cross-attention it also drops a fraction of the strictly-past query–key links at random each pass, never the diagonal. We call that fraction the *past-link mask ratio* m, distinct from the patch-masking ratios of the masked pretexts, and fix $m = 0.5$: half the past links keep the anchor available as context, and dropping the rest stops the decoder resting on any fixed link. It is the one design value we did not sweep.

3.3 Temporal-Distance Attention

The distance s must condition the decoder, but a naive injection fails: s is one value per view, and a vector added identically to every key is cancelled by the shift-invariant softmax. Temporal-distance attention (TDA) therefore encodes the key position p with s:

$$\ell(p, s) = \mathrm{MLP}\big(\mathrm{concat}[\,\mathrm{SE}(p),\ \mathrm{SE}(s)\,]\big), \tag{2}$$

where $\mathrm{SE}(\cdot)$ is a sinusoidal embedding. Because $\ell(p, s)$ differs per key it survives the softmax, biasing *where* each query attends as a function of distance.

Two choices distinguish this from prior work. First, the code is *continuous* in s, replacing TimeSiam's discrete lineage embeddings, which matters because W targets sample a wide, fine-grained range of distances. Second, it is injected on the *key side only* ($K \neq V$): keys set where to attend and values set what is copied, so routing by distance should touch the keys and leave the values clean. TimeSiam instead adds its embedding before the encoder, contaminating both K and V.

3.4 Training Objective and Transfer

Algorithm 1 states one pretraining step. The loop is expository. Given $\mathbf{z}$ the W targets are independent, and the implementation stacks them along the batch dimension. The decoder predicts the clean target directly (x_0-prediction, not noise prediction), in raw space:

$$\mathcal{L} = \mathbb{E}_{a,\, s,\, t,\, \varepsilon} \left\| \hat{x}_0^{\mathrm{tgt}} - x_0^{\mathrm{tgt}} \right\|_2^2. \tag{3}$$

Table 1: Main results. Input length = 96, prediction length $\in \{96, 192, 336, 720\}$, PEMS $\in \{12, 24, 48, 96\}$, 5-seed average. **Bold** / underline = best / 2nd; **SDTA capacity: Medium_4** ($E = 2$ encoder blocks, $W = 4$; see *Capacity naming*). For scale, the same encoder trained from scratch averages 0.3738 MSE here (Table 3).

	SDTA		TimeDART		TimeSiam		TimeMAE-CI		PatchTST-SSL		SimMTM		CPC		SimTS	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.4348**	**0.4340**	0.4397	0.4368	0.4433	0.4373	0.4509	0.4392	0.4722	0.4467	0.4438	0.4376	0.4619	0.4386	0.4616	0.4434
ETTh2	**0.3744**	**0.3994**	0.3781	0.4009	0.3815	0.4077	0.3819	0.4068	0.3826	0.4045	0.3764	0.4030	0.3779	0.4010	0.3840	0.4047
ETTm1	0.3854	0.3979	0.3958	0.4016	0.3888	0.3984	0.3876	0.3988	0.3971	0.4020	**0.3841**	0.3973	0.3872	**0.3968**	0.3984	0.3984
ETTm2	**0.2811**	**0.3283**	0.2836	0.3307	0.2862	0.3337	0.2849	0.3336	0.2860	0.3317	0.2916	0.3383	0.2854	0.3322	0.2846	0.3303
exchange	0.3623	0.4037	0.3656	0.4060	0.3636	0.4057	0.3592	0.4026	**0.3561**	0.4018	0.3613	0.4051	0.3568	**0.4006**	0.3641	0.4072
weather	0.2638	0.2838	0.2650	0.2844	0.2655	0.2860	**0.2620**	**0.2822**	0.2724	0.2899	0.2674	0.2878	0.2683	0.2874	0.2749	0.2927
traffic	**0.5443**	**0.3507**	0.6026	0.3856	0.5576	0.3648	0.5802	0.3731	0.6459	0.4170	0.5777	0.3711	0.5826	0.3841	0.6413	0.3952
electricity	0.2176	0.3032	0.2245	0.3134	0.2189	0.3072	0.2177	0.3004	0.2305	0.3155	**0.2152**	**0.2949**	0.2165	0.2965	0.2250	0.3067
PEMS03	0.2691	0.3488	0.3101	0.3769	0.3239	0.3898	**0.2645**	**0.3405**	0.3268	0.3807	0.3196	0.3774	0.3015	0.3685	0.3710	0.4033
PEMS04	0.3474	0.3995	0.3715	0.4175	0.3736	0.4170	**0.3418**	**0.3901**	0.4020	0.4275	0.3800	0.4169	0.3791	0.4153	0.4109	0.4307
PEMS07	**0.3019**	**0.3658**	0.3251	0.3829	0.3369	0.3925	0.3124	**0.3658**	0.3615	0.3998	0.3455	0.3952	0.3362	0.3844	0.3738	0.4049
PEMS08	0.3517	0.3855	0.3755	0.4034	0.4054	0.4222	**0.3466**	**0.3766**	0.4148	0.4211	0.4087	0.4184	0.3791	0.3990	0.4236	0.4208
4-ETT	**0.3689**	**0.3899**	0.3743	0.3925	0.3750	0.3943	0.3763	0.3946	0.3845	0.3962	0.3740	0.3941	0.3781	0.3922	0.3821	0.3942
4-Wide	**0.3470**	**0.3354**	0.3644	0.3473	0.3514	0.3409	0.3548	0.3396	0.3762	0.3560	0.3554	0.3397	0.3561	0.3421	0.3763	0.3504
4-PEMS	0.3175	0.3749	0.3456	0.3952	0.3599	0.4054	**0.3163**	**0.3683**	0.3763	0.4073	0.3634	0.4020	0.3490	0.3918	0.3948	0.4149
12-ds	**0.3445**	**0.3667**	0.3614	0.3783	0.3621	0.3802	0.3491	0.3675	0.3790	0.3865	0.3643	0.3786	0.3610	0.3754	0.3844	0.3865

The expectation runs over anchors, distances, per-patch steps, and noise draws, spanning all W views. Downstream, only the pretrained patch embedding and encoder transfer, and forecasting adds a linear flatten head.

Why memory stays bounded. The anchor is encoded once regardless of W, and each extra target adds one decoder pass over a fixed-length window, so at fixed batch activation memory is at most linear in W (Appendix G, Table 32).

4 A Fair, Capacity-Matched Benchmark

Datasets. The twelve benchmarks form three groups: *4-ETT*, the four ETT series (7 channels each); *4-Wide*, exchange (8), weather (21), electricity (321), traffic (862); and *4-PEMS*, PEMS03/04/07/08 (358, 307, 883, 170). Prediction lengths are $\{96, 192, 336, 720\}$ except on PEMS, whose finer sampling makes $\{12, 24, 48, 96\}$ standard. Splits are standard: $12/4/4$ months for ETT (Zhou et al., 2021), $0.7/0.1/0.2$ for the other wide sets (Wu et al., 2021; Lai et al., 2018), $0.6/0.2/0.2$ for PEMS (Song et al., 2020).

One fixed configuration. Prior work tunes per dataset, which rewards search budget. We instead fix *one* configuration for all twelve datasets and all eight methods: Adam optimizer, pretrain learning rate 10^{-3} (ExponentialLR, $\times 0.9$/epoch, 50 epochs), downstream max learning rate 10^{-4} (OneCycleLR, $\mathrm{pct_start} = 0.3$, 10 epochs, patience 3), batch size 16, dropout 0.2, head dropout 0.1, input length / patch / stride $= 96/8/8$, model width $d = 32$, feed-forward width 64, 4 attention heads, $E = 2$ encoder blocks, diffusion steps $T = 1000$. Only each method's own design parameter and the random seed differ. Every number we report is a 5-seed average over the fixed seeds 2021–2025, which seed Python, NumPy, and PyTorch at the start of every stage. SimMTM's series-wise contrastive matrix is quadratic in the channel count, so on high-channel data we invoke the channel-subsampling option its official code provides, keeping a random 10% per batch, but only in *pretraining*: its fine-tuning and evaluation read every channel, so its accuracy is comparable. Only its peak memory in Table 32 is measured under subsampling, and therefore understates its true cost.

Accuracy runs used Ubuntu 24.04 on a node with two Intel Xeon Gold 6530 CPUs, 512 GB DDR5, and eight NVIDIA RTX 5090 (32 GB) GPUs, one run per GPU. All wall-clock and memory measurements used a single RTX 4090 (24 GB) with an i7-13700K and 70 GB RAM, under Ubuntu 22.04.5, Python 3.11.14, PyTorch 2.5.1 (CUDA 12.4).

What we searched, and what we did not. Three axes are swept, as *reported experiments* rather than a search for a winning cell: $W \in \{1, 2, 4, 8\}$ (Figure 2), capacity $\in$ {Small, Medium, Large},

and each baseline's own axis (TimeSiam W, CPC $K \in \{1, 2, 4\}$; Appendix C, Table 13). The selection criterion is the seed-averaged twelve-dataset MSE, applied once and globally, never per dataset. A further sweep in Appendix C.1 changed nothing: the shift bound $s_{\max} \in \{12, 18, 24\}$, crossed with $W \in \{2, 4, 8\}$ on the six lower-channel datasets, moves the matched-cell average MSE by 0.0001 ($0.3555/0.3555/0.3556$) and no per-W group average by more than 0.0008, so we keep $s_{\max} = 12$, exactly one window.

The remaining hyperparameters are fixed by the protocol rather than tuned, and every method receives the identical set: optimizer, learning rates, schedules, dropouts, and layer widths take the standard PatchTST values (Nie et al., 2023) that all eight implementations build on, and the epoch budgets (50 pretraining, 10 fine-tuning) follow TimeDART (Wang et al., 2024). Batch size 16 answers the memory budget, not accuracy: activation cost differs by nearly an order of magnitude across the eight pretexts (Table 32), and 16 is where the whole matrix fits one card. Patch length 8 is a compromise across granularities spanning five minutes to one hour. We ran no search over any of these, our own method included.

Evaluation metrics. Over a test set $\mathcal{D}$ of windows with horizon H and C channels,

$$\mathrm{MSE} = \frac{\sum_{i,h,c} \left(\hat{y}^i_{h,c} - y^i_{h,c}\right)^2}{|\mathcal{D}|\, H\, C}, \quad \mathrm{MAE} = \frac{\sum_{i,h,c} \left|\hat{y}^i_{h,c} - y^i_{h,c}\right|}{|\mathcal{D}|\, H\, C}. \tag{4}$$

MSE is dominated by the largest errors and MAE is robust to them, so leading on both rules out exploiting either bias. Both use the globally standardized scale, with no inverse transform.

Matched encoder capacity. All eight methods transfer an identical encoder, which is why TimeMAE runs in its *channel-independent* form (TimeMAE-CI): its official input projection mixes channels, tying the encoder's width to each dataset's channel count. We swap in the shared per-channel projection the other seven already use and leave its decoupled-MAE objective untouched, so the pretext stays the only variable. Pretraining heads necessarily differ; each is what its pretext *requires*, so it is compared rather than controlled (He et al., 2022).

Each method keeps its own design. Every baseline runs at the pretext configuration recommended in its own paper, and each baseline with a scalable supervision axis is reported at its most favorable axis value (Table 13), at its strongest rather than weakened by our protocol. Ours uses $m = 0.5$, $s_{\max} = 12$, $W = 4$; apart from such method-specific choices and the seed, every hyperparameter is the fixed configuration above.

Capacity naming. E counts encoder blocks and D pretraining-decoder blocks, at width $d = 32$: Small ($E = 1, D = 1$; 8,896 transferred parameters), Medium ($E = 2, D = 1$; 17,440), Large ($E = 2, D = 2$; 17,440). Only the encoder transfers. We write Size_W; the default is **Medium_4**.

5 Results

5.1 Main Results

Table 1 reports all eight methods at the single fixed configuration. SDTA wins the 12-ds average on both metrics, is best on $5/12$ datasets, **top-two on** $10/12$, and is the only method inside the top two of every group average. The baselines specialize instead: TimeMAE-CI wins 4-PEMS, whose dense, high-channel traffic favors masked reconstruction, but trails elsewhere, and every other method has multiple weak datasets. That breadth under one configuration is the finding. The win is stable across seeds (12-ds std: SDTA $0.3445 \pm .0010$ vs. TimeMAE-CI $0.3491 \pm .0016$); consistency across *datasets* is tested below, where the gap to TimeMAE-CI, alone among the seven, fails to reach significance.

5.2 Scaling the Supervision Axis

Figure 2 (left) sweeps $W \in \{1, 2, 4, 8\}$ at all three capacities. Two observations hold at every capacity. First, **scaling leads to better performance**: growing W from 1 to 4 lowers the 12-ds MSE monotonically (a -4.1% relative gain at Medium), and the improvement is visible in every

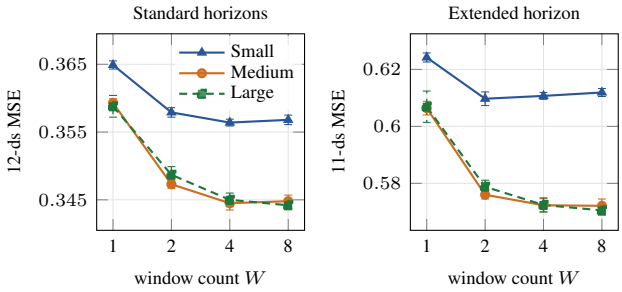

Figure 2: Scaling the supervision axis at three capacities (Small $E=1, D=1$; Medium $E=2, D=1$; Large $E=2, D=2$): 5-seed means with cross-seed standard deviations, the W axis equally spaced per doubling. *Left:* standard horizons. *Right:* the extended horizon 1080 (PEMS 144), where exchange cannot supply test windows and the average is over eleven datasets. Error falls monotonically to $W=4$ at every capacity at the standard horizons and at Medium and Large on the extended horizon; at Small the extended minimum is instead $W=2$, a 0.0010 gap inside the seed spread, and past $W=4$ every change stays inside the combined unpaired seed spread. Per-cell values are in Appendix C.

dataset group (Table 13). The step dwarfs the seed noise, -0.0148 at Medium against a cross-seed standard deviation of 0.0010, and the deeper encoders extract more from it than the shallow one. The count itself pays, not just the presence of varied distances: $W=2$ already draws its second target at a random s, yet $W=2\to4$ still lowers MSE on ten of twelve datasets ($p=0.021$). Second, **the axis saturates at $W=4$ at the standard horizons**: every $W=4\to8$ aggregate difference is smaller than the combined seed spread of the cells it compares, so we default to $W=4$ everywhere. The one consistent signal past $W=4$ sits at the extended horizon, where Large improves in five of five paired seeds (mean -0.0020): the deeper pretraining decoder has headroom only a stretched horizon exposes, but the margin stays inside the combined unpaired spread and $W=8$ doubles pretraining time, so the default stands.

5.3 Scaling Every Baseline's Own Axis

Is the gain in Figure 2 just "more targets help"? We check every baseline for a memory-feasible supervision axis and scale every axis that exists. A pretext scales only if some control adds *training signal*, not merely capacity, at memory at most linear at fixed batch. Five pretexts fail this gate: TimeDART, PatchTST-SSL, and TimeMAE-CI are single-window, SimTS grows only depth, and SimMTM's extra masked views cost *quadratic* memory. CPC's negative count fails too, since at fixed batch the distinct-negative pool is capped, but its step count K passes. Exactly three axes are feasible: SDTA's W, TimeSiam's window count W, and CPC's step count K.

Table 13 scales all three, and the result is one-sided: TimeSiam's and CPC's best axis values are both *one*. CPC is the sharpest test, since its K future steps are already distance-indexed, each with its own head, so distance conditioning alone does not make an axis pay. **Only SDTA converts its axis into scalability**: distance-routed cross-attention gives each added window a genuinely new prediction problem rather than more of the same reconstruction.

5.4 Memory and Runtime Cost

Table 32 prices the axis in both currencies. Every entry is measured at batch 16 with one dataloader worker on an idle card: memory is the per-process allocation peak, time the mean of six epochs with the first dropped, at one seed since cost is near-deterministic. Memory is linear in channels and in W: on the widest dataset the three SDTA rows read $1861\to2890\to4874$ MiB for $W=1,2,4$. Time follows the same law, $48\to75\to139$ seconds per pretraining epoch, 1.5–1.9$\times$ per doubling on high-channel data and flat on small ones. **At $W=2$ SDTA already reaches the second-best 12-ds MSE of the entire benchmark, at a memory and time bill comparable to TimeSiam's** (2890 MiB and 75 s against 2596 MiB and 147 s), and the default $W=4$ then buys the best MSE for roughly twice that.

Table 2: Two-sided Wilcoxon signed-rank tests over the twelve datasets, standard horizons; one paired observation is one dataset's 5-seed mean. "wins" counts datasets where SDTA (resp. the high axis value) has the lower MSE. **Bold** = significant at $\alpha = 0.05$; $^-$ marks significance toward *degradation*.

Comparison	wins	p (MSE)	p (MAE)
SDTA vs. each baseline (Holm-corrected, 7 comparisons)			
TimeMAE-CI	7/12	0.380	0.850
CPC	10/12	**0.028**	0.054
TimeDART	12/12	**0.003**	**0.003**
TimeSiam	12/12	**0.003**	**0.003**
SimMTM	9/12	**0.028**	**0.028**
PatchTST-SSL	11/12	**0.006**	**0.004**
SimTS	12/12	**0.003**	**0.003**
Own supervision axis, low → high (uncorrected)			
SDTA $W:1\to4$	12/12	**0.0005**	**0.001**
SDTA $W:2\to4$	10/12	**0.021**	0.064
TimeSiam $W:1\to4$	4/12	0.110	0.034^-
CPC $K:1\to4$	3/12	0.012^-	0.027^-

Table 3: Ablations: change in 12-ds MSE relative to the default config (Medium_4), 5-seed average; positive = worse. *random init plus a pretrained patch embedding, see text.

Ablation arm	12-ds ΔMSE
Core pretext	
random init (no pretraining)	$+0.0293$
linear probe (frozen encoder)	$+0.0272$
no anchor conditioning*	$+0.0228$
no diffusion (clean targets)	$+0.0221$
Design refinements	
broadcast token (discrete, TimeSiam-style)	$+0.0038$
no temporal-distance attention	$+0.0037$
no forced $s=1$ target	$+0.0015$
$K=V$ (code on keys and values)	$+0.0009$

The cost is also paid once and never reaches deployment: downstream fine-tuning is flat in W, $26/26/30$ seconds per epoch on PEMS07 for $W=1,2,4$ (Appendix G), because the pretext and its decoder are discarded before transfer. That leaves SDTA among the cheapest methods to fine-tune.

5.5 Extended Prediction Length

If multi-window supervision teaches genuinely longer-range structure, its payoff should *grow* where forecasting is hardest. We therefore stress-test one deliberately hard horizon, $1.5\times$ the longest standard one, on eleven datasets under the *same* configuration with no re-tuning. In Table 17 (Appendix D), SDTA keeps the lowest 11-ds MSE of all eight models, the second-lowest MAE behind TimeMAE-CI (MAE aggregates are in Appendix D), and **no weak dataset** even at this stretched horizon: its worst per-dataset rank is third, against sixth for the next most consistent method and eighth for four of the seven baselines.

The payoff also grows where the task gets harder: in the right panel of Figure 2, $W=1\to4$ lowers the 11-ds MSE by $\mathbf{-5.6}\%$ at Medium against -4.1% at the standard horizons, and the capacity pattern repeats.

5.6 Statistical Significance

Aggregate averages hide how a lead is distributed, so we test it. Following Demšar (2006), one dataset is one paired observation: the four prediction lengths averaged, then the five seeds, exactly the cells of Table 1.A Friedman omnibus rejects equivalence of the eight methods (MSE $\chi^2=42.3$, $p=4.6\times10^{-7}$; MAE $\chi^2=34.6$, $p=1.3\times10^{-5}$) with SDTA at the best average rank on both, and Table 2 reports two-sided Wilcoxon signed-rank tests against each baseline, Holm-corrected (Holm, 1979).

SDTA is significantly ahead of six baselines on MSE and five on MAE. **The exception is the runner-up, TimeMAE-CI**, whose gaps reach significance on neither metric: against that one method our claim is parity, not superiority. Ties are expected on a saturated benchmark, where a recent large-scale reproduction found almost no forecaster surviving strict pairwise testing (Brigato et al., 2025), and our ban on per-dataset tuning removes the usual source of separation.

The supervision axis, which is what the paper is about, is unambiguous: raising W lowers MSE on **all twelve datasets** and MAE on eleven, while TimeSiam's axis is flat on MSE and significantly *harmful* on MAE, and CPC's is significantly harmful on both. The comparison we win decisively is the scaling one, and the tie is with a method that has no axis to scale. At the extended horizon SDTA

keeps the best rank, stays significant against five baselines, and the W axis holds ($p = 0.007$); that eleven-dataset test table is in Appendix E.

6 ABLATIONS

Table 3 removes one ingredient at a time from the default configuration, and the arms split into two tiers.

The core pretext is load-bearing. Each core arm costs roughly six to thirty times what any refinement costs. Without anchor conditioning the encoder receives no gradient, so that arm is random init plus a trained patch embedding, and the gap between the two arms is what the embedding alone recovers. Multi-window sampling (Figure 2), diffusion, and pretrain-then-fine-tune only win together.

The attention designs are targeted refinements. The distance code is the larger of the two: removing it costs $+0.0037$ and swapping it for a discrete broadcast token $+0.0038$, both about $3.7\times$ the default's cross-seed standard deviation of 0.0010, and each is worse on ten of twelve datasets, all three group averages, and all five paired seeds. The key-side restriction is smaller: $K = V$ costs $+0.0009$, inside that spread, worse on six of twelve datasets and four of five seeds. Both grow 1.6–$3\times$ at the extended horizon ($+0.0059$, $+0.0027$), so key-side routing earns its place where the horizon is longest, not where the model is largest.

7 LIMITATIONS

A single input geometry and width. All results use one input length (96), one patch size (8), and one width ($d = 32$); other lookbacks, granularities, m, and standard widths such as $d = 128$ are not swept. **Epochs, not FLOPs, are equalized.** The protocol fixes epochs and encoder capacity, not pretraining compute: $W = 4$ costs about $3\times$ what $W = 1$ does per epoch (Table 32). **Not first on every dataset.** The lead holds at the aggregate level; on a few datasets, and on the 4-PEMS group narrowly, another method wins. **A statistical tie with the runner-up.** Over twelve datasets the margin over TimeMAE-CI is not significant on either metric (Table 2); parity, not dominance, is what we claim there. What separates them is the scaling axis, significant for us and absent for TimeMAE-CI. **Windows are VRAM-bounded.** Under a fixed VRAM budget W cannot grow without limit, so the multi-window axis has a practical ceiling. Linear memory defers the wall, but it does not remove it.

8 CONCLUSION

We proposed SDTA, a pretext that scales *supervision* rather than model size: one encoded anchor conditions the denoising of W distance-controlled target windows, and a continuous temporal-distance code routes attention on the key side only. At matched capacity across twelve datasets it attains the best average MSE and MAE, scaling W improves every group at linear memory inside a consumer GPU while no baseline profits from its own axis, and a stretched horizon amplifies the gain. Code, configurations, and analysis scripts accompany this submission, to be released under an MIT license.

AI USE STATEMENT

In this work, we used generative AI tools for three tasks with required disclosure. They assisted in implementing the methods in code. They assisted in analyzing the experimental results and offered interpretations, which the authors accepted or rejected. They suggested one experiment, the *no anchor conditioning* ablation of Table 3. We have not used generative AI tools to generate synthetic data sets, to develop the conceptual framework, to propose or refine the hypotheses, to assist with translation, or to clean or reformat datasets. Formulating or proving mathematical claims and qualitative or thematic data analysis are not applicable to this work. The idea, the architecture and its design details, the extended-prediction-length experiment, the capacity classes, the single patch size

and model size shared across all datasets, and every other ablation are the authors' own. Additionally, we used generative AI tools to search for related work, to write the pseudocode of Algorithm 1, to create and edit software code, to organize experimental results and write documentation, to draft skeleton versions of parts of the paper, to edit the paper for readability, and to format the LaTeX source. We have reviewed all AI-assisted work. The authors audited every section of the paper, ran and tested the AI-assisted code themselves, and checked each AI-provided reading of the results against the recorded numbers. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This work uses only publicly available benchmark datasets (ETT, exchange, weather, electricity, traffic, and PEMS) under their original licenses, involves no human subjects, personal data, or crowdsourced annotation, and introduces no new dataset. The method is a self-supervised pretraining pretext for time-series forecasting whose deployment cost equals that of the underlying encoder; we are not aware of application-specific risks beyond those of standard forecasting models. The authors declare no conflicts of interest.

Reproducibility statement

Every number in this paper is a five-seed average over the fixed seeds 2021–2025 under one fixed configuration; the configuration, the per-method design parameters, the two baseline adaptations, the datasets and splits, and the hardware and software are documented in Appendix A. Algorithm 1 states one pretraining step, and the per-dataset scores, per-cell sweep values, paired statistical inputs, and full cost tables behind every aggregate appear in Appendices B–G. The anonymized code, the run configurations for every reported cell, and the aggregation and significance-test scripts are described in Appendix H and are uploaded with this submission as an anonymized archive.

References

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Lorenzo Brigato, Rafael Morand, Knut Strømmen, Maria Panagiotou, Markus Schmidt, and Stavroula Mougiakakou. There are no champions in supervised long-term time series forecasting. *Transactions on Machine Learning Research*, 2025.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Mingyue Cheng, Qi Liu, Zhiding Liu, Hao Zhang, Rujiao Zhang, and Enhong Chen. TimeMAE: Self-supervised representations of time series with decoupled masked autoencoders. *arXiv preprint arXiv:2303.00320*, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

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

Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. SimMTM: A simple pre-training framework for masked time-series modeling. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Jiaxiang Dong, Haixu Wu, Yuxuan Wang, Yun-Zhong Qiu, Li Zhang, Jianmin Wang, and Mingsheng Long. TimeSiam: A pre-training framework for Siamese time-series modeling. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

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

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 8162–8171, 2021.

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

Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *AAAI Conference on Artificial Intelligence*, 2020.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Daoyu Wang, Mingyue Cheng, Zhiding Liu, Qi Liu, and Enhong Chen. TimeDART: A diffusion autoregressive transformer for self-supervised time series representation. *arXiv preprint arXiv:2410.05711*, 2024.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 22419–22430, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128, 2023.

Xiaochen Zheng, Xingyu Chen, Manuel Schürch, Amina Mollaysa, Ahmed Allam, and Michael Krauthammer. SimTS: Rethinking contrastive representation learning for time series forecasting. *arXiv preprint arXiv:2303.18205*, 2023.

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.

APPENDIX

This appendix contains the material the main text defers, together with the full evidence behind claims the main text reports only in aggregate. Nothing here supersedes the main text: every number below either appears in the main text verbatim or decomposes a number that does, or is evidence the main text cites without printing. Where a rounding convention makes the two differ in the last digit, we say so in the caption.

- **Appendix A** documents the protocol: the one fixed configuration shared by all eight methods, each method's own design parameter, the two baseline adaptations, the capacity classes and their parameter counts, the twelve datasets, and the hardware and randomness.
- **Appendix B** expands the main table into per-dataset scores with cross-seed standard deviations and into per-dataset rank matrices for both metrics.
- **Appendix C** gives the per-cell values behind Figure 2 of the main text, the same sweep resolved by dataset group, and the $s_{\max}$ sweep the main text cites but does not print.
- **Appendix D** gives the per-dataset detail behind the extended-horizon table, including the rank matrix that supports the worst-rank claim.
- **Appendix E** gives the complete statistical apparatus: the paired input matrices, uncorrected and Holm-corrected p values, the Friedman omnibus and average ranks, and the eleven-dataset tests at the extended horizon.
- **Appendix F** expands the ablation table into group and per-dataset scores, paired per-seed counts, and the extended-horizon arms.
- **Appendix G** restores the memory and wall-clock tables to their full five-dataset width and adds the cost–accuracy frontier.
- **Appendix H** describes the accompanying code.

Two protocols, never mixed. Two measurement protocols appear below and must not be combined. *Accuracy* numbers are five-seed means over the seeds 2021–2025 on twelve datasets, or eleven at the extended horizon, with one exception flagged in its own caption: the $s_{\max}$ sweep of Appendix C.1 is a three-seed sub-sweep on six datasets. *Cost* numbers are single-seed measurements at seed 97 taken on one idle GPU at batch 16 with one dataloader worker, and are relative costs under a fixed harness rather than benchmark wall-clock. Every table states which protocol it follows.

Prediction lengths. Unless a caption says otherwise, a per-dataset score is the mean over the four standard prediction lengths: $\{96, 192, 336, 720\}$ for every dataset except the four PEMS series, whose finer five-minute sampling makes $\{12, 24, 48, 96\}$ standard. The extended horizon is the single length 1080, or 144 on PEMS.

A Experimental Protocol

The main text's central methodological claim is that the eight methods are compared at matched capacity under one fixed configuration with no per-dataset tuning. This appendix is the evidence for that claim. Every value below was read out of the recorded configuration of the final runs themselves rather than transcribed from the main text, so a deviation between the two would show up here.

A.1 The Single Fixed Configuration

Table 4 is the shared configuration and Table 5 is the exhaustive list of departures from it.

One point of wording deserves care. The main text compresses the protocol into a single sentence beginning "Adam optimizer, pretrain learning rate 10^{-3} (ExponentialLR ...)", which describes SDTA's own optimizer and schedule. That sentence is shorthand, and the main text states the governing rule two paragraphs later: *every baseline runs at the pretext configuration recommended in its own paper*. Accordingly the epoch budgets, batch size, dropouts, widths and patch geometry are shared verbatim by all eight methods, the base rate is 10^{-3} in pretraining for all eight and 10^{-4} downstream for seven of the eight, the exception being PatchTST-SSL, whose released fine-tuning routine halves the downstream rate to 5×10^{-5}, and the downstream optimizer is Adam for all eight. What is each method's published choice is the *pretraining* optimizer, which is why TimeMAE-CI pretrains under AdamW and SimTS under SGD, and both learning-rate *schedules*: pretraining uses ExponentialLR for SDTA and TimeDART, CosineAnnealingLR for TimeSiam, SimMTM and CPC, OneCycleLR for PatchTST-SSL, and none for TimeMAE-CI and SimTS, while fine-tuning uses OneCycleLR for SDTA, TimeDART, TimeSiam and PatchTST-SSL and step halving for the other four. Table 4 marks those rows "own". Every such deviation is per method, fixed once and then

Table 4: The shared fixed configuration. Every entry that is neither marked "own" nor footnoted is identical for all eight methods and all twelve datasets; the values are read out of the recorded configurations of the 540 final runs (9 model directories $\times$ 12 datasets $\times$ 5 seeds), not out of the main text. Only each method's own design parameter (Table 6), the rows marked "own" or footnoted (notes a–c), and the seed differ. The deviations that do exist are listed exhaustively in Table 5; none of them is a per-dataset tuning choice.

	pretrain	fine-tune	test
Backbone (transferred; identical for all eight)			
input length L	96	96	96
patch P / stride S	$8/8$	$8/8$	$8/8$
patch tokens N	12	12	12
width d_{model}	32	32	32
feed-forward d_{ff}	64	64	64
attention heads	4	4	4
encoder blocks E	2	2	2
dropout	0.2	0.2	–
head dropout	–	0.1	–
channels	all	all	all
Optimization			
optimizer	own[a]	Adam	–
learning rate	10^{-3}	10^{-4}[c]	–
LR schedule	own[a]	own[a]	–
epochs	50	10	–
early stop	none	patience 3[b]	–
batch size	16	16	16
weight decay	none	none	–
warmup / clip / AMP	none	none	none
checkpoint	best val.	best val.	–

Protocol	
transferred	patch embedding + encoder; head discarded
downstream head	linear flatten head, encoder unfrozen
horizons	$\{96, 192, 336, 720\}$; PEMS $\{12, 24, 48, 96\}$
extended	1080; PEMS 144; exchange excluded
metric scale	standardized, *no* inverse transform
instance norm.	per window, at run time
seeds (acc.)	2021, 2022, 2023, 2024, 2025
seeds (cost)	97

[a]Each method keeps its published optimizer and learning-rate schedule; the base rates are shared, with the one exception in note c.
[b]PatchTST-SSL runs through its own released trainer, which has no early-stopping hook; it keeps the best-validation checkpoint over the same ten epochs. [c]Shared by seven of the eight methods; PatchTST-SSL's released fine-tuning routine halves it to 5×10^{-5}.

Table 5: Every deviation from Table 4 that the recorded run configurations contain. Nothing here is a per-dataset hyperparameter search: the only entry that varies with the dataset is SimMTM's pretraining channel subsampling, a memory-feasibility switch confined to pretraining (see text). "per-method" means the value is fixed once for the method and then held across all twelve datasets.

setting	scope	value as recorded
pretrain optimizer	per-method	Adam for SDTA, TimeDART, TimeSiam, SimMTM, CPC and PatchTST-SSL; **AdamW** for TimeMAE-CI; **SGD** for SimTS (encoder at 10^{-3}; the predictor carries the official SimTS learning-rate multiplier of 10^{-4}, i.e. $10^{-3}\cdot10^{-4}$)
pretrain LR schedule	per-method	ExponentialLR $\gamma = 0.9$/epoch (SDTA, TimeDART); CosineAnnealingLR $T_{\max} = 50$ (TimeSiam, SimMTM, CPC); OneCycleLR $\mathrm{lr}_{\max} = 10^{-3}$ (PatchTST-SSL); constant (TimeMAE-CI, SimTS)
downstream LR schedule	per-method	OneCycleLR $\mathrm{lr}_{\max} = 10^{-4}$, $\mathrm{pct_start} = 0.3$ (SDTA, TimeDART, TimeSiam); step halving $\mathrm{lr} = 10^{-4}\cdot0.5^{\,e-1}$ (TimeMAE-CI, SimMTM, CPC, SimTS); OneCycleLR $\mathrm{lr}_{\max} = 5\times10^{-5}$ (PatchTST-SSL, whose released fine-tuning routine halves the base rate after a zero-epoch freeze phase)
downstream early stop	per-method	patience 3 everywhere except PatchTST-SSL (no hook; best-validation checkpoint over the same 10 epochs)
SimMTM channel keep rate	**per-dataset**	1.0 on ETTh1, ETTh2, ETTm1, ETTm2, exchange and weather; 0.1 on electricity, traffic and PEMS03/04/07/08. *Pretraining only*; fine-tuning and evaluation read every channel
SDTA window chunking	bookkeeping	chunk size 0 or 4 across runs; gradient accumulation over the W windows, an exactly equivalent reformulation kept for memory
dataloader workers, checkpoint period	bookkeeping	1/2/4 workers and a 0/10-epoch checkpoint period; I/O only

held across all twelve datasets; not one of them varies with the dataset, which is the property the no-per-dataset-tuning claim actually rests on. The single per-dataset entry anywhere in the protocol is SimMTM's pretraining channel-subsampling switch, which is a memory-feasibility measure rather than a tuning choice and is discussed in Appendix A.3 below.

Table 6: Per-method design parameters, each at its paper-recommended value and held fixed across all twelve datasets. **The "mask kind" column is not decoration.** SDTA's m is a *past-link mask ratio* over decoder attention edges; the mask numbers of TimeDART, TimeMAE-CI, PatchTST-SSL, SimMTM and TimeSiam are *patch or segment masking ratios* over the input. The two quantities are unrelated and must not be read as the same knob. "run" is the exact directory name under `outputs/test/<model>/<ds>/il96/`, with {s} the seed.

Method	mask kind	design parameter(s)	other pretext constants	run
SDTA (ours)	**past-link**	$m = 0.5$, $s_{\max} = 12$, $W = 4$	$s_{\min} = 1$; $T = 1000$, cosine β, patch space; relative code, 1-layer code MLP; forced $s = 1$; causal; $D = 1$	`arch_scan_Enc_2_ Dec_1_Mask_0p5_ SMax_12_Lmlp_1_W_4_S_{s}`
TimeDART	patch	mask $= 1.0$	$T = 1000$, cosine β; causal autoregressive decoder; $D = 1$	`arch_scan_Enc_2_ Dec_1_S_{s}`
TimeSiam	segment	mask $= 0.25$, lineage tokens $= 2$, $W = 1$	$l_m = 3$; channel-continuous rule; sampling range 6; averaged representation; $D = 1$	`arch_scan_Enc_2_ Dec_1_S_{s}`
W control	segment	$W \in \{2, 4\}$, rest as above	supervision-axis control for TimeSiam; $W = 1$ is the main-table entry	`w_scan_W_{W}_ S_{s}`
PatchTST-SSL	patch	mask_ratio $= 0.4$	RevIN on; linear patch head	`arch_scan_Enc_2_ S_{s}`
SimMTM	point/patch	mask $= 0.5$, positives $= 3$	geometric mask, $l_m = 3$; temperature 0.2; 10% channel subsampling in *pretraining* on the six high-channel datasets (see text)	`arch_scan_Enc_2_ S_{s}`
TimeMAE-CI	patch	mask_ratio $= 0.6$ (7 of 12 patches)	$\alpha = 5.0$, $\beta = 1.0$; EMA momentum 0.99; codebook 192; 1 regressor block; label smoothing 0.2	`arch_scan_Enc_2_ S_{s}`
CPC	none	$K = 1$, negatives $= 64$	negatives decoupled from the batch size; full channels	`arch_scan_Enc_2_ K_1_Neg_64_S_{s}`
SimTS	none	future length $= 96$	non-hierarchical; predictor dropout 0.25; predictor LR multiplier 10^{-4} (official)	`arch_scan_Enc_2_ S_{s}`

Past-link mask versus patch mask. SDTA's $m = 0.5$ is the fraction of *strictly past* query–key links that the cross-attention decoder drops uniformly at random on every forward pass. The mask is resampled per call and the diagonal, each query attending to its own position, is never dropped, so no query is ever left without a key. It removes attention *edges*, not input values: nothing is deleted from the series. A patch masking ratio instead deletes that fraction of the *input patches* before encoding. Reading $m = 0.5$ as "half the input is hidden" is therefore wrong, and comparing it against TimeDART's 1.0 or PatchTST-SSL's 0.4 compares two different quantities.

A.2 Per-method Design Parameters

Each method keeps exactly one design parameter of its own, at the value its paper recommends. Table 6 lists them together with the exact run name each configuration produces, so any cell in this appendix can be traced to the directory that holds it.

The table separates two quantities that share the name "mask" and are otherwise unrelated: SDTA's m is a *past-link mask ratio* over decoder attention edges, whereas the mask ratios of TimeDART, TimeMAE-CI, PatchTST-SSL, SimMTM and TimeSiam are *patch masking ratios* over the input. The footnote to Table 6 gives the distinction in full; the two numbers are not the same knob and must not be compared.

Table 7, referenced from the Introduction and Related Work of the main text, places the eight pretexts by family, by the number of windows that carry distance-controlled targets, and by the candidate scaling axis of each together with its memory growth.

A.3 Two Baseline Adaptations

Two of the seven baselines could not be run unmodified under the shared-encoder protocol. Both changes are recorded here in full, because each one bounds what the corresponding comparison can be read to show.

Table 7: Four pretext families. "Dist.?" counts windows carrying distance-controlled targets; the last column names each pretext's *candidate* scaling axis and its memory growth. Only TimeSiam and SDTA look across windows, and only SDTA makes the view count an axis that stays linear in memory.

Method	Family	Dist.?	Axis / memory
PatchTST-SSL	Masked	no	mask ratio / flat
TimeSiam	Masked	1+1	windows W / linear
SimMTM	Masked	no	views / quadratic
TimeMAE-CI	Masked	no	mask ratio / flat
CPC	Contrastive	no	steps K / linear
SimTS	Predictive	no	depth / linear
TimeDART	Diffusion	no	diff. steps / flat
SDTA (ours)	**Diffusion**	**1+W**	**windows W / linear**

TimeMAE-CI: what was changed, and what was not. TimeMAE-CI is TimeMAE with exactly one modification, and it is not to the objective. The pretext task is unchanged: a random patch permutation splits the $N = 12$ patches into a visible set and a masked set, the online encoder embeds the visible patches, a frozen momentum encoder embeds the masked ones, and a cross-attention regressor predicts the momentum representation of each masked patch from the visible encoding. The loss is the official decoupled form

$$\mathcal{L} \;=\; \alpha\,\mathcal{L}_{\text{align}} \;+\; \beta\,\mathcal{L}_{\text{recon}},$$

with $\mathcal{L}_{\text{align}}$ the mean-squared error between the predicted and the momentum representation and $\mathcal{L}_{\text{recon}}$ the cross-entropy of the predicted codeword against the tokenizer's assignment. We run the released values: $\alpha = 5.0$, $\beta = 1.0$, EMA momentum 0.99, codebook size 192, one regressor block, label smoothing 0.2, and $\text{mask_ratio} = 0.6$, which masks $\lfloor 0.6 \cdot 12 \rfloor = 7$ of the 12 patches. The momentum encoder is updated per optimization step, and the single permutation is drawn once per forward pass and shared across the whole batch, which is TimeMAE's own semantics rather than a per-sequence variant.

The one change is the input projection. Official TimeMAE embeds patches with a channel-mixing $\text{Conv1d}(C \to d_{\text{model}})$ whose input width is the dataset's channel count C. That ties the encoder's first layer to C and makes it impossible to hold the transferred backbone identical across datasets whose C ranges from 7 to 883, which is precisely what our shared-encoder protocol requires. We therefore replace it with the channel-independent $\text{Linear}(P \to d_{\text{model}})$ used by PatchTST, TimeSiam and SimMTM: each channel is folded into the batch dimension, patched at $P = S = 8$, and embedded by one shared projection. The transferred encoder is then byte-for-byte the same $17{,}440$-parameter module every other method transfers, and only the patch embedding and the encoder move downstream. The consequence is that a TimeMAE-CI-versus-others gap reflects the pretext objective and not channel handling or head capacity; the cost is that TimeMAE-CI is not a reproduction of published TimeMAE numbers, and we do not present it as one.

SimMTM: channel subsampling is confined to pretraining. SimMTM's series-wise contrastive term builds an $N \times N$ similarity matrix over $N = (\text{positives} + 1) \cdot B \cdot C$ series, so its memory grows as C^2. On our high-channel datasets full-channel pretraining does not fit a 24 GB card. We therefore enable `select_channels`, the subsampling option SimMTM's own released code provides, keeping a uniformly random 10% of channels per batch. It is switched on for the six high-channel datasets (electricity, traffic, PEMS03/04/07/08) and left at 1.0, that is, off, for ETTh1, ETTh2, ETTm1, ETTm2, exchange and weather; the recorded configuration of every one of SimMTM's 60 final runs carries the corresponding value.

The scope of the switch is the point that matters for the comparison. In the source, `select_channels` is read in exactly one place in SimMTM: the guard `if self.config.select_channels < 1.0 and x.shape[2] > 1` at the top of `PretrainModel.forward` in `model/SimMTM/pretrain.py`. It appears nowhere in `model/SimMTM/forecast.py`, `model/SimMTM/train.py` or `model/SimMTM/test.py`, and nowhere in the shared downstream trainer `model/_baseline_forecast.py`. SimMTM's fine-tuning and its evaluation therefore read every channel of every dataset, exactly like the other seven methods, and **its accuracy**

Table 8: **SDTA capacity grid: what is transferred and what is thrown away.** Only the patch embedding plus the E encoder blocks move downstream; the pretraining decoder, the temporal-distance MLP and the noise-step embedding are discarded at transfer. The consequence is the line to read twice: *Medium and Large deploy the identical* $17{,}440$*-parameter encoder* and differ only in pretraining decoder depth ($D=1$ vs. $D=2$), so *Large is not a bigger encoder*, it is a bigger *pretext*; Small deploys $8{,}896$. The window count W adds *no* parameters at any capacity: the one decoder is applied once per target window, so W costs pretraining compute and memory only. A run is therefore named **Size**_W, with Size $\in$ {Small, Medium, Large} fixed by (E, D) and $W \in \{1, 2, 4, 8\}$ target windows per anchor; the default configuration is Medium_4. All eight compared methods transfer this same backbone at the same (d, d_{ff}, h), so downstream model size is identical across methods and only the pretext head differs.

	transferred	pretext head	total	
Size (E, D)	encoder	(discarded)	pretrain	deployed*
Small $(1, 1)$	8,896	54,080	62,976	45,856
Medium $(2, 1)$	**17,440**	54,080	71,520	**54,400**
Large $(2, 2)$	**17,440**	66,912	84,352	**54,400**

fixed for every count above			
d_{model}	32	patch = stride	8
d_{ff}	64	tokens N	12
heads h	4 (dim. 8)	input length	96

Component	params	kept
patch embedding	288	yes
encoder blocks ($E=1$ / 2)	8,608 / 17,152	yes
cross-attn. decoder ($D=1$ / 2)	12,896 / 25,728	no
distance MLP $\ell(p, s)$	2,112	no
noise-step embedding	2,112	no
flatten recon. head	36,960	no
forecast head ($L=96$)*	36,960	–

*Deployed total = transferred encoder + the linear flatten forecast head at prediction length $L=96$, of size $N \cdot d \cdot L + L = 12 \cdot 32 \cdot 96 + 96 = 36{,}960$; it scales with L and is the same head for all eight methods, so it is not part of what the capacity comparison varies.

numbers are full-channel and directly comparable. What is affected is cost: the peak-VRAM and s/epoch figures we report for SimMTM on the high-channel datasets are measured under 10% sub-sampling and so *understate* its true memory and time. We mark those cells and note the direction of the bias rather than quietly reporting a smaller number; full-channel SimMTM pretraining would need on the order of 48 GB on PEMS07 and does not run on the card the cost table uses.

A.4 Capacity Classes and Parameter Counts

Table 8 gives the three capacity classes and what each actually costs at deployment. The point that is easy to misread from the names is that *Medium and Large deploy the same encoder*. E counts encoder blocks and D pretraining-decoder blocks, and only the encoder transfers, so Medium ($E=2$, $D=1$) and Large ($E=2$, $D=2$) both hand $17{,}440$ parameters to the forecaster and differ only in the depth of a decoder that is discarded before fine-tuning. Large is a bigger *pretext*, not a bigger model. Small ($E=1$, $D=1$) transfers $8{,}896$. Every one of the eight methods transfers a backbone of exactly this size, which is what makes the comparison capacity-matched; what differs is each pretext's own head, and that is compared rather than controlled, because it is what the pretext requires.

A.5 Datasets and Preprocessing

Table 9 lists the twelve datasets with their channel counts, sampling intervals, lengths, split rules and sources. Scores are computed on the globally standardized scale with *no* inverse transform, so they are directly comparable across datasets of different physical units; on top of that standardization each window is instance-normalized at run time, and the statistics are restored at the head.

The one dataset that is not evaluated everywhere is exchange, which is dropped at the extended horizon by every method alike. The reason is arithmetic rather than performance: its validation split

Table 9: The twelve datasets. "rows" are the raw timestamps used after the Informer tail truncation for ETT; train/val/test are the numbers of *target* rows inside each split. Validation and test files additionally carry the last 96 rows of the preceding split as input context, so that every prediction target lies strictly inside its own split. The per-channel z-score is fit on the training segment only and applied to the whole series; the model adds per-window instance normalization at run time; all reported errors are on this standardized scale, with *no* inverse transform.

Group	Dataset	C	sampling	rows	train	val.	test	split rule	horizons	source
4-ETT	ETTh1	7	1 h	17,420	8,640	2,880	2,880	12:4:4 months	$\{96, 192, 336, 720\}$	(Zhou et al., 2021)
	ETTh2	7	1 h	17,420	8,640	2,880	2,880	12:4:4 months	$\{96, 192, 336, 720\}$	(Zhou et al., 2021)
	ETTm1	7	15 min	69,680	34,560	11,520	11,520	12:4:4 months	$\{96, 192, 336, 720\}$	(Zhou et al., 2021)
	ETTm2	7	15 min	69,680	34,560	11,520	11,520	12:4:4 months	$\{96, 192, 336, 720\}$	(Zhou et al., 2021)
4-Wide	exchange	8	1 day	7,588	5,311	758	1,519	$0.7/0.1/0.2$	$\{96, 192, 336, 720\}$	(Lai et al., 2018)
	weather	21	10 min	52,696	36,887	5,269	10,540	$0.7/0.1/0.2$	$\{96, 192, 336, 720\}$	(Wu et al., 2021)
	electricity	321	1 h	26,304	18,412	2,630	5,262	$0.7/0.1/0.2$	$\{96, 192, 336, 720\}$	(Wu et al., 2021)
	traffic	862	1 h	17,544	12,280	1,754	3,510	$0.7/0.1/0.2$	$\{96, 192, 336, 720\}$	(Wu et al., 2021)
4-PEMS	PEMS03	358	5 min	26,208	15,724	5,241	5,243	$0.6/0.2/0.2$	$\{12, 24, 48, 96\}$	(Song et al., 2020)
	PEMS04	307	5 min	16,992	10,195	3,398	3,399	$0.6/0.2/0.2$	$\{12, 24, 48, 96\}$	(Song et al., 2020)
	PEMS07	883	5 min	28,224	16,934	5,644	5,646	$0.6/0.2/0.2$	$\{12, 24, 48, 96\}$	(Song et al., 2020)
	PEMS08	170	5 min	17,856	10,713	3,571	3,572	$0.6/0.2/0.2$	$\{12, 24, 48, 96\}$	(Song et al., 2020)

holds 854 rows, so the longest prediction length it can supply an evaluation window for is about 758, short of the 1080 the stress test requires.

Why exchange is dropped at horizon 1080. The extended-horizon stress test uses a single long horizon, 1080 steps for the eight non-PEMS datasets and 144 for the four PEMS datasets, and reports an eleven-dataset total because exchange cannot support it. The arithmetic is exact and comes from the split files themselves. The stored exchange validation file holds 854 rows: 758 validation target rows plus the 96 rows of training-split context prepended so that the first window has a full length-96 input. A forecasting window consumes $L + H$ consecutive rows, so the number of usable validation windows is

$$854 \; - \; \underbrace{96}_{L} \; - \, H \; + \; 1 \; = \; 759 - H,$$

which is positive only for $H \leq 758$. At $H = 1080$ the count is $759 - 1080 = -321$: there is no valid validation window at all, and the data loader raises rather than silently truncating. Every other dataset clears the bar with room to spare, the next-smallest validation split being traffic's $1{,}754$ rows, so exchange alone is excluded and the extended-horizon aggregate is an eleven-dataset (11-ds) mean; its group is reported as 3-Wide.

A.6 Hardware, Software and Randomness

Which machine produced which number is part of the protocol, so Table 10 states both in full.

Measurement protocol. Seeding is applied at the start of every stage and covers Python, NumPy and PyTorch; the five accuracy seeds 2021–2025 are pinned in every final configuration, and the recorded configuration of each of the 540 final runs carries one of them. Wall-clock cost is a six-epoch run at seed 97, run serially on one GPU with a single dataloader worker and with thread counts pinned to one; the first epoch is discarded and epochs two through six are averaged, so warm-up and CUDA context creation do not enter the number. Peak memory is peak *allocated* bytes reported by `torch.cuda.max_memory_allocated` from a two-epoch pretraining probe, which is enough for the peak to stabilize and which, unlike reserved memory, is independent of the allocator's caching behavior and therefore of the machine.

Table 10: The two machines and what each produced. Accuracy and cost are deliberately separated: the memory-feasibility claim concerns an ordinary 24 GB card, so it is measured on that card rather than extrapolated from a 32 GB one, and the reported epoch times are the slower machine's, which is the conservative direction. Every run is a single-GPU run; the eight cards of the accuracy node buy throughput, not scale, and no reported number depends on multi-GPU training. *n/r*: not recorded at run time, which affects no reported number because the cost machine alone produced the timing-sensitive results. Peak *allocated* memory is a property of the configuration rather than of the machine, so Table 33 draws on both: the CPC and SDTA $W = 1$ rows were measured on the cost machine and the remaining rows on the accuracy node, and the two agree to within 2 MiB where they overlap.

	Accuracy node	**Cost machine**
GPU	8× RTX 5090 (32 GB)	1× RTX 4090 (24 GB)
concurrency	1 run per GPU, single-GPU	1 run, serial
CPU	2× Xeon Gold 6530	Core i7-13700K
RAM	512 GB DDR5 (16×32)	70 GB
system disk	960 GB SSD	*n/r*
data storage	3.84 TB U.2 SSD	*n/r*
OS	Ubuntu 24.04	Ubuntu 22.04.5
Python	*n/r*	3.11.14
PyTorch	*n/r*	2.5.1 (CUDA 12.4)
seeds	2021–2025	97
produced	every MSE / MAE	every s/epoch

B FULL STANDARD-HORIZON RESULTS

Table 1 of the main text reports per-dataset means without their spread, and reports the win pattern only as counts. Table 11 restores the cross-seed standard deviation of every cell, and Table 12 turns the same matrix into per-dataset ranks so that the mean-rank and worst-rank statements can be checked directly.

Three things are visible here that the aggregate hides. First, the seed spread is not uniform across methods: on the four PEMS datasets TimeSiam's cross-seed deviation reaches 0.0204 (PEMS08), twenty times SDTA's twelve-dataset deviation, so part of what separates the methods on those datasets is stability rather than mean accuracy. Second, SDTA's own deviations are among the smallest in every group, which is why its twelve-dataset lead survives the paired tests of Appendix E. Third, the rank matrix makes the breadth claim concrete: SDTA is never worse than fifth on MSE and never worse than fourth on MAE, and it is the only method that never falls into the bottom two on either metric. PatchTST-SSL and SimTS land in the bottom two on ten of twelve datasets by MSE. We report the rank matrix rather than a summary statistic because the comparison it supports, breadth under one fixed configuration, is exactly the kind of claim a single average can flatter.

Table 11: Per-dataset accuracy with cross-seed standard deviation, standard horizons. Each cell is the mean ± sample standard deviation (ddof=1) over the five seeds 2021–2025 of the value averaged across the four canonical prediction lengths ($\{96, 192, 336, 720\}$; PEMS $\{12, 24, 48, 96\}$). Input length = 96. SDTA is Medium_4 ($E = 2$, $D = 1$, $W = 4$); every baseline is at its own matched Medium setting, CPC at its best $K = 1$. Group rows are the standard deviation of the per-seed group mean, not an average of the per-dataset deviations. **Bold** / underline = best / second per row. Rankings use the unrounded means, so a cell may win a row it appears to tie at four decimals (PEMS07 MAE: SDTA 0.36576 against TimeMAE-CI 0.36582). This table is the per-dataset, deviation-carrying companion of the main results table. Group and twelve-dataset rows are printed at the values of Table 1 of the main text. Five of them (4-PEMS MSE for TimeDART and SimMTM, 4-Wide MAE for TimeDART and PatchTST-SSL, 4-PEMS MAE for TimeMAE-CI) are the mean of the rounded per-dataset means rather than the mean of the unrounded ones, which moves them by 0.0001; we keep the main text's value so the two tables agree cell for cell, and no ordering or claim depends on the difference.

Dataset	**SDTA**	TimeDART	TimeSiam	TimeMAE-CI	PatchTST-SSL	SimMTM	CPC	SimTS
Panel A: MSE								
ETTh1	**0.4348$_{\pm.0018}$**	0.4397$_{\pm.0012}$	0.4433$_{\pm.0030}$	0.4509$_{\pm.0081}$	0.4722$_{\pm.0036}$	0.4438$_{\pm.0039}$	0.4619$_{\pm.0017}$	0.4616$_{\pm.0031}$
ETTh2	**0.3744$_{\pm.0014}$**	0.3781$_{\pm.0013}$	0.3815$_{\pm.0022}$	0.3819$_{\pm.0016}$	0.3826$_{\pm.0023}$	0.3764$_{\pm.0012}$	0.3779$_{\pm.0006}$	0.3840$_{\pm.0015}$
ETTm1	0.3854$_{\pm.0018}$	0.3958$_{\pm.0017}$	0.3888$_{\pm.0030}$	0.3876$_{\pm.0033}$	0.3971$_{\pm.0019}$	**0.3841$_{\pm.0028}$**	0.3872$_{\pm.0021}$	0.3984$_{\pm.0028}$
ETTm2	**0.2811$_{\pm.0009}$**	0.2836$_{\pm.0004}$	0.2862$_{\pm.0014}$	0.2849$_{\pm.0023}$	0.2860$_{\pm.0014}$	0.2916$_{\pm.0020}$	0.2854$_{\pm.0012}$	0.2846$_{\pm.0012}$
exchange	0.3623$_{\pm.0027}$	0.3656$_{\pm.0031}$	0.3636$_{\pm.0077}$	0.3592$_{\pm.0033}$	**0.3561$_{\pm.0026}$**	0.3613$_{\pm.0017}$	0.3568$_{\pm.0013}$	0.3641$_{\pm.0019}$
weather	0.2638$_{\pm.0009}$	0.2650$_{\pm.0005}$	0.2655$_{\pm.0007}$	**0.2620$_{\pm.0005}$**	0.2724$_{\pm.0028}$	0.2674$_{\pm.0017}$	0.2683$_{\pm.0011}$	0.2749$_{\pm.0016}$
traffic	**0.5443$_{\pm.0010}$**	0.6026$_{\pm.0127}$	0.5576$_{\pm.0075}$	0.5802$_{\pm.0072}$	0.6459$_{\pm.0050}$	0.5777$_{\pm.0049}$	0.5826$_{\pm.0028}$	0.6413$_{\pm.0069}$
electricity	0.2176$_{\pm.0005}$	0.2245$_{\pm.0019}$	0.2189$_{\pm.0011}$	0.2177$_{\pm.0013}$	0.2305$_{\pm.0017}$	**0.2152$_{\pm.0003}$**	0.2165$_{\pm.0011}$	0.2250$_{\pm.0025}$
PEMS03	0.2691$_{\pm.0029}$	0.3101$_{\pm.0052}$	0.3239$_{\pm.0189}$	**0.2645$_{\pm.0035}$**	0.3268$_{\pm.0017}$	0.3196$_{\pm.0050}$	0.3015$_{\pm.0018}$	0.3710$_{\pm.0038}$
PEMS04	0.3474$_{\pm.0032}$	0.3715$_{\pm.0026}$	0.3736$_{\pm.0122}$	**0.3418$_{\pm.0030}$**	0.4020$_{\pm.0081}$	0.3800$_{\pm.0063}$	0.3791$_{\pm.0030}$	0.4109$_{\pm.0037}$
PEMS07	**0.3019$_{\pm.0013}$**	0.3251$_{\pm.0030}$	0.3369$_{\pm.0087}$	0.3124$_{\pm.0029}$	0.3615$_{\pm.0059}$	0.3455$_{\pm.0089}$	0.3362$_{\pm.0054}$	0.3738$_{\pm.0026}$
PEMS08	0.3517$_{\pm.0053}$	0.3755$_{\pm.0034}$	0.4054$_{\pm.0204}$	**0.3466$_{\pm.0018}$**	0.4148$_{\pm.0111}$	0.4087$_{\pm.0098}$	0.3791$_{\pm.0064}$	0.4236$_{\pm.0055}$
4-ETT	**0.3689$_{\pm.0005}$**	0.3743$_{\pm.0006}$	0.3750$_{\pm.0021}$	0.3763$_{\pm.0015}$	0.3845$_{\pm.0016}$	0.3740$_{\pm.0009}$	0.3781$_{\pm.0013}$	0.3821$_{\pm.0016}$
4-Wide	**0.3470$_{\pm.0009}$**	0.3644$_{\pm.0027}$	0.3514$_{\pm.0030}$	0.3548$_{\pm.0025}$	0.3762$_{\pm.0013}$	0.3554$_{\pm.0017}$	0.3561$_{\pm.0006}$	0.3763$_{\pm.0026}$
4-PEMS	0.3175$_{\pm.0027}$	0.3456$_{\pm.0014}$	0.3599$_{\pm.0116}$	**0.3163$_{\pm.0023}$**	0.3763$_{\pm.0043}$	0.3634$_{\pm.0043}$	0.3490$_{\pm.0037}$	0.3948$_{\pm.0031}$
12-ds	**0.3445$_{\pm.0010}$**	0.3614$_{\pm.0011}$	0.3621$_{\pm.0035}$	0.3491$_{\pm.0016}$	0.3790$_{\pm.0011}$	0.3643$_{\pm.0012}$	0.3610$_{\pm.0010}$	0.3844$_{\pm.0015}$
Panel B: MAE								
ETTh1	**0.4340$_{\pm.0011}$**	0.4368$_{\pm.0010}$	0.4373$_{\pm.0014}$	0.4392$_{\pm.0012}$	0.4467$_{\pm.0019}$	0.4376$_{\pm.0013}$	0.4386$_{\pm.0006}$	0.4434$_{\pm.0028}$
ETTh2	**0.3994$_{\pm.0009}$**	0.4009$_{\pm.0008}$	0.4077$_{\pm.0019}$	0.4068$_{\pm.0026}$	0.4045$_{\pm.0012}$	0.4030$_{\pm.0007}$	0.4010$_{\pm.0003}$	0.4047$_{\pm.0011}$
ETTm1	0.3979$_{\pm.0008}$	0.4016$_{\pm.0007}$	0.3984$_{\pm.0021}$	0.3988$_{\pm.0019}$	0.4020$_{\pm.0009}$	0.3973$_{\pm.0012}$	**0.3968$_{\pm.0011}$**	0.3984$_{\pm.0015}$
ETTm2	**0.3283$_{\pm.0009}$**	0.3307$_{\pm.0005}$	0.3337$_{\pm.0007}$	0.3336$_{\pm.0021}$	0.3317$_{\pm.0006}$	0.3383$_{\pm.0009}$	0.3322$_{\pm.0012}$	0.3303$_{\pm.0013}$
exchange	0.4037$_{\pm.0015}$	0.4060$_{\pm.0018}$	0.4057$_{\pm.0036}$	0.4026$_{\pm.0024}$	0.4018$_{\pm.0012}$	0.4051$_{\pm.0013}$	**0.4006$_{\pm.0008}$**	0.4072$_{\pm.0014}$
weather	0.2838$_{\pm.0007}$	0.2844$_{\pm.0002}$	0.2860$_{\pm.0007}$	**0.2822$_{\pm.0002}$**	0.2899$_{\pm.0019}$	0.2878$_{\pm.0007}$	0.2874$_{\pm.0008}$	0.2927$_{\pm.0013}$
traffic	**0.3507$_{\pm.0011}$**	0.3856$_{\pm.0046}$	0.3648$_{\pm.0043}$	0.3731$_{\pm.0037}$	0.4170$_{\pm.0019}$	0.3711$_{\pm.0026}$	0.3841$_{\pm.0031}$	0.3952$_{\pm.0039}$
electricity	0.3032$_{\pm.0006}$	0.3134$_{\pm.0030}$	0.3072$_{\pm.0018}$	0.3004$_{\pm.0017}$	0.3155$_{\pm.0027}$	**0.2949$_{\pm.0003}$**	0.2965$_{\pm.0014}$	0.3067$_{\pm.0032}$
PEMS03	0.3488$_{\pm.0021}$	0.3769$_{\pm.0033}$	0.3898$_{\pm.0157}$	**0.3405$_{\pm.0021}$**	0.3807$_{\pm.0013}$	0.3774$_{\pm.0026}$	0.3685$_{\pm.0018}$	0.4033$_{\pm.0017}$
PEMS04	0.3995$_{\pm.0028}$	0.4175$_{\pm.0030}$	0.4170$_{\pm.0094}$	**0.3901$_{\pm.0024}$**	0.4275$_{\pm.0043}$	0.4169$_{\pm.0046}$	0.4153$_{\pm.0017}$	0.4307$_{\pm.0013}$
PEMS07	**0.3658$_{\pm.0014}$**	0.3829$_{\pm.0035}$	0.3925$_{\pm.0041}$	0.3658$_{\pm.0017}$	0.3998$_{\pm.0036}$	0.3952$_{\pm.0094}$	0.3844$_{\pm.0036}$	0.4049$_{\pm.0011}$
PEMS08	0.3855$_{\pm.0041}$	0.4034$_{\pm.0029}$	0.4222$_{\pm.0113}$	**0.3766$_{\pm.0007}$**	0.4211$_{\pm.0066}$	0.4184$_{\pm.0084}$	0.3990$_{\pm.0037}$	0.4208$_{\pm.0018}$
4-ETT	**0.3899$_{\pm.0004}$**	0.3925$_{\pm.0002}$	0.3943$_{\pm.0013}$	0.3946$_{\pm.0009}$	0.3962$_{\pm.0009}$	0.3941$_{\pm.0006}$	0.3922$_{\pm.0007}$	0.3942$_{\pm.0012}$
4-Wide	**0.3354$_{\pm.0006}$**	0.3473$_{\pm.0013}$	0.3409$_{\pm.0016}$	0.3396$_{\pm.0017}$	0.3560$_{\pm.0008}$	0.3397$_{\pm.0009}$	0.3421$_{\pm.0007}$	0.3504$_{\pm.0019}$
4-PEMS	0.3749$_{\pm.0022}$	0.3952$_{\pm.0017}$	0.4054$_{\pm.0071}$	**0.3683$_{\pm.0014}$**	0.4073$_{\pm.0023}$	0.4020$_{\pm.0050}$	0.3918$_{\pm.0021}$	0.4149$_{\pm.0011}$
12-ds	**0.3667$_{\pm.0008}$**	0.3783$_{\pm.0007}$	0.3802$_{\pm.0021}$	0.3675$_{\pm.0011}$	0.3865$_{\pm.0005}$	0.3786$_{\pm.0015}$	0.3754$_{\pm.0006}$	0.3865$_{\pm.0011}$

Table 12: Per-dataset rank matrices at the standard horizons (rank 1 = best, i.e. lowest error, among the eight methods; ranks are taken on the five-seed mean of the four-horizon average). **Mean** is the rank averaged over the twelve datasets, **Worst** the largest rank any dataset assigns to the method, and **Bot.2** the number of datasets out of twelve on which the method places seventh or eighth. SDTA is the only method that is never worse than fifth on MSE or fourth on MAE, and it never enters the bottom two on either metric. **Bold** = best per column, and every rank-1 entry is bold. Ranks are computed on the unrounded means. Dataset abbreviations: Exch. = exchange, Weat. = weather, Elec. = electricity, Traf. = traffic, P03–P08 = PEMS03–PEMS08.

Method	ETTh1	ETTh2	ETTm1	ETTm2	Exch.	Weat.	Traf.	Elec.	P03	P04	P07	P08	Mean	Worst	Bot.2
Panel A: MSE ranks															
SDTA	**1**	**1**	2	**1**	5	2	**1**	3	2	2	**1**	2	**1.92**	**5**	**0/12**
TimeDART	2	4	6	2	8	3	6	6	4	3	3	3	4.17	8	1/12
TimeSiam	3	5	5	7	6	4	2	5	6	4	5	5	4.75	7	1/12
TimeMAE-CI	5	6	4	4	3	**1**	4	4	**1**	**1**	2	**1**	3.00	6	**0/12**
PatchTST-SSL	8	7	7	6	**1**	7	8	8	7	7	7	7	6.67	8	10/12
SimMTM	4	2	**1**	8	4	5	3	**1**	5	6	6	6	4.25	8	1/12
CPC	7	3	3	5	2	6	5	2	3	5	4	4	4.08	7	1/12
SimTS	6	8	8	3	7	8	7	7	8	8	8	8	7.17	8	10/12
Panel B: MAE ranks															
SDTA	**1**	**1**	3	**1**	4	2	**1**	4	2	2	**1**	2	**2.00**	**4**	**0/12**
TimeDART	2	2	7	3	7	3	6	7	4	6	3	4	4.50	7	3/12
TimeSiam	3	8	4	7	6	4	2	6	7	5	5	8	5.42	8	4/12
TimeMAE-CI	6	7	6	6	3	**1**	4	3	**1**	**1**	2	**1**	3.42	7	1/12
PatchTST-SSL	8	5	8	4	2	7	8	8	6	7	7	7	6.42	8	8/12
SimMTM	4	4	2	8	5	6	3	**1**	5	4	6	5	4.42	8	1/12
CPC	5	3	**1**	5	**1**	5	5	2	3	3	4	3	3.33	5	**0/12**
SimTS	7	6	5	2	8	8	7	5	8	8	8	6	6.50	8	7/12

Table 13: Scaling each model's own supervision axis (W for SDTA and TimeSiam, K for CPC): group-level MSE, 5-seed average, capacity Medium. **Bold** = best per model per group. Only SDTA improves as the axis grows.

Model	Group	1	2	4	$\Delta_{1\to4}$
SDTA	4-ETT	0.3743	0.3704	**0.3689**	-0.0054
	4-Wide	0.3588	0.3495	**0.3470**	-0.0118
	4-PEMS	0.3448	0.3220	**0.3175**	-0.0273
TimeSiam	4-ETT	**0.3750**	0.3767	0.3762	$+0.0012$
	4-Wide	0.3514	**0.3513**	0.3516	$+0.0002$
	4-PEMS	**0.3599**	0.3642	0.3662	$+0.0063$
CPC	4-ETT	**0.3781**	0.3785	0.3796	$+0.0016$
	4-Wide	**0.3561**	0.3569	0.3561	0.0000
	4-PEMS	**0.3490**	0.3492	0.3557	$+0.0068$

C Scaling the Supervision Axis

Figure 2 of the main text plots twelve aggregate cells per horizon regime. Table 14 prints them, with the cross-seed deviation that the error bars encode and with MAE alongside MSE. Table 15 then decomposes the Medium row by dataset group, which is where the size of the effect varies most. Table 13, referenced from Section 5.3 of the main text and printed here, sets the Medium-capacity group means beside the two baseline axes.

The saturation point is capacity dependent, and this appendix is where the two fine points behind the main text's summary become checkable, because the cells are printed rather than plotted. The first is the Small exception the caption of Figure 2 states: at Small on the extended horizon the MSE minimum is $W = 2$ (0.6097) against $W = 4$ (0.6107), a 0.0010 gap set beside cross-seed deviations of 0.0024 and 0.0012. Small's MAE column at the same horizon does fall through $W = 4$ ($0.5262/0.5173/0.5171$), and every Medium and Large column falls through $W = 4$ on both metrics and both horizons, so the shallowest encoder is the only one with no headroom past two windows at that horizon, and the default capacity is unaffected. The second: this is the one place where the main text's default is not the best cell in the grid. At Small and Medium the axis is flat past $W = 4$ on both horizons. At Large the eighth window still helps: it gives the lowest cell of the entire grid on three of the four aggregate columns (0.3442 standard MSE, 0.5705 extended MSE, 0.5015 extended MAE) and is second only to Medium_4 on standard MAE (0.3668 against 0.3667), and at the extended horizon the $W = 4 \to 8$ step improves in all five paired seeds with a mean of -0.0020. We nevertheless keep Medium_4 as the default because that margin sits inside the combined unpaired cross-seed spread of the cells being compared, and because $W = 8$ would double pretraining time for it, extrapolating the near-linear W scaling of Table 34. The honest statement is that a deeper pretraining decoder has headroom past four windows which a stretched horizon exposes, not that eight windows are useless.

C.1 The Shift Bound $s_{\max}$

The shift bound $s_{\max}$ sets how far ahead a target window may be drawn. We fixed it at 12 patches, exactly one input window, and the main text cites one sweep in support. Table 16 is that sweep.

It is a smaller experiment than the rest of this appendix and we mark it as such: three seeds rather than five, and the six lower-channel datasets rather than all twelve, because the grid is 3×3 cells on top of the main protocol. It must not be read against the five-seed numbers elsewhere. Within it the conclusion is unambiguous: doubling the reachable distance moves the matched-cell average MSE by 0.0001 and no per-W group average by more than 0.0008, both far below the 0.0148 that the window count itself buys. Distance *range* is not the active ingredient; distance *count* is. And $s_{\max} = 12$ is the cheapest setting to justify, since it needs no data beyond one window ahead of the anchor.

Table 14: Per-cell values behind Figure 2 of the main text: SDTA aggregate accuracy for three encoder/decoder capacities × four window counts W × two horizon regimes. Each cell is the mean ± sample standard deviation (ddof=1) over the five pretraining seeds 2021–2025; within a seed the score is first averaged over the four standard prediction lengths of a dataset ($\{12, 24, 48, 96\}$ for PEMS, $\{96, 192, 336, 720\}$ otherwise) and then averaged unweighted over datasets. The extended horizon uses the single super-long prediction length (1080; 144 for PEMS) and drops exchange, which is too short to supply the required evaluation windows at this horizon (its 854-row validation split admits at most a prediction length of 758), leaving 11 datasets. Every cell is backed by the complete grid of runs: $12 \times 5 = 60$ (standard) and $11 \times 5 = 55$ (extended). **Bold** marks the best value in a column and underline the second best. The Medium, W=4 row is the default SDTA configuration reported in the main tables.

		Standard horizons (12 datasets)		Extended horizon (11 datasets)	
Capacity	W	MSE	MAE	MSE	MAE
Small E=1, D=1	1	0.3649 ±0.0006	0.3784 ±0.0006	0.6242 ±0.0016	0.5262 ±0.0010
	2	0.3579 ±0.0007	0.3726 ±0.0005	0.6097 ±0.0024	0.5173 ±0.0013
	4	0.3564 ±0.0005	0.3714 ±0.0003	0.6107 ±0.0012	0.5171 ±0.0008
	8	0.3568 ±0.0007	0.3716 ±0.0006	0.6119 ±0.0014	0.5170 ±0.0009
Medium E=2, D=1	1	0.3593 ±0.0006	0.3772 ±0.0006	0.6064 ±0.0024	0.5203 ±0.0012
	2	0.3473 ±0.0006	0.3687 ±0.0006	0.5760 ±0.0015	0.5043 ±0.0011
	4	0.3445 ±0.0010	**0.3667** ±0.0008	0.5723 ±0.0023	0.5021 ±0.0010
	8	0.3448 ±0.0009	0.3671 ±0.0005	0.5721 ±0.0024	0.5022 ±0.0010
Large E=2, D=2	1	0.3588 ±0.0016	0.3768 ±0.0014	0.6069 ±0.0055	0.5211 ±0.0026
	2	0.3487 ±0.0012	0.3697 ±0.0009	0.5788 ±0.0023	0.5054 ±0.0014
	4	0.3450 ±0.0010	0.3672 ±0.0009	0.5724 ±0.0025	0.5024 ±0.0014
	8	**0.3442** ±0.0005	0.3668 ±0.0005	**0.5705** ±0.0016	**0.5015** ±0.0008

Table 15: The supervision axis resolved by dataset group, at capacity Medium ($E = 2$, $D = 1$) and the standard horizons. Each cell is the mean ± sample standard deviation (ddof=1) over the five seeds 2021–2025 of the group mean, where a dataset is first averaged over its four standard prediction lengths. **Bold** marks the best W in each column. The last two rows give the $W = 1 \to 4$ change in absolute and relative terms. Every group improves in MSE as W grows to four and then saturates, but the size of the gain is not uniform: $W = 1 \to 4$ buys -0.0054 MSE on 4-ETT, -0.0118 on 4-Wide and -0.0273 on the high-channel 4-PEMS group, a 1.4%, 3.3% and 7.9% relative reduction, and MAE preserves that ordering. The axis therefore pays most where the data are widest, which is also where a single window supplies the least supervision per encoded anchor. One cell departs from the pattern, and printing the numbers rather than plotting them is what makes it visible: 4-ETT MAE is lowest at $W = 2$, 0.3897 against 0.3899 at $W = 4$, a 0.0002 gap against cross-seed deviations of 0.0003 and 0.0004. The monotonicity the main text claims is on MSE, where 4-ETT does fall through $W = 4$. The Δ and rel. rows are computed on the unrounded group means, so 4-Wide MAE reads -0.0077 where subtracting the two printed cells gives -0.0076. Aggregate per-capacity values are in Table 14, and the same three groups appear in Table 13 beside the two baseline axes, with one last-digit difference: the 4-ETT cell at $W = 2$ reads 0.3703 here, computed from the unrounded per-dataset means, and 0.3704 in Table 13, which averages the rounded ones. No claim depends on it.

	4-ETT		4-Wide		4-PEMS	
W	MSE	MAE	MSE	MAE	MSE	MAE
1	0.3743 ±0.0010	0.3927 ±0.0006	0.3588 ±0.0016	0.3430 ±0.0010	0.3448 ±0.0021	0.3959 ±0.0023
2	0.3703 ±0.0003	**0.3897** ±0.0003	0.3495 ±0.0007	0.3380 ±0.0004	0.3220 ±0.0019	0.3784 ±0.0020
4	**0.3689** ±0.0005	0.3899 ±0.0004	**0.3470** ±0.0009	**0.3354** ±0.0006	**0.3175** ±0.0027	**0.3749** ±0.0022
8	0.3690 ±0.0008	0.3906 ±0.0001	0.3473 ±0.0012	0.3355 ±0.0008	0.3179 ±0.0021	0.3754 ±0.0015
$\Delta_{1\to4}$	-0.0054	-0.0028	-0.0118	-0.0077	-0.0273	-0.0210
rel.	-1.4%	-0.7%	-3.3%	-2.2%	-7.9%	-5.3%

Table 16: Sensitivity of SDTA to the shift bound $s_{\max}$, crossed with the number of target windows W. Six lower-channel datasets (ETTh1, ETTh2, ETTm1, ETTm2, exchange, weather), input length 96, forecast horizons $\{96, 192, 336, 720\}$ averaged, capacity $E{=}2$, $D{=}1$ (Medium) with a single lineage MLP layer and mask ratio 0.5, and *three* seeds (2021–2023). **This is a different protocol from the main results, which use five seeds (2021–2025) over twelve datasets; the two must not be mixed or compared cell by cell.** The weather dataset has no $W{=}8$ runs at $s_{\max} \in \{18, 24\}$, so the (weather, $W{=}8$) cell (marked †) is excluded, leaving 17 of 18 (dataset, W) cells matched across all three $s_{\max}$ values; every matched-cell and per-W average is taken over those matched cells only. Across $s_{\max}$ the overall matched-cell MSE moves by 0.0001 (0.3555/0.3555/0.3556) and the matched-cell MAE by 0.0001 (0.3798/0.3798/0.3797), and **no per-W group average moves by more than** 0.0008: 0.0002 at $W{=}2$, 0.0005 at $W{=}4$ and 0.0008 at $W{=}8$. These are the two figures the main text quotes. The largest displacement of any single cell is 0.0037 MSE (ETTm1, $W{=}8$), an order of magnitude above the aggregate and confined to one cell. All ranges are computed on the unrounded values, so they can differ in the last digit from subtracting the four-decimal numbers printed here. SDTA is therefore insensitive to $s_{\max}$ over this range. No cell is highlighted, deliberately: a marked winner would read as a ranking, and there is none to report. Counting which $s_{\max}$ attains the lowest value in each of the seventeen matched cells gives 6/5/6 for $s_{\max} = 12/18/24$ on MSE and 5/7/5 on MAE, close to the 5.7 per setting that chance alone would produce. No setting is uniformly preferred, and $s_{\max} = 12$ is kept because it is the smallest, needing no data beyond one window ahead of the anchor.

		$s_{\max} = 12$		$s_{\max} = 18$		$s_{\max} = 24$	
Dataset	W	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	2	0.4348	0.4326	0.4355	0.4330	0.4344	0.4328
	4	0.4353	0.4344	0.4345	0.4341	0.4341	0.4339
	8	0.4365	0.4348	0.4366	0.4357	0.4386	0.4365
ETTh2	2	0.3766	0.3997	0.3776	0.4000	0.3778	0.4000
	4	0.3746	0.3997	0.3741	0.3990	0.3748	0.3993
	8	0.3748	0.4007	0.3746	0.4006	0.3744	0.4000
ETTm1	2	0.3891	0.3983	0.3891	0.3980	0.3892	0.3972
	4	0.3853	0.3978	0.3875	0.3998	0.3862	0.3979
	8	0.3845	0.3987	0.3860	0.3983	0.3883	0.4003
ETTm2	2	0.2806	0.3273	0.2808	0.3272	0.2807	0.3276
	4	0.2816	0.3287	0.2805	0.3275	0.2819	0.3291
	8	0.2815	0.3280	0.2827	0.3297	0.2821	0.3284
exchange	2	0.3582	0.4015	0.3581	0.4014	0.3575	0.4009
	4	0.3620	0.4035	0.3596	0.4021	0.3594	0.4022
	8	0.3610	0.4032	0.3593	0.4025	0.3590	0.4024
weather	2	0.2640	0.2837	0.2636	0.2837	0.2637	0.2838
	4	0.2638	0.2839	0.2631	0.2833	0.2632	0.2835
	8†	0.2628	0.2830	–	–	–	–
Matched avg., $W{=}2$ (6 ds)		0.3506	0.3739	0.3508	0.3739	0.3505	0.3737
Matched avg., $W{=}4$ (6 ds)		0.3504	0.3747	0.3499	0.3743	0.3499	0.3743
Matched avg., $W{=}8$ (5 ds)		0.3677	0.3931	0.3678	0.3934	0.3685	0.3935
Matched avg., all 17 cells		0.3555	0.3798	0.3555	0.3798	0.3556	0.3797

Table 17: Extended prediction length, group and aggregate scores. Input length $= 96$, prediction length $= \{1080\}$ (PEMS $\{144\}$), 5-seed average, **SDTA capacity: Medium_4**. The last two columns are each method's mean and worst per-dataset MSE rank. The exchange series is too short at 1080 steps and is dropped for every method, so the Wide group holds three datasets (*3-Wide*) and the aggregate covers eleven. **Bold** / underline = best / 2nd. Per-dataset MSE and MAE are in Appendix D.

	Group MSE			11-ds	rank	
Model	4-ETT	3-Wide	4-PEMS	MSE	mean	worst
SDTA	0.4911	**0.4280**	0.7618	**0.5723**	**2.2**	**3**
TimeDART	**0.4909**	0.4505	0.8427	0.6078	3.6	7
TimeSiam	0.4949	0.4330	0.8682	0.6138	4.5	7
TimeMAE-CI	0.5047	0.4398	**0.7551**	0.5780	3.8	8
PatchTST-SSL	0.5034	0.4667	0.8828	0.6314	6.1	8
SimMTM	0.4995	0.4396	0.8805	0.6217	4.6	8
CPC	0.4980	0.4407	0.8411	0.6072	4.2	6
SimTS	0.5032	0.4629	1.0539	0.6925	6.9	8

D Extended Prediction Length

Table 17, referenced from Section 5.5 of the main text and printed here, reports the extended horizon at group level and states two claims that need per-dataset evidence: that SDTA's worst per-dataset rank is third, and that its MAE is second behind TimeMAE-CI. Table 18 gives the underlying 8×11 matrix on both metrics, Table 19 its cross-seed deviations, and Table 20 the rank matrices.

The aggregate MAE gap is the one comparison in the main text we do not win, and it is small: SDTA 0.5021 against TimeMAE-CI 0.5016, a difference of 0.0005 against a cross-seed deviation of 0.0010, and Appendix E confirms it is not significant in either direction. The per-dataset view explains where it comes from. TimeMAE-CI wins the four PEMS datasets on MAE by a clear margin (0.6182 against 0.6320 at group level), and SDTA recovers most but not all of it on the wide group (0.3711 against 0.3777). The composition of the lead therefore shifts with the horizon: at the standard horizons SDTA leads on both metrics, and at 1080 steps the PEMS weakness grows enough to cost it the MAE aggregate while leaving the MSE aggregate intact.

What does not shift is consistency. No method other than SDTA keeps its worst per-dataset MSE rank below sixth, and four of the seven baselines are last on at least one dataset.

Table 18: Extended-horizon per-dataset accuracy (pred_len 1080; PEMS 144), mean over seeds 2021–2025. Input length 96 for every method. `exchange` is excluded: its validation split cannot supply a window at 1080, so the wide group holds three datasets (*). **Bold** is best, underline is second best, within each row and metric. Marks follow the full-precision values: in the 4-ETT MAE row SDTA (0.47055) and CPC (0.47063) coincide once rounded to four decimals. This is the per-dataset expansion of the grouped extended-horizon table (Table 17). Three group cells (4-ETT MSE for TimeMAE-CI, 4-PEMS MSE for TimeDART, 3-Wide MSE for SimMTM) are printed at the values of Table 17, which rounds the per-dataset means before averaging; a fresh unrounded aggregate moves each by 0.0001.

Dataset	SDTA		TimeMAE-CI		TimeDART		TimeSiam		SimMTM		PatchTST-SSL		CPC		SimTS	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.5582	0.5122	0.5872	0.5230	**0.5475**	0.5128	0.5588	0.5125	0.5730	0.5224	0.5837	0.5171	0.5743	**0.5092**	0.5796	0.5222
ETTh2	0.4630	0.4745	0.4746	0.4826	0.4627	0.4730	0.4691	0.4802	0.4675	0.4794	0.4675	0.4766	**0.4624**	**0.4727**	0.4681	0.4755
ETTm1	0.4858	0.4571	0.4935	0.4610	0.4962	0.4599	0.4900	0.4573	**0.4830**	**0.4558**	0.5062	0.4642	0.4927	0.4586	0.5053	0.4602
ETTm2	0.4574	0.4384	0.4633	0.4454	0.4573	0.4373	0.4617	0.4432	0.4744	0.4492	**0.4564**	0.4368	0.4628	0.4421	0.4598	**0.4359**
weather	0.3955	0.3757	**0.3948**	0.3754	0.3950	**0.3746**	0.3965	0.3769	0.3971	0.3770	0.4025	0.3794	0.3993	0.3789	0.4026	0.3799
traffic	**0.5941**	**0.3760**	0.6297	0.3970	0.6540	0.4094	0.6065	0.3879	0.6295	0.3965	0.6930	0.4384	0.6303	0.4071	0.6839	0.4166
electricity	0.2944	0.3617	0.2949	0.3605	0.3024	0.3714	0.2959	0.3652	**0.2921**	**0.3555**	0.3047	0.3719	0.2926	0.3560	0.3023	0.3661
PEMS03	0.5996	0.5711	**0.5828**	**0.5530**	0.7093	0.6247	0.7297	0.6412	0.7331	0.6299	0.7060	0.6126	0.6655	0.5926	0.9728	0.7307
PEMS04	0.7976	0.6704	**0.7716**	**0.6463**	0.8607	0.7045	0.8705	0.7019	0.8671	0.6970	0.9070	0.7089	0.8837	0.7005	1.0308	0.7634
PEMS07	**0.7251**	**0.6265**	0.7558	0.6313	0.7958	0.6618	0.8108	0.6713	0.8356	0.6743	0.8489	0.6736	0.8085	0.6555	0.9697	0.7270
PEMS08	0.9250	0.6599	**0.9101**	**0.6421**	1.0048	0.7013	1.0619	0.7249	1.0862	0.7217	1.0692	0.7077	1.0066	0.6828	1.2422	0.7774
4-ETT	0.4911	**0.4706**	0.5047	0.4780	**0.4909**	0.4708	0.4949	0.4733	0.4995	0.4767	0.5034	0.4737	0.4980	0.4706	0.5032	0.4734
3-Wide*	**0.4280**	**0.3711**	0.4398	0.3777	0.4505	0.3851	0.4330	0.3767	0.4396	0.3763	0.4667	0.3966	0.4407	0.3807	0.4629	0.3875
4-PEMS	0.7618	0.6320	**0.7551**	**0.6182**	0.8427	0.6731	0.8682	0.6848	0.8805	0.6807	0.8828	0.6757	0.8411	0.6578	1.0539	0.7496
11-ds avg.	**0.5723**	0.5021	0.5780	**0.5016**	0.6078	0.5210	0.6138	0.5239	0.6217	0.5235	0.6314	0.5261	0.6072	0.5142	0.6925	0.5504

Table 19: Cross-seed sample standard deviation (five seeds, $n-1$ denominator) of the extended-horizon cells in Table 18. Seed noise is one to two orders of magnitude below the between-method gaps on PEMS and traffic, which is where the ordering of the methods is decided.

Dataset	SDTA		TimeMAE-CI		TimeDART		TimeSiam		SimMTM		PatchTST-SSL		CPC		SimTS	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.0078	0.0036	0.0225	0.0084	0.0052	0.0030	0.0036	0.0015	0.0047	0.0039	0.0042	0.0024	0.0047	0.0024	0.0074	0.0044
ETTh2	0.0029	0.0017	0.0073	0.0066	0.0033	0.0020	0.0023	0.0009	0.0018	0.0017	0.0025	0.0016	0.0028	0.0015	0.0063	0.0035
ETTm1	0.0012	0.0008	0.0055	0.0025	0.0010	0.0010	0.0049	0.0016	0.0025	0.0032	0.0051	0.0022	0.0032	0.0019	0.0033	0.0016
ETTm2	0.0024	0.0018	0.0047	0.0022	0.0009	0.0013	0.0026	0.0016	0.0069	0.0036	0.0017	0.0007	0.0037	0.0028	0.0014	0.0009
weather	0.0014	0.0006	0.0007	0.0004	0.0003	0.0004	0.0008	0.0002	0.0020	0.0010	0.0018	0.0008	0.0011	0.0010	0.0016	0.0009
traffic	0.0012	0.0014	0.0087	0.0046	0.0066	0.0031	0.0075	0.0040	0.0047	0.0029	0.0042	0.0015	0.0024	0.0031	0.0058	0.0027
electricity	0.0004	0.0005	0.0013	0.0017	0.0022	0.0031	0.0007	0.0013	0.0005	0.0005	0.0018	0.0025	0.0008	0.0008	0.0026	0.0031
PEMS03	0.0081	0.0048	0.0080	0.0043	0.0164	0.0100	0.0389	0.0212	0.0246	0.0117	0.0133	0.0054	0.0023	0.0016	0.0249	0.0119
PEMS04	0.0049	0.0033	0.0101	0.0046	0.0130	0.0064	0.0365	0.0177	0.0188	0.0090	0.0162	0.0065	0.0203	0.0098	0.0224	0.0077
PEMS07	0.0023	0.0014	0.0054	0.0022	0.0096	0.0047	0.0327	0.0180	0.0044	0.0038	0.0151	0.0062	0.0068	0.0032	0.0135	0.0047
PEMS08	0.0129	0.0053	0.0083	0.0043	0.0110	0.0046	0.0505	0.0191	0.0280	0.0123	0.0265	0.0095	0.0177	0.0067	0.0345	0.0124
Mean std	0.0041	0.0023	0.0075	0.0038	0.0063	0.0036	0.0164	0.0079	0.0090	0.0049	0.0084	0.0036	0.0060	0.0031	0.0113	0.0049

Table 20: Per-dataset rank of the eight methods at the extended horizon (pred_len 1080; PEMS 144), 1 = lowest error, over the eleven datasets that admit a window at that horizon. Ranks are taken on the five-seed mean of Table 18. **Bold** marks a rank of 1. In panel (a), SDTA is the only method whose worst rank stays inside the top three; every other method drops to rank 6 or lower on at least one dataset. That is what the grouped averages in the main text compress away: a method can hold a competitive group mean by winning a few datasets by a large margin while losing most of the rest. Ranks are computed on the unrounded means, so ETTh2 on MSE, where PatchTST-SSL and SimMTM both print 0.4675, is separated at full precision. Dataset abbreviations: Weat. = weather, Traf. = traffic, Elec. = electricity.

(a) Rank by MSE

Method	ETTh1	ETTh2	ETTm1	ETTm2	Weat.	Traf.	Elec.	PEMS03	PEMS04	PEMS07	PEMS08	Mean	Worst
SDTA	2	3	2	3	3	**1**	3	2	2	**1**	2	2.2	3
TimeMAE-CI	8	8	5	7	**1**	4	4	**1**	**1**	2	**1**	3.8	8
TimeDART	**1**	2	6	2	2	6	7	5	3	3	3	3.6	7
TimeSiam	3	7	3	5	4	2	5	6	5	5	5	4.5	7
SimMTM	4	5	**1**	8	5	3	**1**	7	4	6	7	4.6	8
PatchTST-SSL	7	4	8	**1**	7	8	8	4	7	7	6	6.1	8
CPC	5	**1**	4	6	6	5	2	3	6	4	4	4.2	6
SimTS	6	6	7	4	8	7	6	8	8	8	8	6.9	8

(b) Rank by MAE

Method	ETTh1	ETTh2	ETTm1	ETTm2	Weat.	Traf.	Elec.	PEMS03	PEMS04	PEMS07	PEMS08	Mean	Worst
SDTA	2	3	2	4	3	**1**	4	2	2	**1**	2	2.4	4
TimeMAE-CI	8	8	7	7	2	4	3	**1**	**1**	2	**1**	4.0	8
TimeDART	4	2	5	3	**1**	6	7	5	6	4	4	4.3	7
TimeSiam	3	7	3	6	4	2	5	7	5	5	7	4.9	7
SimMTM	7	6	**1**	8	5	3	**1**	6	3	7	6	4.8	8
PatchTST-SSL	5	5	8	2	7	8	8	4	7	6	5	5.9	8
CPC	**1**	**1**	4	5	6	5	2	3	4	3	3	3.4	6
SimTS	6	4	6	**1**	8	7	6	8	8	8	8	6.4	8

Table 21: Paired input matrix for the standard-horizon tests. One observation per (method, dataset): the five-seed mean (seeds 2021–2025) of the score averaged over the four canonical prediction lengths, $\{96, 192, 336, 720\}$ and $\{12, 24, 48, 96\}$ for PEMS. These 8×12 numbers are exactly the paired vectors fed to the Friedman and Wilcoxon tests of Table 24, and they reproduce the main results table of the main text (Table 1). Best per column in **bold**, second underlined. Dataset abbreviations: Exch. = exchange, Weat. = weather, Traf. = traffic, Elec. = electricity.

Method	ETTh1	ETTh2	ETTm1	ETTm2	Exch.	Weat.	Traf.	Elec.	PEMS03	PEMS04	PEMS07	PEMS08	**Avg**
MSE													
$SDTA_4$	**0.4348**	**0.3744**	0.3854	**0.2811**	0.3623	0.2638	**0.5443**	0.2176	0.2691	0.3474	**0.3019**	0.3517	**0.3445**
TimeMAE-CI	0.4509	0.3819	0.3876	0.2849	0.3592	**0.2620**	0.5802	0.2177	**0.2645**	**0.3418**	0.3124	**0.3466**	0.3491
CPC	0.4619	0.3779	0.3872	0.2854	0.3568	0.2683	0.5826	0.2165	0.3015	0.3791	0.3362	0.3791	0.3610
TimeDART	0.4397	0.3781	0.3958	0.2836	0.3656	0.2650	0.6026	0.2245	0.3101	0.3715	0.3251	0.3755	0.3614
TimeSiam	0.4433	0.3815	0.3888	0.2862	0.3636	0.2655	0.5576	0.2189	0.3239	0.3736	0.3369	0.4054	0.3621
SimMTM	0.4438	0.3764	**0.3841**	0.2916	0.3613	0.2674	0.5777	**0.2152**	0.3196	0.3800	0.3455	0.4087	0.3643
PatchTST-SSL	0.4722	0.3826	0.3971	0.2860	**0.3561**	0.2724	0.6459	0.2305	0.3268	0.4020	0.3615	0.4148	0.3790
SimTS	0.4616	0.3840	0.3984	0.2846	0.3641	0.2749	0.6413	0.2250	0.3710	0.4109	0.3738	0.4236	0.3844
MAE													
$SDTA_4$	**0.4340**	**0.3994**	0.3979	**0.3283**	0.4037	0.2838	**0.3507**	0.3032	0.3488	0.3995	**0.3658**	0.3855	**0.3667**
TimeMAE-CI	0.4392	0.4068	0.3988	0.3336	0.4026	**0.2822**	0.3731	0.3004	**0.3405**	**0.3901**	0.3658	**0.3766**	0.3675
CPC	0.4386	0.4010	**0.3968**	0.3322	**0.4006**	0.2874	0.3841	0.2965	0.3685	0.4153	0.3844	0.3990	0.3754
TimeDART	0.4368	0.4009	0.4016	0.3307	0.4060	0.2844	0.3856	0.3134	0.3769	0.4175	0.3829	0.4034	0.3783
TimeSiam	0.4373	0.4077	0.3984	0.3337	0.4057	0.2860	0.3648	0.3072	0.3898	0.4170	0.3925	0.4222	0.3802
SimMTM	0.4376	0.4030	0.3973	0.3383	0.4051	0.2878	0.3711	**0.2949**	0.3774	0.4169	0.3952	0.4184	0.3786
PatchTST-SSL	0.4467	0.4045	0.4020	0.3317	0.4018	0.2899	0.4170	0.3155	0.3807	0.4275	0.3998	0.4211	0.3865
SimTS	0.4434	0.4047	0.3984	0.3303	0.4072	0.2927	0.3952	0.3067	0.4033	0.4307	0.4049	0.4208	0.3865

E STATISTICAL SIGNIFICANCE

Every test in the main text and in this appendix follows the same protocol (Demšar, 2006): one dataset contributes one paired observation, equal to the five-seed mean of its prediction-length-averaged score (a single length at the extended horizon), and the test is a two-sided Wilcoxon signed-rank test over those observations. Table 21 and Table 22 print the input vectors themselves, so every p value below can be recomputed without access to the runs; Table 23 is the matching input matrix for the supervision-axis rows. Table 24 then reports the standard-horizon tests, Table 26 the extended-horizon tests, and Table 25 the Friedman average ranks.

Two properties of this design are worth stating because they bound what the tests can show. With twelve datasets the smallest attainable two-sided p under the exact signed-rank distribution is $2/2^{12} = 4.9 \times 10^{-4}$, and with eleven it is 9.8×10^{-4}; a perfect twelve-of-twelve sweep therefore cannot report a p below that floor no matter how large the effect. And n is the number of *datasets*, not runs, so the five seeds buy precision in each observation rather than power in the test. This is the conservative choice and it is why the gap to the runner-up does not reach significance.

Table 22: Paired input matrix for the extended-horizon tests: five-seed mean at the single long prediction length 1080 (144 for PEMS) on the eleven datasets that admit it. exchange is excluded because its validation split cannot supply a 1080-step evaluation window. These are the paired vectors of Table 26. Best per column in **bold**, second underlined. Dataset abbreviations: Weat. = weather, Traf. = traffic, Elec. = electricity.

Method	ETTh1	ETTh2	ETTm1	ETTm2	Weat.	Traf.	Elec.	PEMS03	PEMS04	PEMS07	PEMS08	**Avg**
MSE												
$SDTA_4$	0.5582	0.4630	0.4858	0.4574	0.3955	**0.5941**	0.2944	0.5996	0.7976	**0.7251**	0.9250	**0.5723**
TimeMAE-CI	0.5872	0.4746	0.4935	0.4633	**0.3948**	0.6297	0.2949	**0.5828**	**0.7716**	0.7558	**0.9101**	0.5780
CPC	0.5743	**0.4624**	0.4927	0.4628	0.3993	0.6303	0.2926	0.6655	0.8837	0.8085	1.0066	0.6072
TimeDART	**0.5475**	0.4627	0.4962	0.4573	0.3950	0.6540	0.3024	0.7093	0.8607	0.7958	1.0048	0.6078
TimeSiam	0.5588	0.4691	0.4900	0.4617	0.3965	0.6065	0.2959	0.7297	0.8705	0.8108	1.0619	0.6138
SimMTM	0.5730	0.4675	**0.4830**	0.4744	0.3971	0.6295	**0.2921**	0.7331	0.8671	0.8356	1.0862	0.6217
PatchTST-SSL	0.5837	0.4675	0.5062	**0.4564**	0.4025	0.6930	0.3047	0.7060	0.9070	0.8489	1.0692	0.6314
SimTS	0.5796	0.4681	0.5053	0.4598	0.4026	0.6839	0.3023	0.9728	1.0308	0.9697	1.2422	0.6925
MAE												
$SDTA_4$	0.5122	0.4745	0.4571	0.4384	0.3757	**0.3760**	0.3617	0.5711	0.6704	**0.6265**	0.6599	0.5021
TimeMAE-CI	0.5230	0.4826	0.4610	0.4454	0.3754	0.3970	0.3605	**0.5530**	**0.6463**	0.6313	**0.6421**	**0.5016**
CPC	**0.5092**	**0.4727**	0.4586	0.4421	0.3789	0.4071	0.3560	0.5926	0.7005	0.6555	0.6828	0.5142
TimeDART	0.5128	0.4730	0.4599	0.4373	**0.3746**	0.4094	0.3714	0.6247	0.7045	0.6618	0.7013	0.5210
TimeSiam	0.5125	0.4802	0.4573	0.4432	0.3769	0.3879	0.3652	0.6412	0.7019	0.6713	0.7249	0.5239
SimMTM	0.5224	0.4794	**0.4558**	0.4492	0.3770	0.3965	**0.3555**	0.6299	0.6970	0.6743	0.7217	0.5235
PatchTST-SSL	0.5171	0.4766	0.4642	0.4368	0.3794	0.4384	0.3719	0.6126	0.7089	0.6736	0.7077	0.5261
SimTS	0.5222	0.4755	0.4602	**0.4359**	0.3799	0.4166	0.3661	0.7307	0.7634	0.7270	0.7774	0.5504

Table 23: Paired input matrix for the supervision-axis tests at the standard horizon, built the same way as Table 21. $SDTA_W$ is the Medium capacity $(E, D)=(2, 1)$ at window count W; $TSiam_W$ is TimeSiam with W target windows per anchor; CPC_K is CPC with K prediction steps. No entry is highlighted because the rows are not competing methods. Dataset abbreviations: Exch. = exchange, Weat. = weather, Traf. = traffic, Elec. = electricity.

Method	ETTh1	ETTh2	ETTm1	ETTm2	Exch.	Weat.	Traf.	Elec.	PEMS03	PEMS04	PEMS07	PEMS08	**Avg**
MSE													
$SDTA_1$	0.4394	0.3785	0.3966	0.2826	0.3636	0.2651	0.5838	0.2227	0.3121	0.3699	0.3259	0.3715	0.3593
$SDTA_2$	0.4349	0.3771	0.3891	0.2803	0.3589	0.2638	0.5549	0.2206	0.2765	0.3509	0.3048	0.3557	0.3473
$SDTA_4$	0.4348	0.3744	0.3854	0.2811	0.3623	0.2638	0.5443	0.2176	0.2691	0.3474	0.3019	0.3517	0.3445
$SDTA_8$	0.4360	0.3743	0.3844	0.2815	0.3629	0.2627	0.5464	0.2172	0.2690	0.3486	0.3017	0.3523	0.3448
$TSiam_1$	0.4433	0.3815	0.3888	0.2862	0.3636	0.2655	0.5576	0.2189	0.3239	0.3736	0.3369	0.4054	0.3621
$TSiam_4$	0.4421	0.3807	0.3932	0.2888	0.3642	0.2662	0.5560	0.2202	0.3233	0.3763	0.3493	0.4159	0.3647
CPC_1	0.4619	0.3779	0.3872	0.2854	0.3568	0.2683	0.5826	0.2165	0.3015	0.3791	0.3362	0.3791	0.3610
CPC_4	0.4643	0.3784	0.3896	0.2862	0.3554	0.2683	0.5847	0.2161	0.3051	0.3876	0.3411	0.3891	0.3638
MAE													
$SDTA_1$	0.4377	0.4013	0.4018	0.3300	0.4041	0.2837	0.3728	0.3115	0.3791	0.4168	0.3867	0.4010	0.3772
$SDTA_2$	0.4331	0.4000	0.3983	0.3273	0.4017	0.2838	0.3579	0.3086	0.3545	0.4029	0.3683	0.3878	0.3687
$SDTA_4$	0.4340	0.3994	0.3979	0.3283	0.4037	0.2838	0.3507	0.3032	0.3488	0.3995	0.3658	0.3855	0.3667
$SDTA_8$	0.4350	0.4006	0.3987	0.3281	0.4041	0.2828	0.3525	0.3024	0.3494	0.4004	0.3664	0.3853	0.3671
$TSiam_1$	0.4373	0.4077	0.3984	0.3337	0.4057	0.2860	0.3648	0.3072	0.3898	0.4170	0.3925	0.4222	0.3802
$TSiam_4$	0.4366	0.4074	0.4012	0.3363	0.4059	0.2868	0.3651	0.3096	0.3881	0.4203	0.4055	0.4311	0.3828
CPC_1	0.4386	0.4010	0.3968	0.3322	0.4006	0.2874	0.3841	0.2965	0.3685	0.4153	0.3844	0.3990	0.3754
CPC_4	0.4397	0.4013	0.3966	0.3334	0.4012	0.2866	0.3868	0.2959	0.3709	0.4204	0.3880	0.4056	0.3772

Table 24: Standard-horizon significance tests, Demšar protocol: one paired observation per dataset, equal to the five-seed mean of the score averaged over the four canonical prediction lengths (Table 21). Every test is a two-sided Wilcoxon signed-rank test. With n=12, no zero differences and no tied absolute differences, `scipy.stats.wilcoxon` selects the *exact* signed-rank distribution in every row, so the smallest attainable two-sided p is $2/2^{12}$=0.00049; T is the smaller signed-rank sum. Δ is the mean paired difference over the twelve datasets, negative when the left-hand side of the comparison is better. Panel (a) applies Holm–Bonferroni over the seven baseline comparisons within each metric. Panel (b) reports uncorrected p because the four SDTA steps form a single planned monotonicity probe; note that $W{:}2\to4$ is significant on MSE but not on MAE, and that the TimeSiam and CPC rows have *positive* Δ. **Bold** marks $p<0.05$ toward improvement (Holm-corrected in (a), raw in (b)); a trailing $^-$ marks $p<0.05$ toward *degradation*, following the convention of the main text (Table 2). Four compared pairs coincide at the four decimals printed in Tables 21 and 23, and are non-zero at full precision: SDTA against TimeMAE-CI on PEMS07 MAE, the $W{:}2\to4$ step on weather in both metrics, and CPC's K step on weather MSE. Recomputing those four rows from the printed table alone therefore yields one fewer win and a normal-approximation p, because a zero difference takes `scipy` out of its exact branch. Friedman omnibus over the eight methods: MSE $\chi^2(7)$=42.28, p=4.6×10^{-7}; MAE $\chi^2(7)$=34.61, p=1.3×10^{-5}; average ranks in Table 25.

	MSE					MAE				
Comparison	Δ	Wins	T	p	p_{Holm}	Δ	Wins	T	p	p_{Holm}
(a) SDTA (Medium, W=4) vs. each baseline; n=12 datasets; Holm over the seven comparisons										
TimeMAE-CI	−0.0047	7/12	27	0.380	0.380	−0.0008	6/12	36	0.850	0.850
CPC	−0.0166	10/12	7	0.0093	**0.028**	−0.0086	9/12	11	0.027	0.054
TimeDART	−0.0169	12/12	0	0.00049	**0.0034**	−0.0116	12/12	0	0.00049	**0.0034**
TimeSiam	−0.0176	12/12	0	0.00049	**0.0034**	−0.0135	12/12	0	0.00049	**0.0034**
SimMTM	−0.0198	9/12	7	0.0093	**0.028**	−0.0119	10/12	7	0.0093	**0.028**
PatchTST-SSL	−0.0345	11/12	2	0.0015	**0.0059**	−0.0198	11/12	1	0.00098	**0.0039**
SimTS	−0.0399	12/12	0	0.00049	**0.0034**	−0.0198	12/12	0	0.00049	**0.0034**
(b) supervision-axis steps; n=12 datasets; uncorrected; "Wins" counts datasets improved by the step										
SDTA $W{:}1\to2$	−0.0120	12/12	0	**0.00049**	–	−0.0085	11/12	1	**0.00098**	–
SDTA $W{:}2\to4$	−0.0028	10/12	10	**0.021**	–	−0.0020	9/12	15	0.064	–
SDTA $W{:}1\to4$	−0.0148	12/12	0	**0.00049**	–	−0.0105	11/12	1	**0.00098**	–
SDTA $W{:}4\to8$	+0.0003	6/12	27	0.380	–	+0.0004	4/12	19	0.129	–
TimeSiam $W{:}1\to4$	+0.0026	4/12	18	0.110	–	+0.0026	3/12	12	0.034^-	–
CPC $K{:}1\to4$	+0.0028	3/12	8	0.012^-	–	+0.0018	3/12	11	0.027^-	–

Table 25: Friedman average ranks over the eight methods (rank 1 is best on a dataset; lower is better), for the same paired matrices used by the Wilcoxon tests. `scipy.stats.friedmanchisquare` always evaluates the omnibus statistic against its asymptotic χ^2_7 distribution with the standard tie correction; no exact permutation variant is offered for this test, so these four p values are the only approximate ones in the appendix. SDTA holds the best average rank in all four columns.

	Standard (n=12)		Extended (n=11)	
Method	MSE	MAE	MSE	MAE
SDTA (Medium_4)	**1.92**	**2.00**	**2.18**	**2.36**
TimeMAE-CI	3.00	3.42	3.82	4.00
CPC	4.08	3.33	4.18	3.36
TimeDART	4.17	4.50	3.64	4.27
TimeSiam	4.75	5.42	4.55	4.91
SimMTM	4.25	4.42	4.64	4.82
PatchTST-SSL	6.67	6.42	6.09	5.91
SimTS	7.17	6.50	6.91	6.36
$\chi^2(7)$	42.28	34.61	27.58	21.79
p	4.6×10^{-7}	1.3×10^{-5}	2.6×10^{-4}	0.0028

Table 26: Extended-horizon significance tests. One paired observation per dataset: the five-seed mean at the single long prediction length 1080 (144 for PEMS) over the eleven datasets that admit it, i.e. the vectors of Table 22. Two-sided Wilcoxon signed-rank; with n=11, no zero differences and no tied absolute differences, `scipy.stats.wilcoxon` selects the *exact* distribution in every row and the smallest attainable two-sided p is $2/2^{11}$=0.00098. Δ is the mean paired difference over the eleven datasets, negative when the left-hand side is better. Holm correction in panel (a) runs over the seven baseline comparisons within each metric. After correction SDTA is significantly better than 5 of the seven baselines on MSE (CPC, TimeSiam, SimMTM, PatchTST-SSL, SimTS) and 3 on MAE (TimeSiam, PatchTST-SSL, SimTS); the MSE exceptions are TimeMAE-CI and TimeDART, with TimeMAE-CI remaining statistically indistinguishable. Panel (b) shows the supervision axis still carries the gain at the long horizon: $W\colon 1 \to 4$ remains significant (p=0.0068 on MSE, 9/11 datasets), while $W\colon 4 \to 8$ is flat. Friedman omnibus over the eight methods at this horizon: MSE $\chi^2(7)$=27.58, p=2.6×10^{-4}; MAE $\chi^2(7)$=21.79, p=0.0028 (ranks in Table 25). TimeSiam's W axis and CPC's K axis were never run at the extended horizon, so no long-horizon counterpart of those two axis tests exists.

	MSE					MAE				
Comparison	Δ	Wins	T	p	p_{Holm}	Δ	Wins	T	p	p_{Holm}
(a) SDTA (Medium, W=4) vs. each baseline at prediction length 1080 (144 for PEMS); n=11; Holm over the seven comparisons										
TimeMAE-CI	−0.0057	7/11	23	0.413	0.413	+0.0005	6/11	31	0.898	0.898
CPC	−0.0348	9/11	3	0.0049	**0.020**	−0.0120	8/11	11	0.054	0.107
TimeDART	−0.0355	7/11	12	0.067	0.135	−0.0188	8/11	9	0.032	0.097
TimeSiam	−0.0414	11/11	0	0.00098	**0.0068**	−0.0217	11/11	0	0.00098	**0.0068**
SimMTM	−0.0493	9/11	5	0.0098	**0.029**	−0.0214	9/11	6	0.014	0.055
PatchTST-SSL	−0.0590	10/11	1	0.0020	**0.0098**	−0.0240	10/11	1	0.0020	**0.012**
SimTS	−0.1201	11/11	0	0.00098	**0.0068**	−0.0483	10/11	2	0.0029	**0.015**
(b) SDTA supervision axis at the extended horizon; n=11; uncorrected										
SDTA $W\colon 1 \to 2$	−0.0304	11/11	0	**0.00098**	–	−0.0161	10/11	2	**0.0029**	–
SDTA $W\colon 2 \to 4$	−0.0036	7/11	16	0.147	–	−0.0022	6/11	17	0.175	–
SDTA $W\colon 1 \to 4$	−0.0341	9/11	4	**0.0068**	–	−0.0182	8/11	6	**0.014**	–
SDTA $W\colon 4 \to 8$	−0.0002	7/11	28	0.700	–	+0.0001	6/11	27	0.638	–

Table 27: Full ablation summary at the standard horizon (12 datasets, four prediction lengths each, 5 seeds 2021–2025; PEMS uses $\{12, 24, 48, 96\}$ and every other dataset $\{96, 192, 336, 720\}$). All arms use the Medium encoder (E=2, D=1) with W=4 views. ΔMSE is the change in the twelve-dataset mean relative to SDTA; every arm degrades accuracy, so every Δ is positive. σ_{def} is the cross-seed standard deviation of the default's own twelve-dataset mean (0.0010). *Worse datasets* counts datasets whose MSE exceeds the default's at the four-decimal precision of Table 29; ties at that precision exist, three of them ($K = V$ on ETTm1 and weather, no-TDA on ETTh2), and are counted here as not-worse; resolved on the unrounded means they read 7/12 for $K = V$ and 11/12 for no-TDA. *Paired seeds* counts, out of the five seeds, how often the arm's twelve-dataset mean is worse than the default's at the *same* seed; only $K = V$ ever wins a seed. Every Δ here is computed from the *unrounded* twelve-dataset means, so it can differ by one unit in the last digit from the difference of the two rounded values printed in this table. That happens for five arms (linear probe, no anchor, no diffusion, broadcast token and no forced s=1), because the default's own unrounded twelve-dataset mean, 0.344480, sits just below the 0.34450 that it prints as. The only arm where the main text and this table therefore differ *in print* is the broadcast token, whose unrounded Δ is $+0.00385$: Table 3 of the main text prints $+0.0038$ and $+0.0039$ appears here. No claim in either table turns on the difference.

Ablation arm	Pretraining / training flag	12-ds MSE (mean ± std)	12-ds MAE (mean ± std)	ΔMSE vs. default	$\Delta/\sigma_{\text{def}}$ (σ_{def}=.0010)	Worse datasets	Paired seeds
SDTA (default)	–	$0.3445_{\pm.0010}$	$0.3667_{\pm.0008}$	–	–	–	–
random init (no pretraining)	`--load_pretrain_weights 0`	$0.3738_{\pm.0026}$	$0.3825_{\pm.0014}$	$+0.0293$	29.3	12/12	5/5
linear probe (frozen encoder)	`--freeze_encoder 1`	$0.3716_{\pm.0010}$	$0.3830_{\pm.0009}$	$+0.0272$	27.2	10/12	5/5
no anchor conditioning	`--past_disabled 1`	$0.3672_{\pm.0006}$	$0.3795_{\pm.0005}$	$+0.0228$	22.8	12/12	5/5
no diffusion (clean targets)	`--diffusion_noise_disabled 1`	$0.3665_{\pm.0036}$	$0.3820_{\pm.0035}$	$+0.0221$	22.1	12/12	5/5
broadcast token	`--lineage_type learnable_token`	$0.3483_{\pm.0007}$	$0.3695_{\pm.0006}$	$+0.0039$	3.9	10/12	5/5
no temporal-distance attention	`--lineage_disabled 1`	$0.3482_{\pm.0007}$	$0.3693_{\pm.0005}$	$+0.0037$	3.7	10/12	5/5
no forced s=1 target	`--forced_first_shift_one 0`	$0.3459_{\pm.0008}$	$0.3679_{\pm.0006}$	$+0.0015$	1.5	9/12	5/5
$K = V$ (code on keys and values)	`--kv_share_lineage 1`	$0.3454_{\pm.0008}$	$0.3673_{\pm.0006}$	$+0.0009$	0.9	6/12	4/5

F Ablations in Full

Table 3 of the main text gives one column, the twelve-dataset ΔMSE. Table 27 adds the deviation of each arm, the size of the effect in units of the default's own seed noise, how many datasets it moves, and how many of the five paired seeds it moves. Tables 28, 29 and 30 give the group and per-dataset detail, and Table 31 the three arms that were also run at the extended horizon.

The paired-seed column is the more informative of the two significance views. An arm's twelve-dataset mean and the default's are computed from the same five seeds, so they can be compared seed by seed, which removes the shared seed variance. Under that comparison the two attention arms are unambiguous, worse in five of five seeds, while $K = V$ is worse in only four of five. The unpaired view agrees: measured against the default's cross-seed deviation of 0.0010, removing the distance code costs 3.7 deviations and replacing it with a discrete broadcast token 3.9, whereas $K = V$ costs 0.9 and sits inside the noise. A stricter scale is the standard error of the difference of two five-seed means, about 0.00055, under which the same three arms read 6.8, 7.0 and 1.7. Whichever scale is used, the ordering is the same and the conclusion is the one the main text draws: the distance code is load-bearing and the key-side restriction is a refinement whose value only becomes visible at the longer horizon.

The core arms need no such care. Each costs six to thirty times what any refinement costs, and the two largest, removing pretraining altogether and freezing the encoder, are worse than the default on every seed and on twelve and ten of the twelve datasets respectively. The gap between random init (0.3738) and no-anchor conditioning (0.3672) is what a trained patch embedding alone recovers, since the no-anchor arm gives the encoder no gradient.

Table 28: Group-level ablation at the standard horizon. 4-ETT = {ETTh1, ETTh2, ETTm1, ETTm2}, 4-Wide = {exchange, weather, traffic, electricity}, 4-PEMS = {PEMS03, PEMS04, PEMS07, PEMS08}. Means are taken per seed over the group and then across the five seeds; subscripts are cross-seed sample standard deviations. **Bold** marks the best arm per column and underline the second best. The cost of removing a component is strongly group dependent: the pretraining-signal arms (random init, linear probe, no anchor, no diffusion) lose 0.045–0.064 MSE on 4-PEMS but under 0.009 on 4-ETT, whereas the geometry arms (broadcast token, no TDA) lose only about 0.002 on 4-ETT and 0.003 on 4-Wide against 0.006–0.007 on 4-PEMS. $K = V$ is the only arm that is not uniformly worse: it is marginally *better* than SDTA on 4-ETT (0.3687 vs. 0.3689 MSE and 0.3895 vs. 0.3899 MAE) while losing on 4-Wide (0.3477 vs. 0.3470) and 4-PEMS (0.3198 vs. 0.3175), which is why its twelve-dataset penalty is only $+0.0009$.

Ablation arm	4-ETT		4-Wide		4-PEMS		12-ds	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
SDTA (default)	$0.3689_{\pm.0005}$	$0.3899_{\pm.0004}$	$\mathbf{0.3470_{\pm.0009}}$	$\mathbf{0.3354_{\pm.0006}}$	$\mathbf{0.3175_{\pm.0027}}$	$\mathbf{0.3749_{\pm.0022}}$	$\mathbf{0.3445_{\pm.0010}}$	$\mathbf{0.3667_{\pm.0008}}$
random init (no pretraining)	$0.3774_{\pm.0007}$	$0.3926_{\pm.0005}$	$0.3693_{\pm.0018}$	$0.3472_{\pm.0005}$	$0.3747_{\pm.0065}$	$0.4078_{\pm.0038}$	$0.3738_{\pm.0026}$	$0.3825_{\pm.0014}$
linear probe (frozen encoder)	$0.3738_{\pm.0005}$	$0.3902_{\pm.0004}$	$0.3595_{\pm.0017}$	$0.3419_{\pm.0007}$	$0.3816_{\pm.0015}$	$0.4170_{\pm.0022}$	$0.3716_{\pm.0010}$	$0.3830_{\pm.0009}$
no anchor conditioning	$0.3761_{\pm.0006}$	$0.3919_{\pm.0001}$	$0.3636_{\pm.0009}$	$0.3442_{\pm.0004}$	$0.3621_{\pm.0022}$	$0.4024_{\pm.0018}$	$0.3672_{\pm.0006}$	$0.3795_{\pm.0005}$
no diffusion (clean targets)	$0.3743_{\pm.0006}$	$0.3930_{\pm.0008}$	$0.3629_{\pm.0041}$	$0.3450_{\pm.0020}$	$0.3624_{\pm.0088}$	$0.4080_{\pm.0092}$	$0.3665_{\pm.0036}$	$0.3820_{\pm.0035}$
broadcast token	$0.3707_{\pm.0008}$	$0.3905_{\pm.0005}$	$0.3501_{\pm.0005}$	$0.3377_{\pm.0005}$	$0.3242_{\pm.0016}$	$0.3802_{\pm.0010}$	$0.3483_{\pm.0007}$	$0.3695_{\pm.0006}$
no temporal-distance attention	$0.3708_{\pm.0010}$	$0.3905_{\pm.0001}$	$0.3501_{\pm.0010}$	$0.3375_{\pm.0005}$	$0.3237_{\pm.0012}$	$0.3798_{\pm.0013}$	$0.3482_{\pm.0007}$	$0.3693_{\pm.0005}$
no forced $s{=}1$ target	$0.3714_{\pm.0009}$	$0.3919_{\pm.0003}$	$0.3483_{\pm.0017}$	$0.3363_{\pm.0010}$	$0.3182_{\pm.0020}$	$0.3755_{\pm.0013}$	$0.3459_{\pm.0008}$	$0.3679_{\pm.0006}$
$K = V$ (code on keys and values)	$\mathbf{0.3687_{\pm.0006}}$	$\mathbf{0.3895_{\pm.0004}}$	$0.3477_{\pm.0016}$	$0.3360_{\pm.0008}$	$0.3198_{\pm.0029}$	$0.3763_{\pm.0020}$	$0.3454_{\pm.0008}$	$0.3673_{\pm.0006}$

Table 29: Per-dataset MSE for all nine ablation arms at the standard horizon. Each cell is the mean over the four standard prediction lengths and then over the five seeds 2021–2025, with the cross-seed sample standard deviation as a subscript. Column keys: Default = SDTA ($E{=}2$, $D{=}1$, $W{=}4$); Rand. init = no pretraining; Lin. probe = frozen encoder; No anchor = no anchor conditioning; No diff. = clean (non-diffused) targets; Bcast. tok. = discrete learnable lineage token; No TDA = no temporal-distance attention; No $s{=}1$ = no forced unit-shift target; $K = V$ = lineage code applied to keys and values. **Bold** marks the best arm in each row and underline the second best; marks follow the unrounded means, so the weather row, where the default and $K = V$ both print 0.2638, is separated at full precision. Group rows average the per-seed group means before the cross-seed statistics are taken.

Dataset	Default	Rand. init	Lin. probe	No anchor	No diff.	Bcast. tok.	No TDA	No $s{=}1$	$K = V$
ETTh1	$0.4348_{\pm.0018}$	$0.4490_{\pm.0007}$	$0.4397_{\pm.0016}$	$0.4473_{\pm.0013}$	$0.4406_{\pm.0026}$	$0.4409_{\pm.0023}$	$0.4418_{\pm.0027}$	$0.4421_{\pm.0028}$	$\mathbf{0.4344_{\pm.0014}}$
ETTh2	$0.3744_{\pm.0014}$	$0.3816_{\pm.0016}$	$0.3769_{\pm.0010}$	$0.3797_{\pm.0019}$	$0.3781_{\pm.0028}$	$0.3742_{\pm.0016}$	$0.3744_{\pm.0018}$	$0.3754_{\pm.0009}$	$\mathbf{0.3740_{\pm.0009}}$
ETTm1	$0.3854_{\pm.0018}$	$0.3964_{\pm.0014}$	$0.3983_{\pm.0006}$	$0.3936_{\pm.0007}$	$0.3941_{\pm.0040}$	$\mathbf{0.3844_{\pm.0012}}$	$0.3850_{\pm.0005}$	$0.3849_{\pm.0010}$	$0.3854_{\pm.0014}$
ETTm2	$0.2811_{\pm.0009}$	$0.2825_{\pm.0004}$	$\mathbf{0.2805_{\pm.0007}}$	$0.2837_{\pm.0005}$	$0.2845_{\pm.0016}$	$0.2832_{\pm.0003}$	$0.2821_{\pm.0012}$	$0.2830_{\pm.0011}$	$0.2809_{\pm.0010}$
exchange	$0.3623_{\pm.0027}$	$0.3652_{\pm.0030}$	$\mathbf{0.3609_{\pm.0078}}$	$0.3654_{\pm.0020}$	$0.3747_{\pm.0118}$	$0.3652_{\pm.0033}$	$0.3641_{\pm.0027}$	$0.3636_{\pm.0053}$	$0.3618_{\pm.0042}$
weather	$0.2638_{\pm.0009}$	$0.2698_{\pm.0007}$	$0.2703_{\pm.0007}$	$0.2676_{\pm.0004}$	$0.2706_{\pm.0021}$	$0.2639_{\pm.0010}$	$0.2645_{\pm.0009}$	$0.2642_{\pm.0010}$	$\mathbf{0.2638_{\pm.0006}}$
traffic	$\mathbf{0.5443_{\pm.0010}}$	$0.6191_{\pm.0043}$	$0.5817_{\pm.0012}$	$0.5996_{\pm.0041}$	$0.5841_{\pm.0073}$	$0.5526_{\pm.0013}$	$0.5534_{\pm.0019}$	$0.5480_{\pm.0019}$	$0.5473_{\pm.0018}$
electricity	$0.2176_{\pm.0005}$	$0.2232_{\pm.0004}$	$0.2250_{\pm.0010}$	$0.2216_{\pm.0002}$	$0.2221_{\pm.0012}$	$0.2187_{\pm.0008}$	$0.2184_{\pm.0006}$	$\mathbf{0.2174_{\pm.0002}}$	$0.2181_{\pm.0007}$
PEMS03	$\mathbf{0.2691_{\pm.0029}}$	$0.3463_{\pm.0103}$	$0.3432_{\pm.0054}$	$0.3305_{\pm.0060}$	$0.3313_{\pm.0150}$	$0.2751_{\pm.0033}$	$0.2747_{\pm.0027}$	$0.2698_{\pm.0011}$	$0.2730_{\pm.0035}$
PEMS04	$\mathbf{0.3474_{\pm.0032}}$	$0.3934_{\pm.0048}$	$0.4046_{\pm.0044}$	$0.3846_{\pm.0018}$	$0.3835_{\pm.0081}$	$0.3517_{\pm.0014}$	$0.3539_{\pm.0039}$	$0.3485_{\pm.0028}$	$0.3479_{\pm.0029}$
PEMS07	$0.3019_{\pm.0013}$	$0.3541_{\pm.0066}$	$0.3670_{\pm.0057}$	$0.3396_{\pm.0013}$	$0.3445_{\pm.0062}$	$0.3072_{\pm.0036}$	$0.3079_{\pm.0026}$	$\mathbf{0.3018_{\pm.0024}}$	$0.3035_{\pm.0011}$
PEMS08	$\mathbf{0.3517_{\pm.0053}}$	$0.4049_{\pm.0049}$	$0.4117_{\pm.0039}$	$0.3936_{\pm.0030}$	$0.3905_{\pm.0078}$	$0.3628_{\pm.0042}$	$0.3585_{\pm.0025}$	$0.3526_{\pm.0036}$	$0.3549_{\pm.0053}$
4-ETT	$0.3689_{\pm.0005}$	$0.3774_{\pm.0007}$	$0.3738_{\pm.0005}$	$0.3761_{\pm.0006}$	$0.3743_{\pm.0006}$	$0.3707_{\pm.0008}$	$0.3708_{\pm.0010}$	$0.3714_{\pm.0009}$	$\mathbf{0.3687_{\pm.0006}}$
4-Wide	$\mathbf{0.3470_{\pm.0009}}$	$0.3693_{\pm.0018}$	$0.3595_{\pm.0017}$	$0.3636_{\pm.0009}$	$0.3629_{\pm.0041}$	$0.3501_{\pm.0005}$	$0.3501_{\pm.0010}$	$0.3483_{\pm.0017}$	$0.3477_{\pm.0016}$
4-PEMS	$\mathbf{0.3175_{\pm.0027}}$	$0.3747_{\pm.0065}$	$0.3816_{\pm.0015}$	$0.3621_{\pm.0022}$	$0.3624_{\pm.0088}$	$0.3242_{\pm.0016}$	$0.3237_{\pm.0012}$	$0.3182_{\pm.0020}$	$0.3198_{\pm.0029}$
12-ds avg.	$\mathbf{0.3445_{\pm.0010}}$	$0.3738_{\pm.0026}$	$0.3716_{\pm.0010}$	$0.3672_{\pm.0006}$	$0.3665_{\pm.0036}$	$0.3483_{\pm.0007}$	$0.3482_{\pm.0007}$	$0.3459_{\pm.0008}$	$0.3454_{\pm.0008}$

Table 30: Per-dataset MAE for all nine ablation arms at the standard horizon. Layout and conventions follow Table 29.

Dataset	Default	Rand. init	Lin. probe	No anchor	No diff.	Bcast. tok.	No TDA	No $s{=}1$	$K{=}V$
ETTh1	$0.4340_{\pm.0011}$	$0.4400_{\pm.0009}$	$\mathbf{0.4316_{\pm.0014}}$	$0.4378_{\pm.0002}$	$0.4385_{\pm.0010}$	$0.4366_{\pm.0019}$	$0.4363_{\pm.0008}$	$0.4372_{\pm.0014}$	$0.4340_{\pm.0005}$
ETTh2	$0.3994_{\pm.0009}$	$0.4032_{\pm.0012}$	$0.4008_{\pm.0006}$	$0.4018_{\pm.0016}$	$0.4027_{\pm.0015}$	$0.4002_{\pm.0009}$	$0.4008_{\pm.0009}$	$0.4013_{\pm.0007}$	$\mathbf{0.3990_{\pm.0012}}$
ETTm1	$0.3979_{\pm.0008}$	$0.3985_{\pm.0007}$	$0.4019_{\pm.0012}$	$0.3975_{\pm.0003}$	$0.3994_{\pm.0025}$	$\mathbf{0.3958_{\pm.0009}}$	$0.3959_{\pm.0009}$	$0.3992_{\pm.0008}$	$0.3969_{\pm.0011}$
ETTm2	$0.3283_{\pm.0009}$	$0.3288_{\pm.0006}$	$\mathbf{0.3266_{\pm.0008}}$	$0.3304_{\pm.0009}$	$0.3316_{\pm.0023}$	$0.3295_{\pm.0005}$	$0.3289_{\pm.0011}$	$0.3299_{\pm.0012}$	$0.3282_{\pm.0010}$
exchange	$0.4037_{\pm.0015}$	$0.4067_{\pm.0014}$	$\mathbf{0.4026_{\pm.0041}}$	$0.4067_{\pm.0010}$	$0.4099_{\pm.0057}$	$0.4060_{\pm.0018}$	$0.4051_{\pm.0016}$	$0.4053_{\pm.0029}$	$0.4034_{\pm.0024}$
weather	$0.2838_{\pm.0007}$	$0.2879_{\pm.0004}$	$0.2884_{\pm.0006}$	$0.2856_{\pm.0003}$	$0.2882_{\pm.0014}$	$0.2842_{\pm.0005}$	$0.2846_{\pm.0005}$	$0.2839_{\pm.0007}$	$\mathbf{0.2837_{\pm.0006}}$
traffic	$\mathbf{0.3507_{\pm.0011}}$	$0.3860_{\pm.0008}$	$0.3709_{\pm.0011}$	$0.3782_{\pm.0016}$	$0.3737_{\pm.0033}$	$0.3564_{\pm.0006}$	$0.3561_{\pm.0010}$	$0.3535_{\pm.0016}$	$0.3530_{\pm.0014}$
electricity	$0.3032_{\pm.0006}$	$0.3082_{\pm.0005}$	$0.3058_{\pm.0006}$	$0.3063_{\pm.0004}$	$0.3081_{\pm.0014}$	$0.3043_{\pm.0014}$	$0.3039_{\pm.0012}$	$\mathbf{0.3026_{\pm.0005}}$	$0.3039_{\pm.0013}$
PEMS03	$\mathbf{0.3488_{\pm.0021}}$	$0.3921_{\pm.0060}$	$0.3966_{\pm.0041}$	$0.3855_{\pm.0037}$	$0.3941_{\pm.0157}$	$0.3533_{\pm.0025}$	$0.3528_{\pm.0013}$	$0.3498_{\pm.0014}$	$0.3515_{\pm.0026}$
PEMS04	$\mathbf{0.3995_{\pm.0028}}$	$0.4259_{\pm.0029}$	$0.4375_{\pm.0038}$	$0.4222_{\pm.0016}$	$0.4287_{\pm.0100}$	$0.4035_{\pm.0012}$	$0.4049_{\pm.0030}$	$0.4006_{\pm.0010}$	$0.4000_{\pm.0022}$
PEMS07	$\mathbf{0.3658_{\pm.0014}}$	$0.3985_{\pm.0038}$	$0.4090_{\pm.0050}$	$0.3912_{\pm.0013}$	$0.3994_{\pm.0063}$	$0.3703_{\pm.0022}$	$0.3705_{\pm.0013}$	$0.3662_{\pm.0015}$	$0.3672_{\pm.0006}$
PEMS08	$\mathbf{0.3855_{\pm.0041}}$	$0.4147_{\pm.0031}$	$0.4248_{\pm.0037}$	$0.4107_{\pm.0017}$	$0.4098_{\pm.0054}$	$0.3937_{\pm.0024}$	$0.3911_{\pm.0022}$	$0.3856_{\pm.0033}$	$0.3866_{\pm.0035}$
4-ETT	$0.3899_{\pm.0004}$	$0.3926_{\pm.0005}$	$0.3902_{\pm.0004}$	$0.3919_{\pm.0001}$	$0.3930_{\pm.0008}$	$0.3905_{\pm.0005}$	$0.3905_{\pm.0001}$	$0.3919_{\pm.0003}$	$\mathbf{0.3895_{\pm.0004}}$
4-Wide	$\mathbf{0.3354_{\pm.0006}}$	$0.3472_{\pm.0005}$	$0.3419_{\pm.0007}$	$0.3442_{\pm.0004}$	$0.3450_{\pm.0020}$	$0.3377_{\pm.0005}$	$0.3375_{\pm.0005}$	$0.3363_{\pm.0010}$	$0.3360_{\pm.0008}$
4-PEMS	$\mathbf{0.3749_{\pm.0022}}$	$0.4078_{\pm.0038}$	$0.4170_{\pm.0022}$	$0.4024_{\pm.0018}$	$0.4080_{\pm.0092}$	$0.3802_{\pm.0010}$	$0.3798_{\pm.0013}$	$0.3755_{\pm.0013}$	$0.3763_{\pm.0020}$
12-ds avg.	$\mathbf{0.3667_{\pm.0008}}$	$0.3825_{\pm.0014}$	$0.3830_{\pm.0009}$	$0.3795_{\pm.0005}$	$0.3820_{\pm.0035}$	$0.3695_{\pm.0006}$	$0.3693_{\pm.0005}$	$0.3679_{\pm.0006}$	$0.3673_{\pm.0006}$

Table 31: Extended-horizon ablation (prediction length 1080, or 144 for PEMS), 11 datasets (exchange is excluded because its validation split cannot supply a 1080-step evaluation window), 5 seeds. *No TDA* is the no-temporal-distance-attention arm. **Only the default and these two arms were run at the extended horizon**; the other six ablation arms of Table 27 exist at the standard horizon only. ΔMSE ext. is the penalty on the eleven-dataset extended mean, ΔMSE std. is the same arm's penalty on the twelve-dataset standard-horizon mean, and *Ampl.* is their ratio. Both geometry ablations hurt more at 1080 steps than at 720: no-TDA by $1.6\times$ and $K{=}V$ by $2.8\times$. The $K{=}V$ penalty is nearly invisible at the standard horizon ($+0.0009$, below one cross-seed standard deviation) but becomes a clear $+0.0026$ once the horizon is extended, so sharing the lineage code between keys and values costs precision that only long-range rollout exposes. As in Table 27, deltas and ratios come from the unrounded means: the $K{=}V$ extended penalty is $+0.0026$ unrounded and $+0.0027$ if the two rounded means are subtracted, the latter being the value quoted in the Ablations section of the main text, and the corresponding amplification is $2.8\times$ or $3.0\times$. Both readings sit inside the 1.6–$3\times$ range the main text states for the two attention arms.

Arm	11-ds MSE	11-ds MAE	ΔMSE ext.	ΔMSE std.	Ampl.
SDTA (default)	$0.5723_{\pm.0023}$	$0.5021_{\pm.0010}$	–	–	–
no TDA	$0.5782_{\pm.0022}$	$0.5055_{\pm.0013}$	$+0.0059$	$+0.0037$	$1.6\times$
$K{=}V$	$0.5750_{\pm.0022}$	$0.5036_{\pm.0010}$	$+0.0026$	$+0.0009$	$2.8\times$

Table 32: Pretraining cost at capacity Medium, sorted by peak VRAM *on PEMS07*: allocated memory and wall-clock time per epoch on three datasets of increasing width (channels in parentheses). **Bold** / underline = best / 2nd 12-ds MSE. ‡10% *pretraining* channel subsampling; memory understated (see text). †PatchTST-SSL's timer reports whole seconds.

	Peak VRAM (MiB)			Pretrain s/epoch			12-ds
Model	ETTh1 (7)	elec. (321)	PEMS07 (883)	ETTh1 (7)	elec. (321)	PEMS07 (883)	(MSE)
SDTA (W=4)	58	1790	4874	2.8	45.8	139.2	**0.3445**
CPC	47	1387	3754	1.6	18.7	53.9	0.3610
SDTA (W=2)	41	1088	2890	2.8	25.6	75.0	<u>0.3473</u>
TimeSiam	39	968	2596	6.9	58.4	147.2	0.3621
SimMTM‡	66	393	2316	5.8	41.3	137.7	0.3643
SDTA (W=1)	33	698	1861	2.7	17.3	47.7	0.3593
TimeDART	31	586	1548	3.2	26.9	63.3	0.3614
PatchTST-SSL†	30	553	1475	1.0	11.0	30.0	0.3790
SimTS	29	551	1461	1.7	15.8	46.2	0.3844
TimeMAE-CI	29	504	1343	2.7	19.8	65.2	0.3491

G Memory and Runtime Cost

Table 32, quoted in Section 5.4 of the main text and printed here, was cut to three datasets, the Medium capacity, and pretraining cost only. Tables 33, 34 and 35 restore all five datasets, the last two adding the $E \times D \times W$ grid at $W \in \{1, 2, 4\}$ and the fine-tuning side, and Figure 3 reads the same measurements as a cost–accuracy frontier.

All three tables follow the cost protocol: one seed, batch 16, one dataloader worker, one thread, and a single idle RTX 4090 running one job at a time. They measure relative cost under a fixed harness. They are not benchmark wall-clock and should not be compared against timings from other papers or machines. Peak allocated memory, unlike wall-clock, is a property of the configuration rather than the machine, which we verified by re-measuring the SDTA $W = 4$ row on the cost machine and recovering 4873.2 MiB against the 4874 MiB the accuracy node reports.

Two entries carry caveats that change how they should be read. CPC is measured at $K = 1$, which is both its standard setting and its best-accuracy one; at $K = 4$ the same PEMS07 cell would read 6730 MiB and 102.79 s instead of 3754 MiB and 53.94 s, so quoting $K = 4$ would nearly double its apparent cost. SimMTM's three high-channel cells are measured with the channel subsampling its official code provides, keeping a random ten percent of channels per pretraining batch, so those cells *understate* its true memory and time. That option applies to pretraining only. SimMTM's fine-tuning and evaluation read every channel, so its accuracy column is fully comparable and only its cost columns are optimistic. Its full-channel cost is still visible on weather, where all 21 channels are used and it spends 53.45 s per epoch, more than any other method on that dataset and more than it spends on the 321-channel electricity (41.32 s) under subsampling.

Table 33: **Peak pretraining VRAM (MiB, allocated working set)**, datasets ordered by channel count C; the last column is the twelve-dataset MSE of the same configuration. **Bold** / underline = best / second-best twelve-dataset MSE. SDTA is shown at all three window counts: memory grows close to linearly in W ($1861 \to 2890 \to 4874$ MiB on PEMS07, i.e. $1.8 \to 2.8 \to 4.8$ GiB), so the whole family fits a single consumer card at every W; $W = 2$ needs 59% of the $W = 4$ peak memory and gives up 0.0028 MSE. *CPC is measured at $K = 1$, its standard and best-accuracy setting; at $K = 4$ the same cell costs 6730 MiB, so reporting $K = 4$ would overstate CPC by $1.8\times$. ‡SimMTM's pretraining on the three high-channel datasets shown here, of the six on which the switch is enabled, keeps a random 10% of channels per batch (the `select_channels` option of its official code, used here to avoid the $\sim$48 GB that full-channel pretraining would need); the cells marked † are therefore *understated* and are not full-channel comparable, and the quadratic $(n_{\text{views}} \cdot B \cdot C)^2$ series-wise cost is visible instead on weather (203 MiB at only 21 channels, the largest low-channel cell in the table). SimMTM's *accuracy* is unaffected by this: `select_channels` applies to pretraining only, and fine-tuning and evaluation use every channel, so its MSE column is directly comparable. Protocol: peak `torch.cuda.max_memory_allocated` over a 2-epoch uncapped pretraining run, batch 16, seed 97 (one seed; memory is deterministic), `num_workers`= 1, `OMP_NUM_THREADS`= 1, single serial RTX 4090 (24 GB). The CPC and SDTA $W = 1$ rows were measured on the cost machine and the rest on the accuracy node; peak allocated memory is machine independent for a fixed configuration, verified by re-measuring the SDTA $W = 4$ row on the cost machine, which gives 4873.2 against the 4874 printed here. Accuracy: input length 96, prediction length $\in \{96, 192, 336, 720\}$, PEMS $\in \{12, 24, 48, 96\}$, mean over seeds $\{2021, \ldots, 2025\}$.

Method	ETTh1 ($C = 7$)	weather ($C = 21$)	PEMS08 ($C = 170$)	electricity ($C = 321$)	PEMS07 ($C = 883$)	12-ds MSE
SDTA Medium ($W = 4$)	58	136	955	1790	4874	**0.3445**
CPC*	47	106	739	1387	3754	0.3610
SDTA Medium ($W = 2$)	41	86	570	1088	2890	<u>0.3473</u>
TimeSiam	39	80	516	968	2596	0.3621
SimMTM‡	66	203	152†	393†	2316†	0.3643
SDTA Medium ($W = 1$)	33	61	373	698	1861	0.3593
TimeDART	31	55	315	586	1548	0.3614
PatchTST-SSL	30	55	299	553	1475	0.3790
SimTS	29	53	297	551	1461	0.3844
TimeMAE-CI	29	51	277	504	1343	0.3491

Table 34: **Per-epoch pretraining time (s/epoch)** over the SDTA $E \times D \times W$ grid at $W \in \{1, 2, 4\}$ and all eight methods, datasets ordered by channel count C; the last column is the twelve-dataset MSE of the same configuration. **Bold** / underline = best / second-best twelve-dataset MSE. Capacities are Small = $(E = 1, D = 1)$, Medium = $(E = 2, D = 1)$, Large = $(E = 2, D = 2)$; runs are named Size_W. The three cost axes separate cleanly on the high-channel end: W is close to linear (Medium on PEMS07, $47.67 \to 74.97 \to 139.16$ for $W = 1, 2, 4$), an extra encoder block is comparatively cheap ($E : 1 \to 2$ adds 10.5 s at $W = 1$ and 21.1 s at $W = 4$), and an extra pretraining-decoder block multiplies with W ($D : 1 \to 2$ adds 17.6 s at $W = 1$ but 83.2 s at $W = 4$). Reducing W or D therefore saves far more than reducing E. On low-channel data (≤ 21) fixed host overhead dominates and the W axis is nearly flat. *CPC is at $K = 1$, its standard and best-accuracy setting; the same PEMS07 cell at $K = 4$ costs 102.79 s/epoch, so reporting $K = 4$ would overstate CPC by $1.9\times$. ‡SimMTM's high-channel cells (marked †) are measured with its official `select_channels` option keeping a random 10% of channels per pretraining batch, so they are *understated* and not full-channel comparable; its 53.45 s on weather, which uses all 21 channels, is where its quadratic series-wise similarity cost is actually visible. Pretraining subsampling does not touch its accuracy column, which is fine-tuned and evaluated on every channel. §PatchTST-SSL logs only `mm:ss`, so its row has 1-second resolution; each of its epochs was individually constant (e.g. all six PEMS07 epochs read 00:30). This table prints two decimals where Table 32 prints one. The two agree everywhere except one cell: TimeDART on electricity measures 26.8455 s, which is 26.85 here and 26.8 at one decimal, while Table 32 prints 26.9. The two round differently at the last digit shown; no ordering, comparison or claim in either table depends on the difference. Protocol: 6 epochs, first dropped as warm-up, mean of epochs 2–6; batch 16, seed 97 (one seed; cost is near-deterministic), `num_workers`= 1, `OMP_NUM_THREADS`= 1, single serial RTX 4090 (24 GB) with no co-tenant. These are relative costs under one fixed harness, not benchmark wall-clock, and must not be compared against timings from other papers or machines.

Configuration (s/epoch)	ETTh1 ($C=7$)	weather ($C=21$)	PEMS08 ($C=170$)	electricity ($C=321$)	PEMS07 ($C=883$)	12-ds MSE
SDTA: the full $E \times D \times W$ grid						
SDTA Small_1	2.37	9.18	4.98	13.61	37.17	0.3649
SDTA Small_2	2.39	9.40	7.47	22.08	64.50	0.3579
SDTA Small_4	2.38	9.65	12.90	41.64	118.07	0.3564
SDTA Medium_1	2.71	10.66	6.10	17.28	47.67	0.3593
SDTA Medium_2	2.76	10.92	8.58	25.63	74.97	0.3473
SDTA Medium_4	2.80	11.19	14.14	45.79	139.16	**0.3445**
SDTA Large_1	3.42	13.68	8.36	26.91	65.27	0.3588
SDTA Large_2	3.45	13.69	12.50	38.59	113.78	0.3487
SDTA Large_4	3.93	17.49	26.37	85.46	222.35	<u>0.3450</u>
Baselines at the matched Medium encoder ($E=2$)						
TimeMAE-CI	2.69	12.59	12.73	19.78	65.21	0.3491
TimeDART	3.24	14.14	8.61	26.85	63.32	0.3614
TimeSiam	6.92	22.71	19.82	58.44	147.23	0.3621
SimMTM‡	5.83	53.45	14.72†	41.32†	137.67†	0.3643
PatchTST-SSL§	1.00	5.00	3.00	11.00	30.00	0.3790
CPC*	1.62	6.35	6.39	18.65	53.94	0.3610
SimTS	1.70	6.74	5.51	15.76	46.18	0.3844

Table 35: **Per-epoch fine-tuning (deployment) time (s/epoch, prediction length $= 96$)** over the SDTA $E \times D \times W$ grid at $W \in \{1, 2, 4\}$ and all eight methods, datasets ordered by channel count C. This is the table behind the main text's $26/26/30$ s/epoch on PEMS07 for $W = 1, 2, 4$: the measured values are 26.17, 26.35 and 30.33. Fine-tuning is essentially flat in W within each capacity, because W and the pretraining decoder are pretext-only and are discarded before transfer, and the deployed model is the same $17{,}440$-parameter encoder plus a linear head at Medium and Large (and $8{,}896$ at Small) regardless of W. The residual $W = 4$ excess on PEMS07 (30.33 vs. 26.17 at Medium, 34.57 vs. 26.18 at Large) is run-to-run and thermal noise on the one measured seed, not a structural cost: at every dataset other than PEMS07 the three W columns agree to within 0.8 s. SDTA is among the cheapest methods to fine-tune at every capacity: at PEMS07 only PatchTST-SSL (30.00) matches Medium_4 (30.33), and the three Small rows (16.20–16.27) are the cheapest cells, against 32.01 (TimeDART), 35.51 (SimMTM), 35.61 (SimTS), 35.70 (CPC), 40.63 (TimeMAE-CI) and 66.97 (TimeSiam). *CPC at $K = 1$. §PatchTST-SSL logs only `mm:ss`, so its row has 1-second resolution. Protocol: 6 epochs, first dropped as warm-up, mean of epochs 2–6; batch 16, seed 97 (one seed; cost is near-deterministic), `num_workers=` 1, `OMP_NUM_THREADS=` 1, single serial RTX 4090 (24 GB) with no co-tenant. These are relative costs under one fixed harness, not benchmark wall-clock.

Configuration (s/epoch)	ETTh1 ($C = 7$)	weather ($C = 21$)	PEMS08 ($C = 170$)	electricity ($C = 321$)	PEMS07 ($C = 883$)
SDTA: the full $E \times D \times W$ grid					
SDTA Small_1	1.07	4.13	2.22	5.87	16.27
SDTA Small_2	1.05	4.22	2.26	5.89	16.20
SDTA Small_4	1.06	4.13	2.24	5.88	16.26
SDTA Medium_1	1.39	5.68	3.33	9.40	26.17
SDTA Medium_2	1.49	5.62	3.36	9.44	26.35
SDTA Medium_4	1.44	5.69	3.36	9.74	30.33
SDTA Large_1	1.42	5.72	4.13	9.44	26.18
SDTA Large_2	1.41	5.64	3.33	9.39	26.21
SDTA Large_4	1.69	6.14	3.96	9.42	34.57
Baselines at the matched Medium encoder ($E = 2$)					
TimeMAE-CI	5.17	8.55	4.40	15.33	40.63
TimeDART	1.73	5.99	3.53	12.09	32.01
TimeSiam	2.94	9.41	10.41	29.93	66.97
SimMTM	1.38	5.48	4.36	12.46	35.51
PatchTST-SSL§	1.00	5.00	3.00	11.00	30.00
CPC*	1.44	5.58	4.39	12.49	35.70
SimTS	1.35	5.54	4.38	12.54	35.61

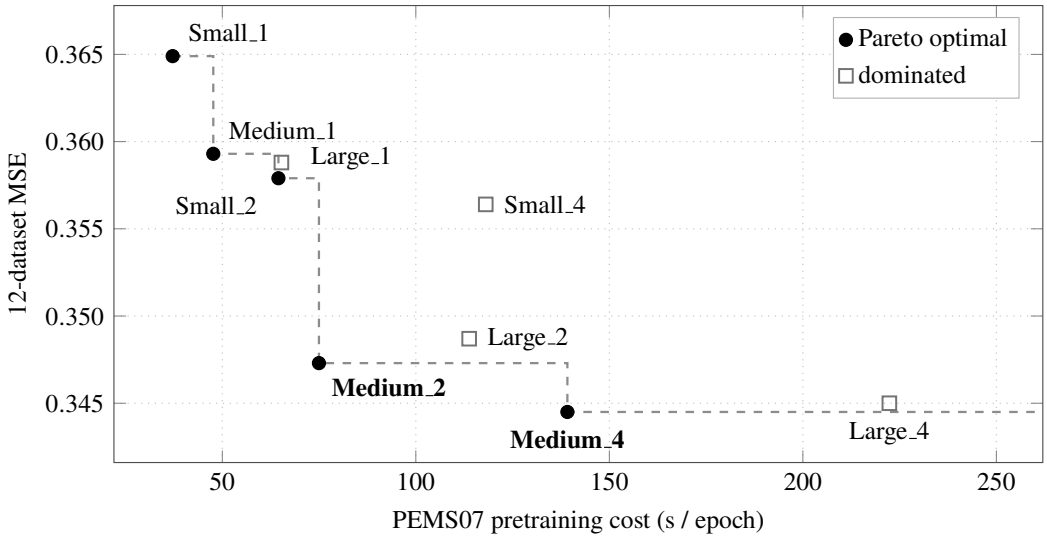

Figure 3: **Cost versus accuracy over the nine SDTA** (E, D, W) **points** (both axes minimised, so down-and-left is better). The horizontal axis is per-epoch pretraining time on PEMS07, the most cost-sensitive and the only reliably GPU-bound cell in the grid; the vertical axis is the twelve-dataset MSE averaged over five seeds. Runs are named Size_W with Small $= (E = 1, D = 1)$, Medium $= (E = 2, D = 1)$, Large $= (E = 2, D = 2)$. Five points are Pareto optimal (Small_1, Medium_1, Small_2, Medium_2, Medium_4) and four are dominated: Large_1 by Small_2 (64.50 s and 0.3579 beats 65.27 s and 0.3588), Small_4 and Large_2 both by Medium_2 (74.97 s and 0.3473 beats 118.07 s at 0.3564 and 113.78 s at 0.3487), and Large_4 by Medium_4 (139.16 s and 0.3445 beats 222.35 s at 0.3450). Two readings follow. First, *every* $D = 2$ *point is dominated*: deepening the pretraining decoder multiplies with W and buys nothing, which is the cost-side argument for $D = 1$ independently of accuracy. Second, the useful half of the frontier is the $E = 2, D = 1$ line: **Medium_2** is the practical sweet spot at 54% of the default's pretraining cost for $+0.0028$ MSE, and **Medium_4** is the accuracy optimum among the nine costed points, the $W = 8$ cells of Table 14 never having been timed; buying accuracy with a deeper encoder beats buying it with more windows, since Small_4 spends 57% more than Medium_2 and still loses by 0.0091 MSE. Costs are single-seed (seed 97) measurements under the fixed harness of the pretraining-time table, not benchmark wall-clock.

H Code Appendix and Reproducibility

The accompanying code reproduces every table and figure in the main text and in this appendix. It is released under an MIT license. The listing below is anonymized.

H.1 Layout

The repository holds one self-contained package per method over one shared encoder implementation and one shared data store, so the pretext is the only variable across the eight.

- `model/SDTA/` is the proposed method; `model/{TimeDART, TimeSiam, TimeMAE_CI, PatchTST, SimMTM, CPC, SimTS}/` are the seven baselines, each run as `python -m model.`*`id`*`.`*`stage`*; and `model/TimeSiam_W/` is TimeSiam fitted with SDTA's multi-window sampler, which is the axis control of Table 13 in the main text.
- `entry/batch.py` is the experiment runner (GPU scheduling, resume) and `entry/batch_queue.py` runs several of its configurations in sequence, and `entry/batch_configs/` holds every final run configuration as JSON, organised by method, capacity and dataset cohort.
- `download/` and `data_preprocess/` fetch the twelve public datasets and build the windowed training store.
- `scripts/` holds the analysis code: result aggregation, the significance tests, per-epoch timing, the peak-VRAM probe, and parameter counts.
- `docs/paper_code_map.md` maps each paper concept to its implementation, each paper table to its configuration, and each aggregate to its run-name template.

H.2 Reproducing Each Result

Every experiment runs through the one config-driven runner. A config expands into (dataset $\times$ grid $\times$ seed) chains of `pretrain` $\rightarrow$ `train` $\rightarrow$ `test` subprocesses; a finished stage leaves a sentinel and is skipped on re-runs, so an interrupted campaign resumes where it stopped. Architecture flags live only in the pretraining stage and fine-tuning inherits them from the pretraining run's recorded configuration, which makes an encoder/head mismatch impossible and is also why every number in Appendix A could be read back out of the runs themselves.

The aggregation script enforces the canonical four prediction lengths, which matters because the default configuration's result files also carry the extended horizon: averaging over every stored length instead of the four standard ones turns SDTA's twelve-dataset MSE from 0.3445 into 0.3866. Ablation arms, capacity cells, and baseline axis values are aggregated through a run-name template, one per table, documented in the repository's paper–code map; the templates themselves appear in the last column of Table 6. Table 36 gives the command behind each reported result.

H.3 Randomness and Determinism

Every stage takes a seed and seeds Python, NumPy and PyTorch (CPU and CUDA) at its start, and disables cuDNN autotuning. We do not force fully deterministic kernels, so a repeated run at a fixed seed agrees with the original to within the cross-seed noise reported throughout this appendix rather than bit for bit. That noise is what the standard deviations in Tables 11, 14 and 27 quantify, and it is why the paired tests of Appendix E compare arms seed by seed wherever the two arms share seeds. Accuracy results use the five seeds 2021–2025; cost measurements use the single seed 97, since cost is near-deterministic and the seed does not enter it.

Table 36: Command for each reported result. The aggregation and significance scripts read the same result files the training runs wrote, so they re-run no experiment and every table in this appendix can be checked without a GPU. The license is recorded in `LICENSE` (MIT) and `THIRD_PARTY.md`, the paper-referencing comments in the module headers and `docs/paper_code_map.md`, the seed method in `seed_everything` and Appendix A.6, and the evaluation metrics in the benchmark section of the main text.

Result	Command
Main results (Table 1), App. B	`entry/batch_queue.py --list main_table.txt`
W sweep and capacities (Figure 2), App. C	`entry/batch_queue.py --list capacity_sweep.txt`
Baseline axes (Table 13)	`entry/batch.py TimeSiam_W/medium/*`, `CPC/medium/*`
Extended horizon (Table 17), App. D	`entry/batch_queue.py --list extended.txt`
Ablations (Table 3), App. F	`entry/batch_queue.py --list SDTA/medium/ablation/all.txt`
Cost (Table 32), App. G	`entry/batch_queue.py --list time/list.txt --concurrent 1`
Aggregate any table	`scripts/aggregate_results.py`
Cross-seed standard deviations	`scripts/aggregate_results.py --std`
Significance (Table 2), App. E	`scripts/significance_test.py`
Wall-clock, App. G	`scripts/epoch_time.py`
Parameter counts, Table 8	`scripts/count_params.py`
Peak VRAM, Table 33	`scripts/vram_probe/`
Pre-processing	`download/`, then `data_preprocess/build_informer_windows.py`