# SPOT-VSA: EFFICIENT STABILITY–PLASTICITY-TUNABLE ON-LINE TIME SERIES FORECASTING WITH VECTOR SYMBOLIC ARCHITECTURES.

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Online time-series forecasting requires rapid adaptation to non-stationary data while preserving previously acquired patterns, confronting the stability–plasticity dilemma. Current approaches struggle along this frontier, either suffering catastrophic forgetting through aggressive adaptation or incurring heavy computational overhead through redundant or frozen backbones, and none quantifies its position on the tradeoff. We propose `SPOT-VSA`, an efficient neuro-vector-symbolic architecture for task-free online forecasting. By projecting inputs into a latent space via Vector Symbolic Architectures, the model introduces an explicit inductive bias where short- and long-term temporal dependencies are bound into a single representation with negligible interference. Forecasts are produced by a hybrid sequence-to-sequence and autoregressive strategy whose granularity sets the model's position on the stability–plasticity frontier. We ground this claim theoretically, deriving bounds on stability and plasticity that identify forecasting granularity as a calibration knob and show that the step-size tradeoff between the two emerges only under distribution drift. Across diverse benchmarks and horizons, `SPOT-VSA` matches or surpasses state-of-the-art online forecasters in accuracy, exhibits the lowest forgetting among methods adapting their full model, and validates theoretical predictions on synthetic drift streams. In addition, it achieves up to $404\times$ speedup and $390\times$ higher energy efficiency on edge devices, with up to $46\times$ fewer parameters.[1]

## 1 INTRODUCTION

Time-series forecasting underpins critical decisions in power grids, finance, transportation, and on-device monitoring. While deep forecasters excel on static benchmarks, real-world data streams are non-stationary: sensors drift, regimes shift, and seasonal dynamics evolve. Because retraining from scratch is computationally prohibitive, especially under edge budgets, practical deployment demands *online forecasting*, where a model predicts the upcoming horizon, observes the ground truth, and updates parameters sequentially in a single pass.

Continual adaptation inevitably confronts the *stability–plasticity dilemma*: forecasters must be plastic enough to track distribution shifts rapidly, yet stable enough to prevent catastrophic forgetting. Existing online forecasters resolve this tension through structural compromise rather than principled design. Fast-adapting architectures like FSNet (Pham et al., 2023) achieve high plasticity at the cost of rapid forgetting; ensemble models like OneNet (Zhang et al., 2023) and variational latent approaches like LSTD (Cai et al., 2025) improve retention, but incur double parameter counts or impose heavy auxiliary objectives; and output-probing models like CoSPOT (Choi et al., 2026) maintain stability by freezing an LLM backbone, but fail to track short-horizon dynamics. Consequently, no existing method occupies a favorable point along both stability-plasticity axes. Moreover, stability and plasticity are conventionally treated as qualitative goals rather than quantifiable metrics, leaving the Pareto frontier unmapped and design tradeoffs unmodeled.

In this work, we show that a compact, lightweight model can explicitly characterize and favorably traverse this frontier. First, we observe that the stability–plasticity tradeoff is directly governed by

[1] We will make our code available after paper acceptance.



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.

how the forecast horizon is generated. A forecaster can map a look-back window directly to the horizon in one pass (sequence-to-sequence), unroll predictions autoregressively one step at a time, or produce the horizon across intermediate chunks. Producing smaller chunks simplifies targets and allows more updates per window, boosting plasticity, but compounds rollout error and amplifies parameter drift, harming stability. Our theoretical analysis (Section 2.8) formalizes this mechanism: defining plasticity via rollout tracking error and stability via functional parameter shifts, we bound both quantities over the horizon, chunk count, and learning rate. These bounds reveal that a finite optimal chunking exists, that stability and plasticity improve simultaneously on stationary streams, and that an explicit tradeoff can emerge under distribution drift.

Second, we show that a forecaster can balance accuracy and retention without duplicating parameters or freezing massive backbones. We propose **SPOT-VSA**, a neuro-vector-symbolic architecture for task-free online forecasting (Pham et al., 2023). SPOT-VSA projects each look-back window into a latent space via a hybrid encoder, a frozen random projection element-wise reweighted by a learnable map, preserving channel similarity geometry at initialization (Proposition 2.1). This representation is bound via a Hadamard product to a persistent, input-independent latent vector (Section 2.5) that accumulates long-term inductive bias across windows. Horizon forecasts are generated through chunked rollouts, where granularity serves as a direct calibration knob along the stability-plasticity frontier. Because the architecture comprises only linear maps and coordinate-wise bindings, it operates strictly within edge compute budgets.

Our key contributions are:

- *A formal stability-plasticity frontier for online forecasting.* We formalize stability and plasticity for chunked horizon predictors and derive theoretical bounds with respect to horizon length, chunk count, and learning rate. These bounds explain when an optimal intermediate granularity exists, show that the step-size tradeoff vanishes without drift and emerges under non-stationarity. We validate these predictions on controlled synthetic streams.
- *The* `SPOT-VSA` *architecture.* We design a vector-symbolic forecaster that binds a dynamic window encoding to a persistent latent representation, using chunk granularity to navigate the stability-plasticity frontier without backbone ensembling, latent causal inference, or frozen foundation models. It includes an explicit inductive bias structure that improves both plasticity and stability.
- *Empirical accuracy, retention, and edge efficiency.* Across eight benchmark datasets with horizons up to 336 timesteps, `SPOT-VSA` matches or exceeds eight state-of-the-art online forecasters (Table 2) and demonstrates the lowest average forgetting among fully adapted models (Table 3). Furthermore, it delivers up to $404\times$ and $390\times$ reductions in latency and energy, respectively, on edge hardware (Jetson Nano and Cortex-A53) relative to existing online baselines under a matched measurement protocol (Appendix A.7).

## 2 Related Work and Preliminaries

### 2.1 Related Work

*Vector symbolic architectures.* Vector symbolic architectures (VSA), or hyperdimensional computing (Kanerva, 2009), embed data into high-dimensional latent spaces ($D \sim 10^3$) using element-wise binding (Hadamard product) and bundling (addition). Because quasi-orthogonal vectors have expected cosine similarities of $\mathcal{O}(1/\sqrt{D})$, interference remains bounded. This property makes VSA attractive for streaming tasks such as model superposition (Menet et al., 2024), compositional reasoning (Mejri et al., 2024b; Hersche et al., 2023), and edge learning (Hernández-Cano et al., 2021; Mejri et al., 2024a). However, standard frameworks rely on fixed random projections with learned readouts, lagging behind deep architectures in predictive accuracy (Duan et al., 2022). `SPOT-VSA` pairs frozen random projections for retention with a learnable map, dual-representation binding, and chunked horizon generation, enabling the first formal characterization and quantitative control of the resulting stability–plasticity trade-off.

*Online forecasting.* Online forecasters navigate stability and plasticity by altering parameter mutability. Highly plastic approaches, such as FSNet's (Pham et al., 2023) per-layer adapters and associative memory, or rehearsal methods like ER (Lin, 1992) and DER++ (Buzzega et al., 2020), suffer

substantial forgetting on high-dimensional streams (Section 3). Dual-model architectures such as OneNet (Zhang et al., 2023) improve retention by ensembling two forecasters at double the parameter cost, while LSTD (Cai et al., 2025) targets latent identifiability through an expensive variational backbone and auxiliary objectives. Conversely, CoSPOT (Choi et al., 2026) freezes a pre-trained language model and updates only an output projection; while stable, it struggles to adapt to short-horizon dynamics. Crucially, most existing methods are evaluated under the stride-1 protocol of FSNet (Pham et al., 2023) and OneNet (Zhang et al., 2023), which, as (Lau et al., 2025) show, leaks up to a forecast horizon of $H - 1$ already-trained targets into subsequent evaluations; all treat stability and plasticity as abstract goals, leaving their Pareto frontier unmapped.

*Stability and plasticity in forecasting.* In forecasting, the stability-plasticity dilemma has historically been treated as a property of the *learner* rather than the prediction mechanism. Existing methods attribute the trade-off to external components: SyReM (Lin et al., 2026) uses gradient-projected replay but assumes known scenario boundaries; Storm (Luo et al., 2025) tunes teacher distillation across segment retrainings; and Liu et al. (2025) find that temporal plasticity tracks pretraining scale, not model size, at high cost. Aligning with broader continual learning literature (Dohare et al., 2024; Lyle et al., 2023; Wang et al., 2024), these approaches rely on replay buffers, customized optimizers, or parameter scaling. In contrast, we show that stability and plasticity also stem directly from the *prediction mechanism* itself, governed by feedback iterations and step size, a relationship formalized in Section 2.8 and empirically validated in Section 3.

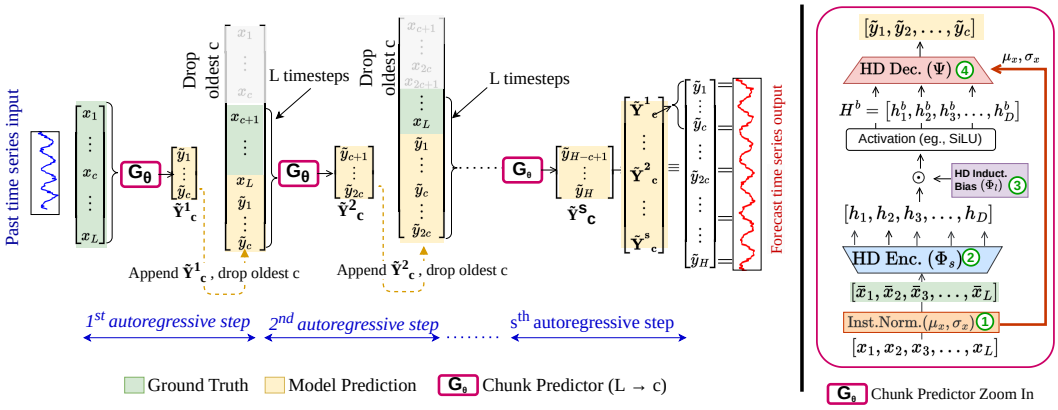



Figure 1: SPOT-VSA Architecture Overview: (left) the prediction mechanism using a hybrid autoregressive and seq2seq approach; (right) the SPOT-VSA architecture components

## 2.2 Preliminaries

Let $x = [x_1; \ldots; x_N] \in \mathbb{R}^{N \times F}$ be a time series of $N$ steps across $F$ features, with $x_t \in \mathbb{R}^F$. At step $t$, the look-back context is $X_t = [x_{t-L}; \ldots; x_{t-1}] \in \mathbb{R}^{L \times F}$ and the target horizon is $Y_t = [x_t; \ldots; x_{t+H-1}] \in \mathbb{R}^{H \times F}$. A standard forecaster parameterized by $\theta$ maps $X_t$ directly to predictions $\hat{Y}_t = G_\theta(X_t) \in \mathbb{R}^{H \times F}$.

*Chunked rollout.* A forecaster need not produce all $H$ steps in a single pass. SPOT-VSA employs a *chunk predictor* $G_\theta : \mathbb{R}^{L \times F} \to \mathbb{R}^{c \times F}$ that forecasts $c = H/s$ steps per call and is rolled out $s$ times. At rollout step $m \in \{1, \ldots, s\}$, the model outputs chunk $\tilde{Y}^m_c \in \mathbb{R}^{c \times F}$, appends it to the input window, drops the oldest $c$ observations, and predicts chunk $\tilde{Y}^{m+1}_c$. The full forecast is the concatenation $\tilde{Y}_t = [\tilde{Y}^1_c; \ldots; \tilde{Y}^s_c] \in \mathbb{R}^{H \times F}$. The chunk count $s$, termed the *granularity*, bridges direct sequence-to-sequence prediction ($s = 1$) and single-step autoregression ($s = H$).

*Offline vs. online learning.* Offline training freezes $G_\theta$ after multi-epoch prefix training. In online learning, this prefix serves as a warm-up before prequential streaming: the model forecasts $\tilde{Y}_t$ over $s$ chunks, observes $Y_t$, updates parameters via a single gradient step, and advances by $H$ ($t \leftarrow t + H$). This strict horizon-stride protocol prevents the target leakage of unit-stride evaluation identified by Lau et al. (2025). Metrics are cumulative, with retention probed periodically on fixed early windows without updates (Section 3.1).

### 2.3 PROPOSED FRAMEWORK

Figure 1 shows the SPOT−VSA architecture (right) and its chunked rollout mechanism (left). It consists of three components: an HD encoder ($\Phi_s$), a persistent HD inductive bias ($\Phi_l$) bound element-wise to the encoding, and a linear HD decoder ($\Psi$) that maps the bound representation to forecast chunks.

### 2.4 HD ENCODER $\Phi_s$: FROZEN ANCHOR BOUND TO A LEARNABLE MAP

The encoder maps a look-back window $\bar{X}_t \in \mathbb{R}^{L\times F}$ at time $t$ to hyperdimensional space via $\Phi_s : X \mapsto B \cdot X$, where $B \in \mathbb{R}^{D\times L}$ ($D > L$) yields one hypervector $H_t = [h_1, \cdots, h_D]^t$, per channel (Figure 2). Rather than using an unconstrained matrix, $B$ factorizes into the Hadamard product $B = W \odot P$, where $W \in \mathbb{R}^{D\times L}$ is a learnable weight matrix and $P \in \mathbb{R}^{D\times L}$ is a frozen random anchor whose columns $P_i \sim \mathcal{N}(0, \sigma_i^2 I_D)$ are sampled once at initialization. This factorized projection balances stability and plasticity:

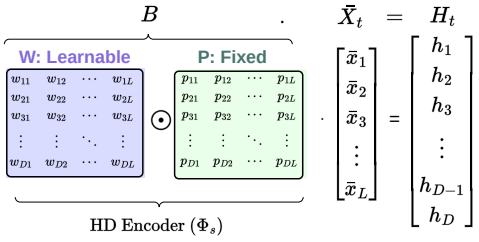



Figure 2: HD Encoder ($\Phi_s$) Architecture

**$P$ acts as the stability anchor.** Being immutable, $P$ insulates input-space geometry from online updates. Independent high-dimensional rows ensure the $L$ lag hypervectors remain quasi-orthogonal with pairwise cosine similarities of $\mathcal{O}(1/\sqrt{D})$ (Appendix A.1). Inter-lag interference is bounded at this scale, matching the retention of large frozen backbones (e.g., CoSPOT) via a compact $L \times D$ projection without freezing deep layers.

**$W$ provides plastic adaptation.** Frozen random projections cannot track non-stationary inputs (Duan et al., 2022); the learnable map $W$ adds online adaptivity while $W \odot P$ stays anchored to the correlation-preserving $P$, similar to parametrized random projections (Johnson & Lindenstrauss, 1984). Hadamard binding preserves coordinate independence: each $[W \odot P]_{ij} \sim \mathcal{N}(0, w_{ij}^2\sigma_i^2)$ conditioned on $W$ (Lemma A.2, Appendix A.1), reweighting coordinates without creating spurious cross-entry correlations.

**Proposition 2.1** (Correlation Preservation)**.** *Let $m, n \in [\![1, F]\!]$ be two integers, and $\boldsymbol{x}_m, \boldsymbol{x}_n \in \mathbb{R}^{L\times 1}$ denote $m$-th and $n$-th channels of $X_t \in \mathbb{R}^{L\times F}$. Furthermore, define $H_t = \Phi_s(X_t) = (W \odot P)X_t \in \mathbb{R}^{D\times F}$, where $W$ has fixed columns $w_i \in \mathbb{R}^D$ and $P$ has independent columns $p_i \sim \mathcal{N}(0, \sigma_i^2 I_D)$ for $i = 1, \ldots, L$. We also have $\boldsymbol{h}_m = \Phi_s(\boldsymbol{x}_m)$ and $\boldsymbol{h}_n = \Phi_s(\boldsymbol{x}_n)$. Assume that for each lag $i$ :*

$$\lim_{D\to\infty} \frac{1}{D}\|w_i\|_2^2 = \bar{w}_i^2 < \infty.$$

*Then, as $D \to \infty$, the uncentered cosine similarity satisfies*

$$\rho(\boldsymbol{h}_m, \boldsymbol{h}_n) \xrightarrow{p} \frac{\boldsymbol{x}_m^\top \Sigma_W \boldsymbol{x}_n}{\sqrt{\boldsymbol{x}_m^\top \Sigma_W \boldsymbol{x}_m}\sqrt{\boldsymbol{x}_n^\top \Sigma_W \boldsymbol{x}_n}},$$

*where $\Sigma_W = \operatorname{diag}\big(\sigma_1^2\bar{w}_1^2, \ldots, \sigma_L^2\bar{w}_L^2\big)$. In particular, if the lag scaling $\sigma_i\bar{w}_i$ is uniform across all lags $i$, then $\rho(\mathbf{h}_m, \mathbf{h}_n) \xrightarrow{p} \rho(\mathbf{x}_m, \mathbf{x}_n)$ identically.*

Proposition 2.1 shows that $P$ preserves channel geometry at initialization, while $W$ provides bounded plasticity through the lag reweighting $\Sigma_W$; after training the division is empirical (Remark A.5) and is supported by the ablations of Appendix A.6.5, where making $P$ trainable induces forgetting and removing the binding ($P = \mathbf{1}$) reduces accuracy. The latent size $D$ is not a stability knob: forgetting decreases with $D$ on some streams and increases on others (Appendix A.6.6).

### 2.5 HD INDUCTIVE BIAS $\Phi_l$

Unlike prior online models that use associative memory buffers to retain long-term information Pham et al. (2023), we adopt an input-independent relational inductive bias motivated by abstract reasoning architectures and CLS theory Altabaa et al. (2023); Webb et al. (2020); Mejri et al.

(2024b). We define this bias as a persistent matrix $\Phi_l \in \mathbb{R}^{D\times F}$, carried across windows and excluded from weight decay. $\Phi_l$ is bound element-wise via Hadamard product to the encoder output $\Phi_s(\bar{X})$, gating which latent directions propagate to the decoder. The generated output is termed $H^b \in \mathbb{R}^{D\times F}$ so that $H^b = \Phi_s(\bar{X}) \odot \Phi_l$

### 2.6 HD Decoder $\Psi$

As shown in Figure 3, $H^b$ is first forwarded to a `SiLU` activation. The decoder maps the bound representation to a forecast chunk of $c$ steps per channel via a linear projection. The generated matrix is referred to as $Z \in \mathbb{R}^{c\times F}$: $Z = W_\Psi^\top H^b$, where $W_\Psi \in \mathbb{R}^{D\times c}$ is a trainable matrix. Adjusting the chunk size $c$ controls rollout granularity: direct sequence-to-sequence ($c = H$), autoregression ($c = 1$), or chunked rollouts ($c = H/s$; Section 2.7).

*Channel mixing.* For correlated streams, a learnable matrix $M \in \mathbb{R}^{F\times F}$ (initialized to $I$) couples forecasts via $\tilde{Y} \mapsto \tilde{Y}M$. Operating on output predictions rather than latents.

*Instance normalization.* Presented in Kim et al. (2021), each window is normalized per channel as $\bar{X} = (X - \mu_x)/\sigma_x$ and de-normalized as $\tilde{Y} \mapsto \tilde{Y}\sigma_x + \mu_x$ without learnable parameters ($\sigma_x = \mathrm{Std}(X)$). Selecting the anchor $\mu_x$ as the last observed value fits trending series, whereas the window mean suits mean-reverting streams; both maintain rollout scale consistency and preserve inter-channel correlations (Proposition 2.1).

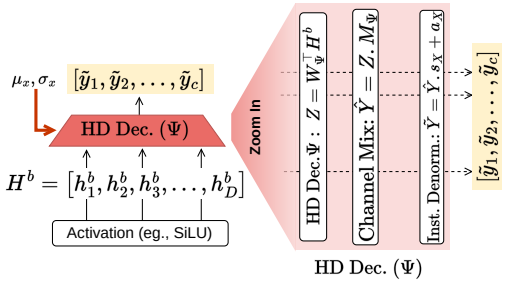



Figure 3: HD decoder components $\Psi$

### 2.7 Online Learning Mechanism

Algorithm 1 summarizes end-to-end training of `SPOT-VSA`. An offline warm-up (lines 11–14) sweeps the training prefix for $E$ epochs at stride 1. The online phase (lines 15–18) then traverses the stream with stride $H$, so each target is revealed only after being forecast: no label leakage and no replay. At each step, `ChunkedPass` first runs with *learn* = FALSE (line 16) to produce $\tilde{Y}_t$ without touching the parameters; after $Y_t$ is observed and scored, it runs once more with *learn* = TRUE (line 18), teacher-forcing the rollout on $Y_t$ and applying one AdamW step per chunk, where $\mathcal{L}_s$ updates $(\mathbf{\Phi}_s, \mathbf{\Psi})$ and $\mathcal{L}_l$ updates $\mathbf{\Phi}_l$ alone. Warm-up and online adaptation share this same pass, differing only in step size ($\eta_0$ vs. $\eta$).

### 2.8 Theoretical Analysis: Stability and Plasticity of Chunked Forecasting

While the stability–plasticity dilemma is often invoked conceptually, we establish an analytical framework to make it predictive for chunked online forecasters. We define both properties operationally, bound each with respect to the three operational knobs $(H, s, \eta)$, and identify when they conflict.

**Operational Formulation:** The chunk predictor $G_\theta : \mathbb{R}^L \to \mathbb{R}^c$ emits $c = H/s$ timesteps per call and is called $s$ times. At step $k \in [\![1, s]\!]$, let $y^{(k)}$ be the ground-truth chunk, $\hat{y}^{(k)} = G_\theta(u_k)$ the prediction from context $u_k$ that is defined as $u_1 = X_t$, $u_{k+1} \triangleq [\,u_k[c{+}1{:}L];\, \hat{y}^{(k)}\,]$ (append the generated chunk, drop the $c$ oldest rows), so $u_k$ holds $k{-}1$ generated chunks; $u_k^{\text{true}}$ uses $y^{(k)}$ instead. We also define $a_k = \|\hat{y}^{(k)} - y^{(k)}\|$ the chunk error. We quantify **plasticity (tracking error)** as the average prequential rollout error over the horizon: $\mathcal{P} \triangleq \frac{1}{s}\sum_{k=1}^{s} a_k$, where lower values correspond to faster, more faithful signal tracking. Conversely, because parameters can shift along flat loss basins without altering predictions, we measure **stability (function-space displacement)** by the functional shift $\Delta \triangleq \mathbb{E}_{x\sim\mathcal{D}_{\text{ref}}}\|G_{\theta_{t+1}}(x) - G_{\theta_t}(x)\|$, the expected change in forecasts on a fixed reference set of past contexts $\mathcal{D}_{\text{ref}}$ caused by one online update; $\Delta = 0$ means earlier inputs are forecast exactly as before, which is what the probe-based forgetting ratio (Section 3.1) records over the stream. Under gradient descent with step size $\eta$, this displacement satisfies $\Delta \le \eta G_x \|g_t\|$, where $g_t$ denotes the rollout loss gradient (Appendix A.2). Error and gradient compounding across rollout steps are governed by the geometric factors $\Phi_p(s) = \frac{\gamma^s - 1}{s(\gamma - 1)}$ and $\Phi_\sigma(s) = \frac{\gamma^{2s} - 1}{s(\gamma^2 - 1)}$ (with $\gamma = 1 + \Lambda$

**Algorithm 1:** SPOT-VSA: warm-up (orange), chunked prediction (blue), and update (green) via $G_\theta$ rollout.

**Require :** Horizon $H$, granularity $s$ (chunk size $c = H/s$), look-back $L$, warm-up epochs $E$, step sizes $\eta_0$ (warm-up), $\eta$ (online), weight decay $\lambda$, anchor $\mu_x \in \{\text{last}, \text{mean}\}$

**Init :** $\theta = (\mathbf{\Phi}_s, \mathbf{\Psi}, \mathbf{\Phi}_l)$; AdamW$(\eta_0, \lambda)$

**Notation:** $Y_c^m \in \mathbb{R}^{c \times F}$ is the $m$-th chunk of $Y \in \mathbb{R}^{H \times F}$, so $Y = [Y_c^1; \ldots; Y_c^s]$.

**Procedure** ChunkedPass($X$, $Y$, learn)

/* — Phase 1: chunked rollout (Fig. 1) — */

Roll $G_\theta$ out for $m = 1, \ldots, s$: normalize the context with anchor $\mu_x$, predict $\tilde{Y}_c^m = \sigma_x\, G_\theta(\bar{U}_m) + \mu_x$, and shift the context by $c$ steps using $Y_c^m$ if *learn* (teacher forcing) or $\tilde{Y}_c^m$ otherwise (free running)

$\tilde{Y} \leftarrow [\tilde{Y}_c^1; \ldots; \tilde{Y}_c^s]$ // $G_\theta = \Psi\big(\Phi_s(\cdot) \odot \Phi_l\big)$

/* — Phase 2: update (only if learn = TRUE) — */

**if** learn **then**

  **for** $m \leftarrow 1$ ***to*** $s$ **do**

    $\mathcal{L}_s \leftarrow \text{MSE}(\tilde{Y}_c^m, Y_c^m)$ // updates $\mathbf{\Phi}_s, \mathbf{\Psi}$

    $\mathcal{L}_l \leftarrow \text{Loss}_{\text{long}}(\tilde{Y}_c^m, Y_c^m)$ // updates $\mathbf{\Phi}_l$ only

    $g \leftarrow \big(\nabla_{\mathbf{\Phi}_s, \mathbf{\Psi}} \mathcal{L}_s,\ \nabla_{\mathbf{\Phi}_l} \mathcal{L}_l\big)$; $\theta \leftarrow \text{AdamW}(\theta, g)$

**return** $\tilde{Y}$

**Algorithm** SPOT-VSA

/* Warm-up (offline, stride 1) */

**for** $e \leftarrow 1$ ***to*** $E$ **do**

  **for** $t \leftarrow L + 1$ ***to*** $N_{train} - H + 1$ **do**

    ChunkedPass($X_t$, $Y_t$, TRUE)

Set AdamW step size to $\eta$

/* Online (prequential stream, stride $H$) */

**for** $t \leftarrow L + 1$ ***to*** $N_{test} - H + 1$ ***step*** $H$ **do**

  $\tilde{Y}_t \leftarrow$ ChunkedPass($X_t$, $\emptyset$, FALSE) // forecast, no gradients

  Observe $Y_t$; Score$(\tilde{Y}_t, Y_t)$

  ChunkedPass($X_t$, $Y_t$, TRUE) // single adaptation pass, $n_{\text{inner}} = 1$

for a $\Lambda$-Lipschitz predictor $G_\theta$; any network with Lipschitz activations is Lipschitz with $\Lambda$ bounded by the product of its layers' spectral norms (Virmaux & Scaman, 2018), and this is the standard assumption under which rollout error compounds geometrically (Venkatraman et al., 2015)), both of which grow monotonically in $s$ at asymptotic rates $\mathcal{O}(\gamma^s/s)$ and $\mathcal{O}(\gamma^{2s}/s)$, respectively.

**Granularity ($s$) under Stationarity: The Single-Chunk vs. Compounding Tradeoff** In stationary streams (drift $q = 0$), the primary calibration knob is the chunk count $s$:

**Theorem 2.2** (Granularity Bounds under Stationarity)**.** *Under standard Lipschitz conditions (Assumption A.6, Appendix A.2), for any parameters $\theta$ and any divisor $s$ of horizon $H$, the plasticity error satisfies $\mathcal{P} \le \delta(H/s)\, \Phi_p(s)$ and the stability shift satisfies $\Delta \le \eta\, G_x G\, \delta(H/s)\, \Phi_\sigma(s)$, where $\delta(c)$ bounds the intrinsic prediction error on a clean single chunk of length $c$.*

Theorem 2.2 exposes the balance between *chunk difficulty* $\delta(H/s)$ (which decreases as chunks size $(H/s)$ shrinks) and *rollout compounding* $\Phi(s)$ (which rises with $s$) multiplying both leads to non-monotonic behavior (an optimum exists):

*Optimal intermediate granularity ($s^\star$):* Compounding eventually dominates, causing both bounds to grow unbounded for large $s$. However, when single-chunk difficulty falls rapidly with shorter targets ($\delta(c) \propto c^\alpha$ with $\alpha > \log_2(\frac{1+\gamma}{2})$), an interior optimum $s^\star > 1$ emerges where moderate chunking beats both single-shot ($s = 1$) and full autoregression ($s = H$).

*Asymmetric degradation:* For free-running updates, stability degrades faster than plasticity ($\mathcal{O}(\gamma^{2s})$ vs. $\mathcal{O}(\gamma^{s})$; teacher forcing: Remark A.7).

*Coupled under stationarity:* Because $\Phi_\sigma \le \gamma^{s-1}\Phi_p$, parameter shifts under stationarity stem from rollout error, so reducing it (e.g., via $\eta$) improves stability and plasticity together.

**Step Size ($\eta$) and the Emergence of the Drift Tradeoff** When distributions drift ($q = \mathbb{E}\|\theta^\star_{t+1} - \theta^\star_t\|^2 > 0$), parameter updates must track a moving minimizer against effective noise $\sigma^2_{\text{eff}} = \sigma^2 + \Pi(H, s)$, where $\Pi(H, s)$ denotes rollout compounding noise (formal formulation in Appendix A.2). We define $e_t = \theta_t - \theta^\star_t$ as the tracking error in parameter space: the difference between where the online learner's parameters actually are at time $t$ ($\theta_t$) and where they should be ($\theta^\star_t$, the minimizer of the current, drifting loss). With a quadratic local loss, one gradient step with noisy gradient $e_t - \epsilon_t$ followed by a drift $\omega_t$ of the optimum gives $e_{t+1} = (1-\eta)e_t + \eta\epsilon_t - \omega_t$ (noise $\epsilon_t$, drift $\omega_t$), and $V_\infty(\eta) \triangleq \lim_{t\to\infty} \mathbb{E}\|e_t\|^2$ is the steady-state variance of this gap.

**Theorem 2.3** (Tradeoff under Drift; Informal)**.** *Under linear tracking dynamics with drift energy $q > 0$, the plasticity error satisfies $\mathcal{P} \le \big(\delta(H/s) + G_x\sqrt{V_\infty(\eta)}\big)\Phi_p(s)$ and the stability shift satisfies $\Delta \le \eta\, G_x G\big(\delta(H/s) + G_x\sqrt{V_\infty(\eta)}\big)\Phi_\sigma(s)$, with $V_\infty(\eta) = \frac{\eta}{2-\eta}\sigma^2_{\text{eff}} + \frac{q}{\eta(2-\eta)}$ decomposing into a noise-chasing term and a lag term.*

Theorem 2.3 reveals the exact origin of the dilemma: *Conflicting objectives in $\eta$:* Plasticity requires an intermediate step size $\eta^\star = \frac{2\sqrt{q}}{\sqrt{q}+\sqrt{q+4\sigma^2_{\text{eff}}}} \in (0, 1)$ to minimize tracking lag without overfitting noise. In contrast, the stability bound is strictly increasing in $\eta$, favoring $\eta \to 0$. *Tradeoff disappears when $q = 0$:* When drift vanishes ($q = 0$), the lag term vanishes, $V_\infty(\eta)$ becomes monotonically increasing in $\eta$, and both criteria favor minimal step sizes.

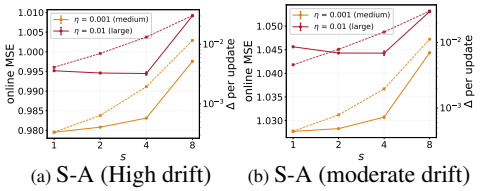
**Empirical Predictions.** Table 1 states the bounds as directional predictions, where an *interior* optimum lies strictly inside a knob's range. Under stationarity, plasticity has an interior optimum in $s$ when $\delta(c) \propto c^\alpha$ with $\alpha > \log_2(\frac{1+\gamma}{2})$ (some $1 < s^\star < H$ beats both $s = 1$ and $s = H$), while stability degrades monotonically at rate $\mathcal{O}(\gamma^{2s})$ (P1); longer horizons hurt both (P2). Without drift, $V_\infty(\eta)$ is increasing and both favor $\eta \to 0$ (P3). Under drift, the lag term $q/(\eta(2-\eta))$ diverges as $\eta \to 0$ while the noise term grows with $\eta$, giving plasticity an interior $\eta^\star$ that the monotone stability bound lacks (P4). Since stability is monotone in $\eta$, a step-size tradeoff arises only under drift.

| | | Plasticity | | Stability | |
|---|---|---|---|---|---|
| | Knob | Property | MSE/RMSE | Property | FR |
| P1 | $s \uparrow$ | peak at $s^\star$ | min at $s^\star$ | $\downarrow^\dagger$ | $\uparrow^\dagger$ ($\mathcal{O}(\gamma^{2s})$) |
| P2 | $H \uparrow$ | $\downarrow$ | $\uparrow$ | $\downarrow$ | $\uparrow$ |
| P3 | $\eta \uparrow, q = 0$ | $\downarrow$ | $\uparrow$ | $\downarrow$ | $\uparrow$ |
| P4 | $\eta \uparrow, q > 0$ | peak at $\eta^\star$ | U-shaped ($\eta^\star$) | $\downarrow$ | $\uparrow$ |

Table 1: Predictions of Thms. 2.2–2.3 (validated in §3.2); plasticity/stability $\uparrow$ as MSE/FR $\downarrow$. $^\dagger$Free-running; teacher-forced: $\propto s^{1-\alpha}$ (Rem. A.7).

## 3 Experiments

We evaluate SPOT-VSA to verify: (i) faster adaptation to streaming non-stationarity over online and static baselines; (ii) empirical validation of stability–plasticity bounds via chunked autoregression; and (iii) edge-viable efficiency. Experimental details, datasets, and baselines are in Appendix A.4.




(a) S-A (High drift) (b) S-A (moderate drift)

Figure 4: **Granularity sweep along $s$.** MSE (plasticity, left axis) and stability (right axis): functional displacement $\Delta$ on the synthetic streams (a, b).

### 3.1 Experimental Setup

**Datasets and strict protocol.** We benchmark on eight multivariate streams: **ETT** (4 subsets) (Zhou et al., 2021), **ECL**, **Traffic**, **Weather**, and **Exchange** (split 20%/5%/75% for warm-up, validation, and streaming)—alongside synthetic abrupt (S-A) and gradual (S-G) drift streams (Pham et al., 2023) (Appendix A.4.1).

Motivated by the leakage analysis of Lau et al. (2025) and the delayed-feedback setting of Zhang et al. (2023), where models are updated once every $H$ rounds, we enforce a **horizon-strided proto-**

**col** (stride $= H \in \{24, 168, 336\}$), as in concurrent work (Fujimoto & Nishi, 2026): at step $t$, the model predicts $Y_t \in \mathbb{R}^{H \times F}$ from $X_t$, logs the error, updates once ($n_{\text{inner}} = 1$), and advances by $H$.

**Baselines and metrics.** We compare against:(1) continual forecasters: FSNet (Pham et al., 2023), OneNet/OneNet-TCN (Zhang et al., 2023), LSTD (Cai et al., 2025), and DSOF (Lau et al., 2025); (2) replay: ER and DER++ (Buzzega et al., 2020); and (3) prompt-tuned LLM: CoSPOT (Choi et al., 2026). Errors are tracked via stream MSE/RMSE/MAE. Retention uses cached initial probe windows via *Forgetting Ratio* ($\text{FR} = \frac{p_{\text{final}}}{\min_t p_t}$, $p_t$: probe MSE at checkpoint $t$; 1 = none), a scale-free form of the forgetting measure of Chaudhry et al. (2018).

Table 2: Online RMSE and MAE (stride $H$, mean of 4 seeds; std in App. A.6). Best: **green**, second: **blue**.

| Metric | | RMSE | | | | | | | | | MAE | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | H | FSNet | OneNet | OneNet-T | ER | DER++ | LSTD | CoSPOT | DSOF | **Ours** | FSNet | OneNet | OneNet-T | ER | DER++ | LSTD | CoSPOT | DSOF | **Ours** |
| ETTh1 | 24 | 0.926 | 0.903 | **0.796** | 0.854 | 0.849 | 0.801 | 1.325 | 0.800 | **0.747** | 0.642 | 0.637 | 0.544 | 0.592 | 0.584 | 0.552 | 0.879 | **0.533** | **0.487** |
| | 168 | 1.540 | 1.184 | 1.135 | 1.332 | 1.324 | 1.234 | 1.388 | **1.059** | **1.026** | 1.128 | 0.834 | 0.808 | 0.994 | 0.984 | 0.908 | 0.937 | **0.716** | **0.696** |
| | 336 | 2.251 | 1.470 | 1.388 | 1.747 | 1.746 | 1.434 | **1.376** | **1.141** | **1.141** | 1.744 | 1.095 | 1.036 | 1.332 | 1.321 | 1.073 | 0.941 | **0.801** | **0.803** |
| ETTh2 | 24 | 1.398 | 1.062 | 1.127 | 1.342 | 1.330 | 1.144 | 1.139 | **0.868** | **0.832** | 0.979 | 0.731 | 0.820 | 0.961 | 0.953 | 0.834 | 0.780 | **0.599** | **0.555** |
| | 168 | 3.351 | 2.372 | 2.811 | 2.989 | 2.970 | 2.858 | 1.880 | **1.760** | **1.714** | 2.383 | 1.605 | 1.913 | 2.063 | 2.041 | 1.968 | 1.174 | **1.133** | **1.049** |
| | 336 | 4.524 | 3.812 | 3.268 | 3.690 | 3.706 | 3.433 | **2.428** | 3.061 | **2.893** | 3.330 | 2.710 | 2.185 | 2.522 | 2.534 | 2.289 | **1.424** | 1.776 | **1.610** |
| ETTm1 | 24 | 0.582 | **0.536** | 0.545 | 0.572 | 0.554 | 0.613 | 0.973 | 0.632 | **0.485** | 0.460 | **0.421** | 0.429 | 0.453 | 0.437 | 0.489 | 0.680 | 0.492 | **0.370** |
| | 168 | 1.033 | 1.056 | **0.840** | 0.947 | 0.933 | 0.922 | 1.145 | 0.926 | **0.744** | 0.815 | 0.815 | **0.647** | 0.735 | 0.725 | 0.717 | 0.767 | 0.704 | **0.553** |
| | 336 | 2.156 | 1.175 | **1.026** | 1.247 | 1.220 | 1.133 | 1.267 | 1.197 | **0.901** | 1.774 | 0.904 | **0.775** | 0.964 | 0.946 | 0.882 | 0.862 | 0.869 | **0.659** |
| ETTm2 | 24 | 0.827 | **0.768** | 0.794 | 0.799 | 0.790 | 0.804 | 0.900 | 0.886 | **0.781** | 0.560 | **0.510** | 0.534 | 0.542 | 0.534 | 0.548 | 0.630 | 0.592 | **0.518** |
| | 168 | 1.252 | 1.071 | **1.012** | 1.139 | 1.127 | 1.056 | 1.087 | 1.083 | **1.034** | 0.922 | 0.758 | **0.707** | 0.822 | 0.809 | 0.740 | 0.739 | 0.733 | **0.690** |
| | 336 | 1.572 | **1.171** | 1.189 | 1.332 | 1.325 | **1.155** | 1.356 | 1.334 | 1.896 | 1.202 | 0.856 | **0.838** | 0.989 | 0.981 | **0.835** | 0.912 | 0.915 | 1.428 |
| WTH | 24 | **0.547** | **0.551** | 0.554 | 0.566 | 0.562 | **0.547** | 0.663 | 0.601 | 0.554 | 0.392 | 0.391 | 0.396 | 0.410 | 0.405 | **0.390** | 0.481 | 0.431 | **0.381** |
| | 168 | 0.774 | 0.714 | 0.753 | 0.764 | 0.752 | **0.699** | 0.774 | 0.744 | **0.709** | 0.595 | 0.542 | 0.569 | 0.581 | 0.570 | **0.523** | 0.585 | 0.561 | **0.522** |
| | 336 | 0.862 | **0.752** | 0.987 | 0.904 | 0.834 | **0.741** | 0.776 | 0.810 | 0.844 | 0.679 | **0.575** | 0.774 | 0.706 | 0.642 | **0.565** | 0.588 | 0.613 | 0.642 |
| Exchange | 24 | 1.338 | 0.754 | 0.812 | 1.002 | 0.997 | 0.303 | 0.419 | **0.296** | **0.298** | 1.082 | 0.592 | 0.605 | 0.810 | 0.806 | 0.219 | 0.298 | **0.208** | **0.205** |
| | 168 | 2.221 | 1.393 | 1.855 | 2.156 | 2.148 | 1.093 | **0.948** | **0.881** | 0.952 | 1.812 | 1.117 | 1.481 | 1.750 | 1.748 | 0.835 | 0.681 | **0.640** | **0.641** |
| | 336 | 2.395 | 1.761 | 1.945 | 2.433 | 2.427 | 1.641 | **1.332** | 1.758 | **1.300** | 1.919 | 1.365 | 1.551 | 1.962 | 1.956 | 1.205 | **0.959** | 1.186 | **0.896** |
| ECL | 24 | 2.492 | 2.457 | 3.059 | 2.293 | 2.272 | **1.971** | 2.976 | 2.075 | **1.996** | 0.820 | 0.571 | 0.709 | 0.611 | 0.605 | 0.541 | 0.521 | **0.395** | **0.429** |
| | 168 | 8.095 | 4.643 | 4.073 | 6.629 | 6.678 | 4.618 | **3.172** | **3.199** | 3.320 | 2.229 | 0.898 | 0.774 | 1.669 | 1.583 | 1.565 | **0.503** | **0.515** | 0.596 |
| | 336 | 13.9 | 7.266 | 5.583 | 10.1 | 10.4 | 6.101 | **3.329** | **3.843** | 3.938 | 2.419 | 0.996 | 1.185 | 2.991 | 2.860 | 2.530 | **0.514** | **0.580** | 0.881 |
| Traffic | 24 | 0.620 | 0.674 | 0.703 | **0.598** | **0.597** | 0.696 | 0.924 | 0.691 | 0.685 | 0.322 | 0.363 | 0.376 | **0.296** | **0.294** | 0.384 | 0.598 | 0.364 | 0.391 |
| | 168 | 0.732 | 0.725 | 0.692 | 0.723 | 0.722 | 0.686 | 0.869 | **0.685** | **0.632** | 0.405 | 0.434 | 0.377 | 0.406 | 0.402 | 0.360 | 0.542 | **0.359** | **0.342** |

Table 3: Forgetting ratio (FR; ↓; mean of 4 seeds; mean±std in App. A.6). Best: **green**, second: **blue**.

| Method | ETTh1 | | | ETTh2 | | | ETTm1 | | | ETTm2 | | | WTH | | | Exchange | | | ECL | | | Traffic | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 |
| FSNet | 1.25 | 1.17 | 1.25 | 1.80 | 1.74 | 1.23 | 1.41 | 1.40 | 1.07 | 1.13 | 1.31 | 1.12 | 1.38 | 1.12 | **1.05** | 9.71 | 1.92 | 1.24 | 104.77 | 76.63 | 102.66 | 1.80 | 1.09 |
| OneNet | 1.15 | **1.01** | 1.25 | 1.12 | 1.13 | 1.17 | 1.12 | 1.23 | 1.31 | 1.23 | 1.26 | 1.19 | 1.15 | 1.65 | 1.35 | 7.67 | 2.36 | 1.30 | 1.45 | 39.79 | 21.28 | 1.10 | **1.01** |
| OneNet-TCN | 1.15 | 1.31 | **1.02** | 1.89 | 1.29 | 1.25 | 1.86 | 1.25 | 1.13 | 1.56 | 1.11 | 1.07 | 1.04 | 1.20 | 1.38 | 2.88 | 1.49 | 1.09 | 7.31 | 1.09 | **1.06** | **1.01** | **1.00** |
| ER | 1.13 | 1.38 | 1.04 | 1.27 | 2.21 | **1.01** | 1.09 | **1.00** | **1.06** | 1.21 | 1.27 | **1.05** | **1.02** | **1.01** | 1.09 | 1.28 | 1.51 | **1.00** | 1.72 | **1.02** | 1.21 | 1.41 | 1.06 |
| DER++ | 1.20 | 1.43 | 1.20 | 1.21 | 1.88 | **1.01** | 1.11 | **1.00** | 1.08 | 1.21 | 1.14 | 1.08 | 1.03 | 1.04 | 1.09 | 1.19 | 1.41 | **1.00** | **1.25** | 1.13 | 1.67 | 1.29 | **1.01** |
| LSTD | 1.14 | 1.09 | 1.07 | 1.36 | 3.88 | 7.73 | 3.65 | 1.86 | 1.35 | 1.02 | 1.21 | 1.09 | **1.00** | 1.51 | 1.43 | **1.01** | 1.75 | 3.59 | 1.33 | 1.04 | **1.00** | **1.03** | **1.00** |
| CoSPOT | **1.06** | **1.00** | **1.00** | **1.01** | **1.00** | **1.00** | **1.03** | **1.00** | **1.00** | **1.01** | **1.00** | **1.00** | 1.08 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.05** | **1.00** | **1.00** | 1.09 | **1.01** |
| DSOF | 1.44 | 1.06 | 1.41 | 1.23 | **1.10** | 1.31 | 3.35 | 2.39 | 1.63 | 2.51 | 2.74 | 1.34 | 1.04 | 1.26 | 1.82 | 1.11 | **1.10** | 2.99 | 2.69 | 3.00 | 2.33 | 2.27 | 1.85 |
| SPOT-VSA | **1.08** | 1.02 | 1.04 | **1.10** | 1.13 | **1.00** | **1.05** | **1.11** | **1.00** | **1.00** | **1.06** | **1.00** | **1.00** | **1.00** | **1.00** | 1.26 | 1.13 | **1.03** | 1.59 | 1.20 | 1.22 | 1.06 | 1.02 |

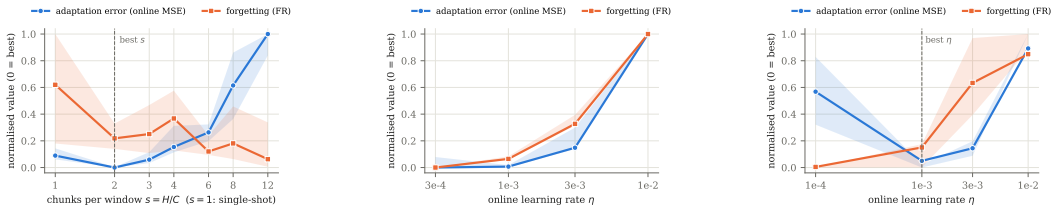
## 3.2 Plasticity–Stability Comparison

Table 4: Compute Overhead Eval. ($H$=168)

| Method | Data | Params (M) | Jetson Nano (GPU) ms | Jetson Nano (GPU) mJ | ARM-Cortex (CPU) ms | ARM-Cortex (CPU) mJ |
|---|---|---|---|---|---|---|
| SPOT-VSA | Exch. | **0.17** | **17.1** | **71** | **2.4** | **28** |
| FSNet | Exch. | 1.7 | 362.5 | 1,460 | 538.3 | 6,056 |
| OneNet | Exch. | 3.1 | 721.0 | 2,871 | 969.7 | 10,910 |
| LSTD | Exch. | 1.7 | 513.1 | 2,034 | 435.1 | 4,891 |
| SPOT-VSA | ECL | **0.43** | **23.6** | **159** | **32.6** | **393** |
| FSNet | ECL | 18.6 | 437.0 | 2,172 | 1163.7 | 13,539 |
| OneNet | ECL | 20.0 | 812.5 | 3,755 | 1895.2 | 21,799 |
| LSTD | ECL | 4.8 | 507.6 | 2,211 | 686.2 | 7,805 |

Tables 2–3 set $s$=1, matching baselines' output setup to isolate design gains. SPOT-VSA ranks top-two in online RMSE/MAE across most setups (Table 2), topping FSNet, OneNet, ER/DER++, and LSTD. DSOF is the best recent SOTA, close on ETTh1/ETTh2, Exchange, and ECL, yet SPOT-VSA still leads on most cases (all ETTh2 and ETTm1 steps). Its FR is near 1 (Table 3), while FSNet/OneNet and replay drop sharply on ECL/Exchange, and DSOF's accuracy incurs heavy forgetting (FR up to 3.35 on ETTm1 and 2.3–3 on ECL). CoSPOT forgets the least via its GPT-2 base (∼82M params) but has worse short-step error; so, SPOT-VSA yields the best stability-plasticity trade-off with fewer parameters.

**Plasticity Stability Tradeoff Evaluation** Figure 4 sweeps $s$ on a high- and a low-drift stream at two learning rates. $\Delta$ grows monotonically with $s$ at every $\eta$ (P1), whereas MSE shows a shallow interior minimum only at the larger $\eta$: with near-constant chunk difficulty ($\alpha \approx 0$) the bound




(a) P1: chunk count $s$ (real streams) (b) P3: learning rate $\eta$, no drift ($q$=0, AR(1)) (c) P4: learning rate $\eta$, drift ($q$>0, real streams)

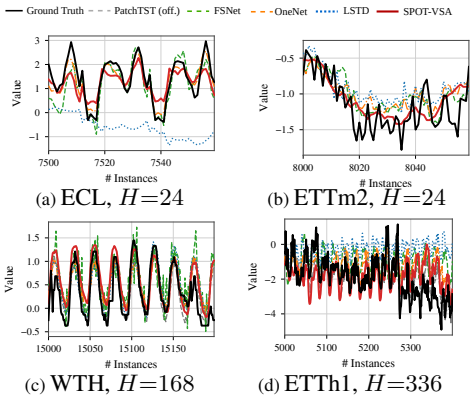
Figure 5: Stability (FR)/plasticity (MSE) across Table 1 knobs; min–max normalized, data-horizon averaged.

predicts $s^\star$=1, and the interior optimum at large $\eta$ arises because $s$ sub-updates per window damp the effective step size, an effect beyond our worst-case analysis (App. A.7.4).

Sweeping $s$ (Fig. 5a) yields a U-shaped adaptation error with an optimum at $s$=2, while forgetting decreases quasi-monotonically (bump at $s$=4 from heterogeneous pairs), as the teacher-forced bound $\Delta \propto s^{1-\alpha}$ predicts for $\alpha > 1$ (Remark A.7). Raising $\eta$ degrades both monotonically on a stationary AR(1) stream (P3, Fig. 5b), whereas on the 8 drifting real streams, adaptation error becomes U-shaped with an optimum near $10^{-3}$ while forgetting still rises (P4, Fig. 5c).

### 3.2.1 Pointwise Stream Predictions

**Visual Assessment** Figure 6 illustrates behavior from Tables 2 and 3. SPOT−VSA tracks levels and envelopes where SOTA fails: avoiding LSTD's bias on ECL (a), reducing lag under downward drift on ETTm2 (b), preventing FSNet's overshoot on periodic WTH (c), and adapting within one window to ETTh1's shift (d). Overall, it smooths fine details rather than missing levels.




(a) ECL, $H$=24 (b) ETTm2, $H$=24 (c) WTH, $H$=168 (d) ETTh1, $H$=336

Figure 6: Visual evaluation (pred Vs GT)

### 3.2.2 Ablation and Edge Efficiency

**Computational efficiency.** On Jetson Nano, SPOT−VSA is 18–42× faster and 14–40× more energy-efficient than the baselines, while using 10–46× fewer parameters. On Edge CPU, the gains reach up to 404× in speed and 390× in energy efficiency, with a minimum of 21× and 20×, respectively (Table 4). These gains follow from replacing deep, sequential kernels with depth-one parallel matrix operations of cost $\mathcal{O}(FD(T+H+F))$.

Table 5: Ablations (MSE). (a) Components. (b) $\Phi_l$

| Cell | Base | w/o $P$ | $s$=2 | $s$=4 | $s$=$H$ |
|---|---|---|---|---|---|
| ETTh1-24 | 0.677 | 0.682 | 0.698 | 0.733 | 1.290 |
| ETTm1-24 | 0.337 | 0.357 | 0.343 | 0.399 | 1.341 |
| ETTh1-168 | 1.238 | 1.274 | 1.239 | 1.310 | 5.535 |
| S-A-24 | 1.240 | 1.241 | 1.241 | 1.243 | 1.545 |
| S-G-24 | 1.246 | 1.246 | 1.246 | 1.250 | 1.577 |

(a)

| Cell | MSE w/ $\Phi_l$ | MSE w/o $\Phi_l$ | FR w/ $\Phi_l$ | FR w/o $\Phi_l$ |
|---|---|---|---|---|
| Exchange-1 | **0.0106** | 0.0107 | 1.09 | **1.08** |
| ETTm1-1 | **0.0825** | 0.0849 | 1.03 | **1.02** |
| ECL-1 | **1.958** | 2.045 | **1.50** | 1.52 |
| ETTm1-24 | **0.512** | 0.558 | **1.054** | 1.202 |
| ETTh1-168 | **1.077** | 1.083 | **1.016** | 1.016 |

(b)

**Architectural ablations.** Table 5a isolates the key components. Removing HD binding ($P$=**1**) degrades MSE on all real streams while leaving the synthetic ones unchanged, consistent with a regularizing effect of the frozen projection. Full autoregression ($s$=$H$) inflates MSE by up to 4.5× on real streams through compounding errors. Table 5b shows $\Phi_l$ (**1** when omitted) reduces MSE across all cells by 0.6–8.2% and cuts ETTm1-24 FR from 1.20 to 1.05. Persisting across windows, $\Phi_l$ prevents per-window updates from erasing latent directions.

## 4 Conclusion and Limitations

We bounded the stability-plasticity dilemma over horizon $H$, granularity $s$, and the online learning rate $\eta$. Guided by these bounds, SPOT-VSA uses vector-symbolic operations to bind dynamic and persistent latents. On leak-free streams, it rivals representative baselines with minimal forgetting while slashing edge latency and energy by up to 404× and 390×, respectively. However, its shallow architecture limits long-horizon nonlinear modeling, and the worst-case bounds capture directional trends rather than exact thresholds under adaptive optimization.

## AI USE STATEMENT

AI tools (Anthropic's Claude and Claude Code) were employed to support several stages of this work, in alignment with the ICLR 2027 AI policy:

- **Writing and Polish:** Assisting with drafting refinements, proofreading, and visual styling for figures and tables.
- **Research Ideation and Execution:** Assisting during the exploratory phases of model design, algorithm implementation, and experimental scripting.
- **Proving Mathematical Claims:** Providing initial verification steps, algebraic simplifications, and structural suggestions for formal proofs.

All intermediate outputs, mathematical proofs, experimental results, and final interpretations were rigorously vetted, verified, and remain the sole responsibility of the authors.

## REFERENCES

Awni Altabaa, Taylor Webb, Jonathan Cohen, and John Lafferty. Abstractors: Transformer modules for symbolic message passing and relational reasoning. *stat*, 1050:25, 2023.

Keith Ball. An elementary introduction to modern convex geometry. In Silvio Levy (ed.), *Flavors of Geometry*, volume 31 of *MSRI Publications*, pp. 1–58. Cambridge University Press, 1997.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Ruichu Cai, Haiqin Huang, Zhifan Jiang, Zijian Li, Changze Zhou, Yuequn Liu, Yuming Liu, and Zhifeng Hao. Disentangling long-short term state under unknown interventions for online time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15641–15649, 2025.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Mouxiang Chen, Lefei Shen, Han Fu, Zhuo Li, Jianling Sun, and Chenghao Liu. Calibration of time-series forecasting: Detecting and adapting context-driven distribution shift. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 341–352, 2024.

Seungyoon Choi, Hyunchul Kim, Jae-Gil Lee, and Chanyoung Park. Compositional spectral prompts for llm-based online time series forecasting. *arXiv preprint arXiv:2609.02093*, 2026.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Shijin Duan, Yejia Liu, Shaolei Ren, and Xiaolin Xu. Lehdc: Learning-based hyperdimensional computing classifier. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1111–1116, 2022.

Takumi Fujimoto and Hiroaki Nishi. When does online adaptation pay on the edge? a leakage-free evaluation of warmup, learning-rate selection, and resource trade-offs for time-series forecasting. *arXiv preprint arXiv:2609.01126*, 2026.

Alejandro Hernández-Cano, Namiko Matsumoto, Eric Ping, and Mohsen Imani. Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 56–61. IEEE, 2021.

Michael Hersche, Mustafa Zeqiri, Luca Benini, Abu Sebastian, and Abbas Rahimi. A neuro-vector-symbolic architecture for solving raven's progressive matrices. *Nature Machine Intelligence*, 5 (4):363–375, 2023.

Xiannan Huang, Shuhan Qiu, Jiayuan Du, and Chao Yang. Online time series prediction using feature adjustment. In *The Fourteenth International Conference on Learning Representations*, 2026.

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1:139–159, 2009.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021.

Ying-yee Ava Lau, Zhiwen Shao, and Dit-Yan Yeung. Fast and slow streams for online time series forecasting without information leakage. In *The Thirteenth International Conference on Learning Representations*, 2025.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

Yunlong Lin, Chao Lu, Tongshuai Wu, Xiaocong Zhao, Guodong Du, Yanwei Sun, Zirui Li, and Jianwei Gong. Escaping stability-plasticity dilemma in online continual learning for motion forecasting via synergetic memory rehearsal. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2026. doi: 10.1109/TNNLS.2026.3700190. Early access.

Jia Liu, Cheng Jinguo, Xia Fang, Zhenyuan Ma, and Yuankai Wu. Evaluating temporal plasticity in foundation time series models for incremental fine-tuning. *arXiv preprint arXiv:2504.14677*, 2025.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Tinghui Luo, Ziquan Fang, Kaixuan Duan, Lu Chen, Panpan Feng, and Mingfan Lu. Towards online spatio-temporal prediction: A knowledge distillation driven continual learning approach. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 2642–2655, 2025. doi: 10.1109/ICDE65448.2025.00199.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pp. 23190–23211. PMLR, 2023.

Mohamed Mejri, Chandramouli Amarnath, and Abhijit Chatterjee. A novel hyperdimensional computing framework for online time series forecasting on the edge. *arXiv preprint arXiv:2402.01999*, 2024a.

Mohamed Mejri, Chandramouli Amarnath, and Abhijit Chatterjee. Resolve: Relational reasoning with symbolic and object-level features using vector symbolic processing. *arXiv preprint arXiv:2411.08290*, 2024b.

Nicolas Menet, Michael Hersche, Geethan Karunaratne, Luca Benini, Abu Sebastian, and Abbas Rahimi. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven C. H. Hoi. Learning fast and slow for online time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Taylor W Webb, Ishan Sinha, and Jonathan D Cohen. Emergent symbols through binding in external memory. *arXiv preprint arXiv:2012.14601*, 2020.

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.

Yi-Fan Zhang, Qingsong Wen, Xue Wang, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. In *Advances in Neural Information Processing Systems*, volume 36, pp. 69949–69980, 2023.

Lifan Zhao and Yanyan Shen. Proactive model adaptation against concept drift for online time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 2020–2031. ACM, 2025. doi: 10.1145/3690624.3709210.

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.

# A Appendix

## A.1 Encoder Properties

Throughout this section $\rho(u, v) = uv^\top/(\|u\|\|v\|)$ denotes the uncentred correlation (cosine similarity) of two row vectors; it coincides with the Pearson correlation for mean-centred vectors, which is the case for windows normalized with the mean anchor, and asymptotically for the latent vectors $H_k$, whose coordinates have zero mean.

**Lemma A.1** (Quasi-orthogonality of the lag latent vectors)**.** *Let $P_i, P_k \in \mathbb{R}^D$, $i \neq k$, be independent with $P_i \sim \mathcal{N}(0, \sigma_i^2 I_D)$ and $P_k \sim \mathcal{N}(0, \sigma_k^2 I_D)$. Then $\mathbb{E}\,\rho(P_i, P_k) = 0$ and $\operatorname{Var}\rho(P_i, P_k) = 1/D$. Moreover, for any $\varepsilon \in (0, 1)$,*

$$\Pr\big(|\rho(P_i, P_k)| \geq \varepsilon\big) \leq 2\exp\big(-D\varepsilon^2/2\big), \tag{1}$$

*so that, with probability at least $1 - \delta$, $|\rho(P_i, P_k)| \leq \sqrt{2\ln(2/\delta)/D}$, and $\rho(P_i, P_k) = \mathcal{O}_p(1/\sqrt{D})$. If the rows $P_1, \ldots, P_L$ of the anchor are mutually independent, then with probability at least $1 - \delta$, simultaneously over all $L(L-1)/2$ pairs,*

$$\max_{i \neq k} |\rho(P_i, P_k)| \leq \sqrt{2\ln\big(L(L-1)/\delta\big)/D}. \tag{2}$$

*Proof.* $\rho$ is scale invariant, so take $\sigma_i = \sigma_k = 1$. Conditionally on $P_k$, rotational invariance of $P_i$ gives $\rho(P_i, P_k) \overset{d}{=} P_{i1}/\|P_i\|$, the first coordinate of a uniform point on the unit sphere $S^{D-1}$. By symmetry its mean is zero, and since the $D$ squared coordinates are exchangeable and sum to one, its second moment is $1/D$. For the tail, the normalized measure of the spherical cap $\{u \in S^{D-1} : u_1 \geq \varepsilon\}$ is at most $e^{-D\varepsilon^2/2}$ (Ball, 1997, Lemma 2.2); applying this to both caps $u_1 \geq \varepsilon$ and $u_1 \leq -\varepsilon$ and integrating over $P_k$ gives equation 1. Setting the right-hand side of equation 1 to $\delta$ and solving for $\varepsilon$ gives the high-probability bound, and equation 2 follows from a union bound over the $L(L-1)/2$ pairs, each assigned failure probability $2\delta/(L(L-1))$. □

**Numerical values.** The bounds are non-asymptotic and hold at the dimensions used in the paper. At $D = 500$, a single pair satisfies $|\rho| \leq 0.12$ with 95% confidence (the exact 95% quantile, from $(1+\rho)/2 \sim \mathrm{Beta}\left(\frac{D-1}{2}, \frac{D-1}{2}\right)$, is 0.088), and all pairs of an $L = 336$ lag basis satisfy $\max_{i \neq k} |\rho| \leq 0.24$ (observed maximum 0.20). At $D = 2000$ the corresponding values are 0.061 (0.044 exact) and 0.12 (0.10 observed). Figure 7 compares these bounds with Monte Carlo estimates over $D \in [10, 10^4]$: the empirical mean of $\rho$ is zero at every $D$, its 95% interval matches $\pm 1.96/\sqrt{D}$, and both high-probability bounds hold. Quasi-orthogonality in the sense of Lemma A.1 therefore does not require $D$ in the thousands: it is a statement about the $1/\sqrt{D}$ concentration of $\rho$, and $D = 500$ already places the typical pairwise similarity at $\mathbb{E}|\rho| \approx \sqrt{2/(\pi D)} = 0.036$.

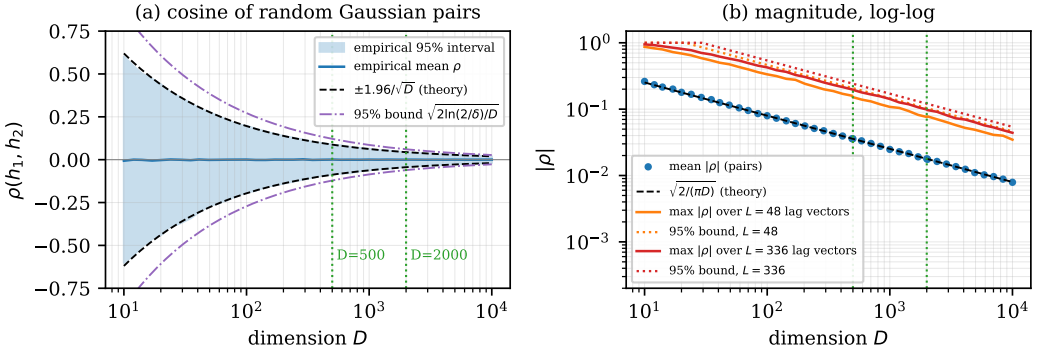



Figure 7: Cosine similarity $\rho$ between independent Gaussian vectors versus dimension $D$ (Monte Carlo, 5000 pairs per $D$). (a) Empirical mean and 95% interval of $\rho$, the Gaussian approximation $\pm 1.96/\sqrt{D}$, and the non-asymptotic 95% bound $\sqrt{2\ln(2/\delta)/D}$ of Lemma A.1. (b) Mean $|\rho|$ against $\sqrt{2/(\pi D)}$, and the largest $|\rho|$ over all pairs of an $L$-row lag basis against the uniform bound equation 2, for $L \in \{48, 336\}$. Dotted vertical lines mark $D = 500$ and $D = 2000$.

**Lemma A.2** (Independence under binding)**.** *Let $W \in \mathbb{R}^{L \times D}$ be deterministic and $P \in \mathbb{R}^{L \times D}$ have independent entries $p_{ij} \sim \mathcal{N}(0, \sigma_i^2)$. Then the entries of $W \odot P$ are mutually independent, with $[W \odot P]_{ij} \sim \mathcal{N}(0, w_{ij}^2 \sigma_i^2)$.*

*Proof.* Each entry is $b_{ij} = w_{ij} p_{ij}$, a deterministic scalar multiple of $p_{ij}$ alone. Measurable functions of independent random variables are independent, so the $b_{ij}$ are mutually independent, and a Gaussian scaled by $w_{ij}$ is Gaussian with mean zero and variance $w_{ij}^2 \sigma_i^2$ (degenerate at zero if $w_{ij} = 0$). Composition with a matrix, $WP^\top$ or $PW$, would instead mix several $p_{ij}$ into each entry and correlate them. □

**Lemma A.3** (Weighted correlation preservation)**.** *Let $B = W \odot P$ be as in Lemma A.2, with $\sup_{i,j} |w_{ij}| < \infty$, $\frac{1}{D}\|w_i\|^2 \to \bar{w}_i^2$ for every lag $i$, and $x_k \Sigma_W x_k^\top > 0$ for $k = 1, 2$. Then $\frac{1}{D}\mathbb{E}[BB^\top] \to \mathrm{diag}(\sigma_1^2 \bar{w}_1^2, \ldots, \sigma_L^2 \bar{w}_L^2) = \Sigma_W$, and for $H_k = x_k B$, $\rho(H_1, H_2) \xrightarrow{p} x_1 \Sigma_W x_2^\top / \left(\sqrt{x_1 \Sigma_W x_1^\top}\sqrt{x_2 \Sigma_W x_2^\top}\right)$ as $D \to \infty$. This is Proposition 2.1.*

*Proof.* For $i \neq k$, $\mathbb{E}[BB^\top]_{ik} = \sum_j \mathbb{E}[b_{ij} b_{kj}] = 0$ by independence and zero mean; for $i = k$, $\sum_j \mathbb{E}[b_{ij}^2] = \sigma_i^2 \sum_j w_{ij}^2 = \sigma_i^2 \|w_i\|^2$, and dividing by $D$ gives the first claim. Next, $\frac{1}{D} H_1 H_2^\top =$

$\frac{1}{D}\sum_{j=1}^{D}\xi_j$ with $\xi_j = (\sum_i x_{1i}b_{ij})(\sum_k x_{2k}b_{kj})$. The $\xi_j$ depend on disjoint columns of $P$ and are therefore independent, with $\mathbb{E}\,\xi_j = \sum_i x_{1i}x_{2i}\sigma_i^2 w_{ij}^2$ and variances bounded uniformly in $j$ because the $w_{ij}$ are bounded and Gaussian fourth moments are finite. By the weak law of large numbers for independent, uniformly square-integrable terms, $\frac{1}{D}H_1H_2^\top - \frac{1}{D}\sum_j \mathbb{E}\,\xi_j \xrightarrow{p} 0$, and $\frac{1}{D}\sum_j \mathbb{E}\,\xi_j \to x_1\Sigma_W x_2^\top$. The same argument with $x_1 = x_2$ gives $\frac{1}{D}\|H_k\|^2 \xrightarrow{p} x_k\Sigma_W x_k^\top > 0$, and the continuous mapping theorem yields the limit of the ratio. □

**Corollary A.4.** *If $\sigma_i\bar{w}_i = \kappa$ for all $i$, then $\Sigma_W = \kappa^2 I_L$ and $\rho(H_1, H_2) \xrightarrow{p} \rho(x_1, x_2)$.*

**Remark A.5** (Scope)**.** *Lemmas A.2–A.3 treat $W$ as fixed independently of $P$, which holds at initialization. Online training makes $W$ a function of $P$, and since $p_{ij} \neq 0$ almost surely, $W \odot P$ with unconstrained $W$ can represent any $L \times D$ matrix: the frozen anchor is not a hard constraint on the encoder. Its effect is on the dynamics, because $\partial\mathcal{L}/\partial w_{ij} = p_{ij}\,\partial\mathcal{L}/\partial b_{ij}$, so $P$ acts as a fixed random diagonal preconditioner of the encoder update. The statements above are therefore exact at initialization and describe the trained encoder only to the extent that $W$ stays weakly dependent on $P$; the stability role of $P$ after training is supported empirically by the encoder ablations. Note also that Proposition 2.1 implies that strongly correlated input channels have strongly correlated latent vectors: quasi-orthogonality holds across lags (Lemma A.1), not across channels.*

### A.2 Proofs for Section 2.8

**Assumption A.6.** *(i) $g_\theta$ is $\Lambda$-Lipschitz in its input, $\Lambda \geq 1$; write $\gamma = 1 + \Lambda$. (ii) With a ground-truth window, $\mathbb{E}\|g_\theta(u_k^{\mathrm{true}}) - y^{(k)}\| \leq \delta(c)$, where $\delta$ is nondecreasing (e.g. $\delta(c) = \delta_0 c^\alpha$, $\alpha > 0$). (iii) $\|\partial_\theta g_\theta\| \leq G$ and $\|g_{\theta'}(x) - g_\theta(x)\| \leq G_x\|\theta' - \theta\|$.*

*Proof of Theorem 2.2. Plasticity.* The windows $u_k$ and $u_k^{\mathrm{true}}$ differ only in the $k-1$ self-generated chunks, so $\|u_k - u_k^{\mathrm{true}}\| \leq \sum_{j<k} a_j$. By Assumption A.6(i)–(ii) and the triangle inequality,

$$\mathbb{E}\,a_k \leq \mathbb{E}\|g_\theta(u_k^{\mathrm{true}}) - y^{(k)}\| + \mathbb{E}\|g_\theta(u_k) - g_\theta(u_k^{\mathrm{true}})\| \leq \delta(c) + \Lambda\sum_{j<k}\mathbb{E}\,a_j.$$

Induction gives $\mathbb{E}\,a_k \leq \delta(c)\gamma^{k-1}$: if it holds up to $k-1$, then $\mathbb{E}\,a_k \leq \delta(c)\big(1 + \Lambda\frac{\gamma^{k-1}-1}{\gamma-1}\big) = \delta(c)\gamma^{k-1}$ because $\gamma - 1 = \Lambda$. Averaging over $k$ with $c = H/s$ gives the first bound:

$$\frac{1}{s}\sum_{k=1}^{s}\mathbb{E}[a_k] \leq \delta(H/s)\left(\frac{1}{s}\sum_{k=1}^{s}\gamma^{k-1}\right) = \delta(H/s)\Phi_p(s).$$

*Stability.* The total derivative through the rollout satisfies $J_k = \partial_\theta g_\theta(u_k) + \partial_u g_\theta(u_k)\,\frac{du_k}{d\theta}$, where $\frac{du_k}{d\theta}$ stacks the $J_j$, $j < k$, of the chunks fed back. Hence $\|J_k\| \leq G + \Lambda\sum_{j<k}\|J_j\|$, and the same induction gives $\|J_k\| \leq G\gamma^{k-1}$, deterministically. Therefore $\mathbb{E}\|g_t\| \leq \frac{1}{s}\sum_k \|J_k\|\,\mathbb{E}\,a_k \leq G\,\delta(c)\,\frac{1}{s}\sum_k \gamma^{2(k-1)} = G\,\delta(c)\,\Phi_\sigma(s)$, and $\Delta \leq G_x\eta\|g_t\|$ by Assumption A.6(iii).

*Shape in $s$.* For $\delta(c) = \delta_0 c^\alpha$ the plasticity bound is $\delta_0 H^\alpha s^{-\alpha}\Phi_p(s)$. Since $\Phi_p(s) \geq \gamma^{s-1}/s$ grows exponentially, the bound is eventually increasing and unbounded. Its value at $s = 2$ is below its value at $s = 1$ iff $2^{-\alpha}\Phi_p(2) < 1$, i.e. iff $\alpha > \log_2\big(\frac{1+\gamma}{2}\big)$ since $\Phi_p(2) = (1+\gamma)/2$; in that case the minimizer over the divisors of $H$ is some $s^\star > 1$ (for even $H$). The stability bound is handled identically with $\Phi_\sigma(2) = (1+\gamma^2)/2$, requiring $\alpha > \log_2\big(\frac{1+\gamma^2}{2}\big)$. Finally $\gamma^{2(k-1)} \leq \gamma^{s-1}\gamma^{k-1}$ for $k \leq s$ gives $\Phi_\sigma \leq \gamma^{s-1}\Phi_p$. □

**Remark A.7** (Teacher-forced updates)**.** *The stability bound above differentiates through self-generated chunks, i.e. it bounds a free-running update. Algorithm 1 instead teacher-forces the learning pass (line 23): each context $u_k$ is built from ground truth, so $du_k/d\theta = 0$ and $\|J_k\| \leq G$, and each chunk's residual is bounded by $\delta(c)$ through Assumption A.6(ii), without compounding. Algorithm 1 also applies one step per chunk, so by the triangle inequality over the $s$ steps and Assumption A.6(iii),*

$$\mathbb{E}\,\Delta \leq \eta\,G_x\sum_{k=1}^{s}G\,\delta(H/s) = \eta\,G_xG\,s\,\delta(H/s) = \eta\,G_xG\,\delta_0 H^\alpha s^{1-\alpha}.$$

*Hence, under teacher forcing, the displacement grows polynomially in $s$ when $\alpha < 1$, is constant when $\alpha = 1$, and* decreases *with $s$ when $\alpha > 1$; the $\mathcal{O}(\gamma^{2s})$ rate applies to free-running or scheduled-sampling updates. The plasticity bound concerns the free-running forecast (line 22) and is unaffected.*

*Proof of Theorem 2.3.* (i) Subtracting $\theta^\star_{t+1} = \theta^\star_t + \omega_t$ from the update gives $e_{t+1} = (1-\eta)e_t + \eta\epsilon_t - \omega_t$. With $\epsilon_t$, $\omega_t$ zero-mean and independent of $e_t$ and of each other, $V_{t+1} = (1-\eta)^2 V_t + \eta^2\sigma^2_{\text{eff}} + q$. For $0 < \eta < 2$ this is a contraction with fixed point $V_\infty = \big(\eta^2\sigma^2_{\text{eff}} + q\big)/\big(\eta(2-\eta)\big)$, which is the stated decomposition:

$$V_\infty(\eta) = \frac{\eta}{2-\eta}\sigma^2_{\text{eff}} + \frac{q}{\eta(2-\eta)}.$$

(ii) If $q = 0$, $V_\infty = \eta\sigma^2_{\text{eff}}/(2-\eta)$ is increasing in $\eta$. If $q > 0$, $V_\infty \to \infty$ as $\eta \to 0$ and as $\eta \to 2$; setting $dV_\infty/d\eta = 0$ gives $\sigma^2_{\text{eff}}\eta^2 + q\eta - q = 0$, whose only positive root is $\eta^\star = \big(\sqrt{q^2 + 4q\sigma^2_{\text{eff}}} - q\big)/\big(2\sigma^2_{\text{eff}}\big) = 2\sqrt{q}/\big(\sqrt{q} + \sqrt{q + 4\sigma^2_{\text{eff}}}\big) \in (0,1)$. It is decreasing in $\sigma^2_{\text{eff}} = \sigma^2 + \Pi$, hence in $\Pi$, and increasing in $q$.

(iii) At steady state the deployed parameters differ from the optimum by $e_t$, so by Assumption A.6(iii) and Jensen's inequality the single-chunk error on a ground-truth window is at most $\delta(c) + G_x\mathbb{E}\|e_t\| \le \delta(c) + G_x\sqrt{V_\infty(\eta)}$. Repeating the proof of Theorem 2.2 with this quantity in place of $\delta(c)$ yields the two bounds of Theorem 2.3: $\mathcal{P} \le \big(\delta(H/s) + G_x\sqrt{V_\infty(\eta)}\big)\Phi_p(s)$ and $\Delta \le \eta\, G_x G\big(\delta(H/s) + G_x\sqrt{V_\infty(\eta)}\big)\Phi_\sigma(s)$. The plasticity bound is increasing in $V_\infty$ and is therefore minimized at $\eta^\star$. In the stability bound, $\eta\,\delta(H/s)$ is increasing in $\eta$ and so is $\eta\sqrt{V_\infty(\eta)} = \sqrt{(\eta^3\sigma^2_{\text{eff}} + \eta q)/(2-\eta)}$, so that bound is strictly increasing. For $q = 0$ both bounds are increasing in $\eta$ and no disagreement remains. □

### A.3 Extended Optimization and Update Rules

#### A.3.1 Disentangled Loss Objectives

During learning passes, `SPOT-VSA` segregates gradients across two distinct modules:

- **Short-term loss ($\mathcal{L}_s$):** Evaluated as the MSE between $\hat{Y}_{\text{chunk}}$ and the ground-truth targets, updating the fast-adapting encoder $\mathbf{\Phi}_s$ and decoder $\mathbf{\Psi}$.
- **Long-term loss ($\mathcal{L}_l$):** Evaluated on the same chunk to update only the persistent inductive bias $\mathbf{\Phi}_l$. While $\mathcal{L}_l$ can leverage MAE (L1 loss) as a robust regularizer for slow latent drift, benchmark evaluations under the strict online protocol found that configuring both $\mathcal{L}_s$ and $\mathcal{L}_l$ with MSE yielded optimal empirical performance.

Because gradients flow into disjoint parameter partitions, the combined gradient vector $g = (\nabla_{\mathbf{\Phi}_s,\mathbf{\Psi}}\mathcal{L}_s,\ \nabla_{\mathbf{\Phi}_l}\mathcal{L}_l)$ is applied simultaneously in a single clipped AdamW step per chunk.

#### A.3.2 Loss-Driven Scheduled Sampling

In autoregressive rollouts ($c < H$), the training update rule governs how past predictions and true targets populate the contextual look-back window $X_{\text{auto}}$. Although pure teacher forcing is the default in all reported headline benchmarks, the framework supports adaptive scheduled sampling (Bengio et al., 2015) where the probability $\epsilon$ of appending ground-truth observations scales dynamically with the instantaneous loss:

$$\epsilon = f(\mathcal{L}_s), \qquad X_{\text{auto}} \leftarrow \Big(X_{\text{auto}} \cup \begin{cases} Y_{\text{gt}} & \text{with probability } \epsilon \\ \hat{Y}_{\text{chunk}} & \text{with probability } 1-\epsilon \end{cases}\Big)[-L:]. \tag{3}$$

Here, $f(\cdot)$ is a non-decreasing transfer function: elevated loss (signaling a distribution shift) increases the probability of ground-truth grounding for fast recovery, while low loss allows the model to encounter its own feedback errors. Supported formulations include:

1. **Clipped linear:** $f(\mathcal{L}_s) = \min\big(1,\ \alpha_1\mathcal{L}_s + \alpha_2\big)$ with $\alpha_1 \ge 0, \alpha_2 \in [0,1]$.

2. **Sigmoidal:** $f(\mathcal{L}_s) = \frac{\alpha_1}{1+\exp(-\alpha_2 \mathcal{L}_s)}$.

Setting $\epsilon = 1$ recovers pure teacher forcing, which was maintained across all main-paper experiments to strictly isolate the impact of chunk granularity $c$ from stochastic sampling dynamics.

### A.4 Experimental Details

#### A.4.1 Datasets and Preprocessing

Table 6 summarizes the benchmark configurations, temporal lengths, sampling granularities, and dataset splits. Each multivariate sequence is standardized per channel using the mean and standard deviation estimated strictly on the initial warm-up split; all downstream adaptation steps and error metrics operate in this coordinate space.

For the Electricity Transformer Temperature (ETT) suite, we preserve the exact month demarcations formalized by Pham et al. (2023): warm-up corresponds to months 1–4, validation to month 5, and the online streaming phase encompasses months 6–20. A standard month is set to 720 observations for the hourly series (ETTh1, ETTh2) and is aligned to the identical 720-step monthly boundaries for the 15-minute variants (ETTm1, ETTm2), truncating their streaming evaluation to the first $14\,400$ observations to maintain parity in evaluation duration across sampling resolutions. The remaining benchmarks (WTH, Exchange, ECL, Traffic) utilize an exact $20\%/5\%/75\%$ chronological split. For WTH and ECL, the primary evaluation targets correspond to channel indices `WetBulbCelsius` and `MT_320`, respectively.

Under the horizon-strided execution (stride $= H$), the number of non-overlapping evaluated windows scales inversely with the prediction horizon: ETT evaluates $449$, $64$, and $32$ windows; WTH evaluates $1\,095$, $156$, and $78$; Exchange evaluates $237$, $33$, and $16$; ECL evaluates $821$, $117$, and $58$; and Traffic evaluates $197$, $28$, and $14$ windows across $H \in \{24, 168, 336\}$, respectively. At extended horizons ($H = 336$), datasets with shorter temporal lengths exhibit higher cross-seed variance ($\approx 5\%$). On Traffic-336, default data-loaders for several baselines fail due to validation splits shorter than $L + H$; `SPOT-VSA` directly reuses its $H = 168$ validation-selected hyperparameter assignment for this setting.

Table 6: Benchmark datasets, dimensionalities, sampling rates, and temporal partitions.

| **Dataset** | **Channels ($F$)** | **Length** | **Sampling** | **Warm-up / Validation / Stream Partition** |
|---|---|---|---|---|
| ETTh1, ETTh2 | 7 | 17 420 | 1 hour | Months 1–4 / 5 / 6–20 (720 steps/month) |
| ETTm1, ETTm2 | 7 | 69 680 | 15 min | First 14 400 steps (using identical month indices) |
| WTH | 12 | 35 064 | 1 hour | 20% / 5% / 75% chronological |
| Exchange | 8 | 7 588 | 1 day | 20% / 5% / 75% chronological |
| ECL | 321 | 26 304 | 1 hour | 20% / 5% / 75% chronological |
| Traffic | 862 | 6 334 | 1 hour | 20% / 5% / 75% chronological |

**Synthetic drift generation.** The synthetic benchmark S-A consists of six concatenations of stationary AR(1) processes parameterized by autoregressive coefficients $\phi \in \{0.1, 0.4, 0.6, 0.1, 0.4, 0.6\}$ under unit-variance Gaussian innovations:

$$x_t = \phi^{(m)} x_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1), \quad m \in \{1, \ldots, 6\}. \tag{4}$$

The S-G benchmark incorporates identical underlying autoregressive parameters, but models continuous transitions across adjacent processes over the final 20% of each regime via convex combination:

$$x_t = \left(1 - \frac{t - t_0}{\Delta t}\right) x_t^{(m)} + \left(\frac{t - t_0}{\Delta t}\right) x_t^{(m+1)}. \tag{5}$$

Standard synthetic sequences instantiate segment lengths of $\Delta t = 1\,000$ steps. For the stability and forgetting analyses, we employ the extended variants S-A-L and S-G-L, which specify regime lengths of $\Delta t = 10\,000$ steps ($6 \times 10^4$ total steps, with 75% assigned to streaming evaluation). To evaluate varying drift velocities, we additionally introduce a stationary AR(1) baseline ($\phi = 0.4$, drift energy $q = 0$) and accelerated variants of S-A comprising 24 regimes ($\Delta t = 2\,500$) and 60 regimes ($\Delta t = 1\,000$), corresponding to $4\times$ and $10\times$ higher drift frequencies, respectively.

### A.4.2 Online Streaming Evaluation Protocol

Prior online forecasting literature (Pham et al., 2023; Zhang et al., 2023) frequently executes evaluations using an autoregressive sliding window with unit stride (stride $= 1$), updating parameters after each sequential observation. This introduces structural target overlap: adjacent forecast targets share $H - 1$ values. Under this setting, models can achieve low empirical error by simply memorizing target histories. Furthermore, chunked autoregressive architectures ($c < H$) under unit stride inadvertently gain access to future ground-truth values revealed in preceding parameter updates.

To ensure strict delayed-feedback evaluation, we enforce a non-overlapping horizon-strided protocol (stride $= H$) across all evaluated methods:

1. At streaming window index $k$, the model ingests historical context $X_k = X[k - L : k] \in \mathbb{R}^{L \times F}$ and generates prediction $\hat{Y}_k = f_\theta(X_k) \in \mathbb{R}^{H \times F}$ with gradient tracking disabled.
2. The forecast $\hat{Y}_k$ is scored against the true target $Y_k = X[k : k + H]$.
3. Ground truth $Y_k$ is revealed, and the model executes one online learning pass on $(X_k, Y_k)$ with $n_{\text{inner}} = 1$.
4. The temporal index advances by $H$, preventing any evaluation target overlap between updates.

For SPOT-VSA, this update corresponds to the chunked pass in Algorithm 1 ($s = H/c$ optimizer steps, with $s = 1$, i.e. a single step, in all headline runs). Baseline models execute a single step of their respective adaptation rules on $(X_k, Y_k)$. Context lengths are set to $L = 60$ ($L = 288$ on ETTm) at $H = 24$, and $L = 2H$ for $H \geq 168$ for SPOT-VSA. Baseline architectures adopt their native look-back lengths ($L \in \{60, 288\}$ at $H = 24$, and $L = H$ for $H \geq 168$).

## A.5 Online Vs Offline Plasticity Comparison

Table 7: SPOT-VSA (online, one update per window) vs. offline forecasters trained on the warm-up split and frozen during the stream (stride $H$, 4 seeds). Best: **green**, second: **blue**. $\Delta$: relative change of SPOT-VSA vs. the better offline model (%; gain, loss).

| Metric | | RMSE | | | | MAE | | | |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | H | PatchTST | DLinear | SPOT-VSA | $\Delta$ (%) | PatchTST | DLinear | SPOT-VSA | $\Delta$ (%) |
| ETTh1 | 24 | 0.845 | **0.810** | **0.747** | +7.8 | 0.567 | **0.547** | **0.487** | +11.0 |
| | 168 | 1.190 | **1.068** | **1.026** | +3.9 | 0.806 | **0.735** | **0.696** | +5.3 |
| | 336 | **1.247** | 1.332 | **1.141** | +8.5 | **0.877** | 0.945 | **0.803** | +8.4 |
| ETTh2 | 24 | **0.895** | 1.003 | **0.832** | +7.0 | **0.589** | 0.691 | **0.555** | +5.8 |
| | 168 | **1.667** | **1.694** | 1.714 | -2.8 | **1.026** | 1.082 | **1.049** | -2.2 |
| | 336 | **2.650** | **2.398** | 2.893 | -20.6 | **1.524** | **1.416** | 1.610 | -13.7 |
| ETTm1 | 24 | 0.547 | **0.534** | **0.485** | +9.2 | 0.418 | **0.413** | **0.370** | +10.4 |
| | 168 | **0.769** | 0.824 | **0.744** | +3.3 | **0.577** | 0.631 | **0.553** | +4.2 |
| | 336 | **0.955** | 0.987 | **0.901** | +5.7 | **0.706** | 0.733 | **0.659** | +6.7 |
| ETTm2 | 24 | **0.790** | 0.830 | **0.781** | +1.1 | **0.499** | 0.533 | **0.518** | -3.8 |
| | 168 | **0.946** | 1.070 | **1.034** | -9.3 | **0.623** | 0.724 | **0.690** | -10.8 |
| | 336 | **1.145** | **1.310** | 1.896 | -65.6 | **0.786** | **0.907** | 1.428 | -81.7 |
| WTH | 24 | 0.571 | **0.569** | **0.554** | +2.6 | **0.392** | 0.401 | **0.381** | +2.8 |
| | 168 | 0.727 | **0.701** | **0.709** | -1.1 | 0.537 | **0.521** | **0.522** | -0.2 |
| | 336 | **0.772** | **0.728** | 0.844 | -15.9 | **0.573** | **0.541** | 0.642 | -18.7 |
| Exchange | 24 | **0.276** | 0.374 | **0.298** | -8.0 | **0.189** | 0.266 | **0.205** | -8.5 |
| | 168 | **0.759** | 1.113 | **0.952** | -25.4 | **0.529** | 0.749 | **0.641** | -21.2 |
| | 336 | **1.228** | 2.384 | **1.300** | -5.9 | **0.824** | 1.543 | **0.896** | -8.7 |
| ECL | 24 | **3.131** | 10.9 | **1.996** | +36.3 | **0.417** | 0.885 | **0.429** | -2.9 |
| | 168 | **2.881** | 15.2 | **3.320** | -15.2 | **0.403** | 1.124 | **0.596** | -47.9 |
| | 336 | **3.415** | 15.3 | **3.938** | -15.3 | **0.472** | 1.142 | **0.881** | -86.7 |
| Traffic | 24 | **0.743** | 0.835 | **0.685** | +7.8 | **0.400** | 0.495 | **0.391** | +2.3 |
| | 168 | **0.642** | 0.774 | **0.632** | +1.6 | **0.320** | 0.444 | **0.342** | -6.9 |

Table 7 shows the plasticity performance of SPOT-VSA compared to two offline baselines, **DLinear** Zeng et al. (2023) and **PatchTST** Nie et al. (2022), in terms of RMSE and MAE. SPOT-VSA

achieves the best RMSE in 12 of the 23 dataset–horizon settings and ranks second in 7 more: all three horizons of ETTh1 and ETTm1, both horizons of Traffic, and $H{=}24$ on ETTh2, ETTm2, WTH, and ECL. For MAE, it is best in 9 settings and second in 11. The largest gains appear at short horizons, reaching $+36.3\%$ RMSE on ECL ($H{=}24$) and up to $+11.0\%$ MAE on ETTh1 ($H{=}24$). SPOT−VSA consistently leads on ETTh1 and ETTm1, improving RMSE by 3.9–$9.2\%$ and MAE by 4.2–$11.0\%$ over the stronger offline baseline at every horizon. It trails the offline models mainly at the longest horizon, $H{=}336$, on ETTm2, ETTh2, and WTH, and on Exchange. The largest drops occur on ETTm2 ($-65.6\%$ RMSE, $-81.7\%$ MAE). These horizons leave few stream updates, so a model trained once on the warm-up split has an advantage.

### A.6 Online Learning Results Standard Deviation

Tables 8 and 9 show the standard deviation across 4 seeds for the plasticity and stability results generation, respectively.

Table 8: Standard deviation (over seeds) of the online RMSE and MAE in Table 2 (stride $H$). 4 seeds per cell, except CoSPOT (3) and LSTD on ECL-24/168 and Traffic (3) and ECL-336 (2).

| Metric | | RMSE (std) | | | | | | | | | MAE (std) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | H | FSNet | OneNet | OneNet-TCN | ER | DER++ | LSTD | CoSPOT | DSOF | SPOT-VSA | FSNet | OneNet | OneNet-TCN | ER | DER++ | LSTD | CoSPOT | DSOF | SPOT-VSA |
| ETTh1 | 24 | 0.0575 | 0.0868 | 0.0045 | 0.0187 | 0.0205 | 0.0105 | 0.0416 | 0.0083 | 0.0001 | 0.0445 | 0.0804 | 0.0022 | 0.0165 | 0.0178 | 0.0086 | 0.0250 | 0.0054 | 0.0001 |
| | 168 | 0.1677 | 0.0793 | 0.0146 | 0.0231 | 0.0242 | 0.0716 | 0.0739 | 0.0068 | 0.0003 | 0.1482 | 0.0645 | 0.0081 | 0.0214 | 0.0219 | 0.0611 | 0.0526 | 0.0064 | 0.0003 |
| | 336 | 0.1011 | 0.2461 | 0.0826 | 0.0940 | 0.0769 | 0.1261 | 0.0083 | 0.0100 | 0.0013 | 0.1006 | 0.1828 | 0.0674 | 0.0750 | 0.0646 | 0.0843 | 0.0041 | 0.0096 | 0.0011 |
| ETTh2 | 24 | 0.0196 | 0.0427 | 0.1563 | 0.0440 | 0.0460 | 0.0146 | 0.0134 | 0.0051 | 0.0006 | 0.0157 | 0.0223 | 0.1401 | 0.0303 | 0.0301 | 0.0191 | 0.0120 | 0.0031 | 0.0004 |
| | 168 | 0.2424 | 0.1891 | 0.2440 | 0.0725 | 0.0707 | 0.0674 | 0.0750 | 0.0569 | 0.0018 | 0.2095 | 0.0813 | 0.2124 | 0.0857 | 0.0844 | 0.0537 | 0.0381 | 0.0335 | 0.0011 |
| | 336 | 0.1896 | 0.2531 | 0.1005 | 0.5220 | 0.5160 | 0.2000 | 0.0034 | 0.1328 | 0.0167 | 0.1472 | 0.1740 | 0.0853 | 0.4385 | 0.4338 | 0.1050 | 0.0014 | 0.0903 | 0.0076 |
| ETTm1 | 24 | 0.0107 | 0.0103 | 0.0031 | 0.0057 | 0.0044 | 0.0242 | 0.1271 | 0.0074 | 0.0004 | 0.0097 | 0.0095 | 0.0027 | 0.0049 | 0.0041 | 0.0203 | 0.0969 | 0.0066 | 0.0005 |
| | 168 | 0.0442 | 0.1565 | 0.0047 | 0.0159 | 0.0142 | 0.0291 | 0.0695 | 0.0128 | 0.0004 | 0.0306 | 0.1359 | 0.0056 | 0.0121 | 0.0111 | 0.0300 | 0.0497 | 0.0064 | 0.0005 |
| | 336 | 0.8699 | 0.1726 | 0.0462 | 0.0686 | 0.0759 | 0.0137 | 0.0005 | 0.0285 | 0.0010 | 0.7457 | 0.1213 | 0.0391 | 0.0529 | 0.0568 | 0.0179 | 0.0004 | 0.0261 | 0.0007 |
| ETTm2 | 24 | 0.0189 | 0.0030 | 0.0127 | 0.0051 | 0.0045 | 0.0270 | 0.0722 | 0.0053 | 0.0001 | 0.0123 | 0.0028 | 0.0126 | 0.0059 | 0.0063 | 0.0238 | 0.0660 | 0.0040 | 0.0002 |
| | 168 | 0.1055 | 0.0712 | 0.0093 | 0.0099 | 0.0029 | 0.0201 | 0.0065 | 0.0156 | 0.0066 | 0.0945 | 0.0494 | 0.0121 | 0.0222 | 0.0116 | 0.0221 | 0.0113 | 0.0082 | 0.0025 |
| | 336 | 0.2814 | 0.0171 | 0.0551 | 0.2083 | 0.2095 | 0.0370 | 0.0001 | 0.0501 | 0.0135 | 0.2612 | 0.0176 | 0.0414 | 0.1867 | 0.1873 | 0.0320 | 0.0001 | 0.0389 | 0.0108 |
| WTH | 24 | 0.0006 | 0.0071 | 0.0114 | 0.0105 | 0.0089 | 0.0008 | 0.0053 | 0.0003 | 0.0001 | 0.0006 | 0.0053 | 0.0144 | 0.0108 | 0.0091 | 0.0006 | 0.0040 | 0.0005 | 0.0001 |
| | 168 | 0.0292 | 0.0102 | 0.0085 | 0.0179 | 0.0105 | 0.0037 | 0.0060 | 0.0014 | 0.0016 | 0.0258 | 0.0108 | 0.0080 | 0.0175 | 0.0110 | 0.0030 | 0.0039 | 0.0013 | 0.0013 |
| | 336 | 0.0874 | 0.0267 | 0.0454 | 0.0524 | 0.0087 | 0.0099 | 0.0001 | 0.0036 | 0.0280 | 0.0792 | 0.0269 | 0.0393 | 0.0462 | 0.0089 | 0.0099 | 0.0002 | 0.0045 | 0.0237 |
| Exchange | 24 | 0.0663 | 0.1376 | 0.3168 | 0.0584 | 0.0565 | 0.0004 | 0.0058 | 0.0146 | 0.0073 | 0.0544 | 0.1084 | 0.2043 | 0.0442 | 0.0437 | 0.0006 | 0.0042 | 0.0104 | 0.0047 |
| | 168 | 0.0247 | 0.0974 | 0.2798 | 0.0314 | 0.0314 | 0.1401 | 0.0020 | 0.0454 | 0.1124 | 0.0219 | 0.0781 | 0.2372 | 0.0261 | 0.0261 | 0.1207 | 0.0009 | 0.0318 | 0.0657 |
| | 336 | 0.0031 | 0.0959 | 0.1149 | 0.0396 | 0.0339 | 0.2292 | 0.0467 | 0.0690 | 0.0046 | 0.0043 | 0.0766 | 0.1040 | 0.0375 | 0.0317 | 0.2000 | 0.0255 | 0.0594 | 0.0069 |
| ECL | 24 | 0.0202 | 0.5852 | 1.8562 | 0.0246 | 0.0271 | 0.0059 | 0.0286 | 0.0058 | 0.0050 | 0.0398 | 0.0750 | 0.2379 | 0.0110 | 0.0072 | 0.0068 | 0.0076 | 0.0028 | 0.0010 |
| | 168 | 0.4531 | 0.4951 | 0.1855 | 0.1538 | 0.1408 | 0.0241 | 0.0160 | 0.2911 | 0.0478 | 0.7568 | 0.1273 | 0.1257 | 0.0358 | 0.0506 | 0.3608 | 0.0009 | 0.0277 | 0.0034 |
| | 336 | 0.1373 | 1.4689 | 0.9542 | 1.7881 | 1.4515 | 0.0827 | 0.0329 | 0.1658 | 0.0677 | 0.0973 | 0.1051 | 0.2754 | 0.4440 | 0.6339 | 0.1118 | 0.0027 | 0.0145 | 0.0180 |
| Traffic | 24 | 0.0039 | 0.0163 | 0.0176 | 0.0025 | 0.0030 | 0.0097 | 0.0200 | 0.0052 | 0.0006 | 0.0028 | 0.0155 | 0.0136 | 0.0030 | 0.0037 | 0.0049 | 0.0213 | 0.0044 | 0.0006 |
| | 168 | 0.0052 | 0.0087 | 0.0060 | 0.0166 | 0.0164 | 0.0013 | 0.0123 | 0.0096 | 0.0069 | 0.0063 | 0.0136 | 0.0063 | 0.0199 | 0.0193 | 0.0014 | 0.0139 | 0.0078 | 0.0113 |

Table 9: Standard deviation (over seeds) of the forgetting ratio in Table 3; seed counts as in Table A.6. $0^*$: FR $= 1.00$ (no forgetting) in every seed, so the spread is exactly zero.

| | ETTh1 | | | ETTh2 | | | ETTm1 | | | ETTm2 | | | WTH | | | Exchange | | | ECL | | | Traffic | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 | 336 | 24 | 168 |
| FSNet | 0.093 | 0.196 | 0.463 | 0.221 | 0.615 | 0.123 | 0.285 | 0.160 | 0.087 | 0.049 | 0.171 | 0.088 | 0.104 | 0.114 | 0.046 | 3.008 | 0.051 | 0.044 | 4.557 | 14.529 | 20.260 | 0.351 | 0.056 |
| OneNet | 0.135 | 0.014 | 0.093 | 0.039 | 0.106 | 0.272 | 0.231 | 0.368 | 0.283 | 0.090 | 0.097 | 0.094 | 0.142 | 0.254 | 0.176 | 2.964 | 2.379 | 0.113 | 0.090 | 29.155 | 10.603 | 0.030 | 0.017 |
| OneNet-TCN | 0.115 | 0.086 | 0.048 | 0.783 | 0.291 | 0.171 | 0.165 | 0.118 | 0.054 | 0.283 | 0.113 | 0.075 | 0.036 | 0.092 | 0.039 | 0.363 | 0.747 | 0.116 | 11.705 | 0.089 | 0.081 | 0.019 | $0^*$ |
| ER | 0.062 | 0.140 | 0.080 | 0.101 | 0.260 | 0.014 | 0.080 | $0^*$ | 0.060 | 0.157 | 0.018 | 0.052 | 0.018 | 0.007 | 0.106 | 0.229 | 0.064 | $0^*$ | 0.139 | 0.028 | 0.326 | 0.059 | 0.076 |
| DER++ | 0.086 | 0.067 | 0.077 | 0.066 | 0.195 | 0.015 | 0.073 | $0^*$ | 0.084 | 0.134 | 0.014 | 0.071 | 0.021 | 0.017 | 0.026 | 0.221 | 0.093 | $0^*$ | 0.184 | 0.049 | 0.261 | 0.026 | 0.012 |
| LSTD | 0.055 | 0.127 | 0.063 | 0.159 | 0.812 | 4.787 | 1.087 | 0.478 | 0.666 | 0.019 | 0.163 | 0.044 | $0^*$ | 0.080 | 0.402 | 0.005 | 0.696 | 1.161 | 0.088 | 0.011 | <0.001 | 0.027 | $0^*$ |
| CoSPOT | 0.031 | $0^*$ | $0^*$ | 0.006 | <0.001 | <0.001 | 0.052 | 0.002 | <0.001 | 0.010 | <0.001 | <0.001 | 0.007 | <0.001 | <0.001 | 0.005 | $0^*$ | $0^*$ | 0.026 | 0.002 | <0.001 | 0.052 | 0.017 |
| DSOF | 0.038 | 0.018 | 0.042 | 0.049 | 0.024 | 0.160 | 0.663 | 0.733 | 0.341 | 0.301 | 0.987 | 0.188 | 0.025 | 0.052 | 0.028 | 0.058 | 0.042 | 0.536 | 0.083 | 0.560 | 0.142 | 0.117 | 0.147 |
| SPOT-VSA | <0.001 | <0.001 | 0.002 | 0.003 | 0.003 | $0^*$ | 0.002 | <0.001 | $0^*$ | $0^*$ | 0.002 | $0^*$ | $0^*$ | 0.002 | 0.001 | 0.082 | 0.077 | 0.015 | 0.027 | 0.034 | 0.140 | 0.002 | 0.007 |

#### A.6.1 Stability–plasticity sweeps along the knobs

**Normalisation and aggregation.** For each dataset–horizon pair, online MSE (plasticity) and FR (stability) are min–max normalized over the sweep (0 = best, 1 = worst) and then averaged across pairs; the shaded bands in Figure 5 show the spread across pairs.

**Granularity (P1).** Figure 5a sweeps $s = H/C$. The forgetting curve is not strictly monotone (e.g. a bump at $s{=}4$): the aggregated pairs differ in drift, trend, seasonality and noise level, and individual datasets respond differently to $s$, but the overall trend follows P1.

**Learning rate without drift (P3).** To isolate $\eta$ in the absence of drift we use a synthetic stationary AR(1) stream, which contains no drift by construction. The adaptation error increases with $\eta$ by only a few percent, while forgetting increases monotonically.

**Learning rate under drift (P4).** Real-world streams always exhibit some drift, in amounts that vary across datasets; we therefore evaluate P4 on eight real dataset–horizon pairs: ETTh1, ETTm1, WTH, ECL and Exchange at $H{=}24$, and ETTh1, ETTm2 and Exchange at $H{=}168$.

### A.6.2 Evaluation Metrics and Probing Methodology

Streaming predictive performance is measured via cumulative Mean Squared Error (MSE) and Mean Absolute Error (MAE) computed over the complete sequence of $K$ non-overlapping windows:

$$\mathrm{MSE} = \frac{1}{K \cdot H \cdot F} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{j=1}^{F} \left( \hat{Y}_{k,h,j} - Y_{k,h,j} \right)^2, \tag{6}$$

$$\mathrm{MAE} = \frac{1}{K \cdot H \cdot F} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{j=1}^{F} \left| \hat{Y}_{k,h,j} - Y_{k,h,j} \right|. \tag{7}$$

Table 2 reports the root mean squared error computed per window and averaged over the stream, $\mathrm{RMSE} = \frac{1}{K} \sum_{k=1}^{K} \left( \frac{1}{HF} \sum_{h,j} (\hat{Y}_{k,h,j} - Y_{k,h,j})^2 \right)^{1/2}$, which weights every window equally and is less dominated by a few high-error windows than the cumulative MSE; the sweeps and ablations report the cumulative MSE.

**Retention probe and forgetting metrics.** To disentangle plasticity from catastrophic forgetting, we periodically evaluate retention on initial streaming observations. Specifically, the initial $P = 16$ windows of the test stream are cached. At eight equidistant checkpoints $\{t_1, t_2, \ldots, t_8\}$ throughout the stream, the current model parameters $\theta_t$ are evaluated on this cached set without parameter adaptation. Letting $p_t$ denote the aggregate MSE on the probe set at checkpoint $t$, we compute:

$$\mathrm{FR} = \frac{p_{t_8}}{\min_{t \in \{t_1, \ldots, t_8\}} p_t}, \qquad \mathrm{BWT} = p_{t_8} - p_{t_1}. \tag{8}$$

By construction $\mathrm{FR} \geq 1$; $\mathrm{FR} = 1$ means that the final probe error equals the best checkpoint error (no final degradation), and $\mathrm{FR} > 1$ quantifies the degradation on historical patterns relative to that best checkpoint. FR does not penalize forgetting that is later recovered, and BWT complements it as a first-versus-final measure. Static offline models satisfy $\mathrm{FR} = 1.0$ by construction. For extended synthetic streams, probe sets are sampled per segment (64 windows per regime across 40 checkpoints).

**Function-space parameter stability.** In the stability–plasticity ablation (Appendix A.7.2), post-update network deviation is measured in function space:

$$\Delta_t = \frac{1}{|\mathcal{X}_{\text{probe}}|} \sum_{x \in \mathcal{X}_{\text{probe}}} \| f_{\theta_{t+1}}(x) - f_{\theta_t}(x) \|_2, \tag{9}$$

evaluated across 32 stationary reference windows distributed across the stream.

### A.6.3 SPOT-VSA Configuration and Hyperparameter Selection

Table 10 outlines the configuration parameters for SPOT-VSA. Offline warm-up is conducted for a single epoch ($E = 1$) with unit stride using AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$, weight decay $\lambda = 2 \times 10^{-3}$, and gradient clipping $\kappa = 1.0$). On ECL, warm-up is extended to 15 epochs ($\eta_0 = 2 \times 10^{-3}$) to accommodate the representational capacity needed for 321 interacting series. Here we use a chunk size of H to build the Tables 3 and 2 even though this is not the optimal chunk size, to make sure we are tied to the state of the art we are comparing against and to assess the architectural performance provided by SPOT-VSA using the same prediction steps and mechanism as the SOTA.

The two free hyperparameters—instance normalization anchor $\mu_x \in \{\text{none}, \text{last}, \text{mean}\}$ and online learning rate $\eta \in \{10^{-4}, 10^{-3}, 10^{-2}\}$—are determined via grid search exclusively on the validation stream under $H = 24$ with stride $= 24$. The selected pair is then held fixed across all horizons ($H \in \{168, 336\}$). The selected configurations (anchor, $\eta$) are: ETTh1 $(\text{mean}, 10^{-3})$, ETTh2

Table 10: Hyperparameters and operational settings of SPOT−VSA in all headline results (strict online protocol). Per-dataset choices are listed in Table 11.

| Parameter | Setting |
|---|---|
| *Architecture (Sections 2.4–2.6)* | |
| Hypervector dimension $D$ | 500 (ETTh1: 2000); not tuned, see Appendix A.6.6 |
| Frozen anchor $P \in \mathbb{R}^{L\times D}$ | rows $P_i \sim \mathcal{N}(0, \sigma_i^2 I_D)$, sampled once and never updated; binding $B = W \odot P$ with $W \in \mathbb{R}^{L\times D}$ learnable (Lemma A.2) |
| Activation | SiLU, element-wise on the bound representation $Z$ |
| Inductive bias $\mathbf{\Phi}_l \in \mathbb{R}^{D\times F}$ | learnable, initialised to $\mathbf{1}$, bound to $\mathbf{\Phi}_s(\tilde{X})$ by Hadamard product |
| Decoder $\mathbf{\Psi}$ | linear readout $W_\Psi \in \mathbb{R}^{D\times c}$, channel-independent |
| Channel mixing matrix $M$ | learnable $F \times F$, initialised to $I$, applied to the output forecast $\tilde{Y} \mapsto \tilde{Y}M$ (Exchange, ECL, Traffic, ETTh2, WTH); none otherwise |
| Look-back window $L$ | 60 at $H$=24 (ETTm1/m2: 288); $2H$ for $H \in \{168, 336\}$ (Appendix A.4.2) |
| Chunk size $c$ | $c = H$ ($s = 1$, single-shot) in all headline runs; $c < H$ only in the granularity sweeps and ablations (Appendices A.6.5, A.7.2, A.7.3) |
| Instance normalisation | per channel over $L$: $\tilde{X} = (X - \mu_x)/\sigma_x$, $\sigma_x = \mathrm{Std}(X)$, anchor $\mu_x \in \{\text{none, last, mean}\}$ selected on validation; no learnable parameters |
| Trainable parameters | $W$, $W_\Psi$, $\mathbf{\Phi}_l$ (and $M$ where enabled): $LD + Dc + DF$ $(+F^2)$ |
| *Objective and optimiser (Algorithm 1, Appendix A.3)* | |
| Losses | $\mathcal{L}_s = \mathrm{MSE}(\tilde{Y}, Y)$ updates $(\mathbf{\Phi}_s, \mathbf{\Psi})$ and $M$ where enabled; $\mathcal{L}_l = \mathrm{MSE}(\tilde{Y}, Y)$ updates $\mathbf{\Phi}_l$ only |
| Optimiser | AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $\lambda = 2 \times 10^{-3}$ |
| Gradient clipping | global norm $\kappa = 1.0$ ($\kappa = \infty$ in the synthetic stability sweeps, Appendix A.7.2) |
| *Warm-up (offline, training split)* | |
| Epochs / stride | $E = 1$, stride 1 (ECL: 15 epochs) |
| Learning rate $\eta_0$ | $10^{-3}$ (ECL: $2 \times 10^{-3}$) |
| *Online phase (test split; Appendix A.4.2)* | |
| Window stride | $H$ (non-overlapping windows, each target seen once) |
| Online learning rate $\eta$ | selected from $\{10^{-4}, 10^{-3}, 10^{-2}\}$ on the validation split at $H$=24, reused for all $H$; the same $\eta$ is applied to every parameter group |
| Updates per window | one ChunkedPass with one AdamW step per chunk ($n_{\text{inner}} = 1$); ECL additionally takes 10 inner steps on $W_\Psi$ only (see below) |
| Teacher forcing | $p_{\text{tf}} = 1$ (scheduled sampling disabled; irrelevant at $s = 1$) |
| Seeds | 4 (2023–2026); mean reported |

Table 11: Per-dataset settings of SPOT−VSA, selected on the validation split at $H$=24 and reused for $H \in \{168, 336\}$.

| Dataset | $D$ | Channel mixing | Anchor $\mu_x$ | Online $\eta$ |
|---|---|---|---|---|
| ETTh1 | 2000 | – | mean | $10^{-3}$ |
| ETTh2 | 500 | ✓ | last | $10^{-3}$ |
| ETTm1 | 500 | – | last | $10^{-3}$ |
| ETTm2 | 500 | – | none | $10^{-4}$ |
| WTH | 500 | ✓ | last | $10^{-3}$ |
| Exchange | 500 | ✓ | last | $10^{-2}$ |
| ECL | 500 | ✓ | last | $10^{-3}$ |
| Traffic | 500 | ✓ | none | $10^{-4}$ |

(last, $10^{-3}$), ETTm1 (last, $10^{-3}$), ETTm2 (none, $10^{-4}$), WTH (last, $10^{-3}$), Exchange (last, $10^{-2}$), ECL (last, $10^{-3}$), and Traffic (none, $10^{-4}$).

On ECL, validation selection also incorporates ten inner gradient iterations restricted exclusively to the linear projection $W_\Psi$ on the revealed window. All synthetic streams, theoretical evaluations, and ablations in Appendices A.7.2 and A.7.3 disable these additional iterations, enforcing strictly one update per chunk ($n_{\text{inner}} = 1$).

### A.6.4 Baseline Implementation Details

Baselines are implemented via their official codebases and evaluated under identical splits, scalers, look-back contexts, and horizon-strided execution schemes across four random seeds. Input normalization schemes are summarized in Table 12.

Table 12: Input normalization configurations across evaluated models. Instance-level statistics are computed over the look-back $L$ per channel and inverted on output forecasts.

| Model Family | Instance Normalization | Statistical Moments |
|---|---|---|
| SPOT−VSA | Yes (Validation-selected) | Anchor $\mu_x \in \{\text{none, last, mean}\}$; Scale $\sigma_x = \mathrm{Std}(X)$ |
| iTransformer (+ Modular Adapters) | Yes | Mean $\mu = \mathrm{Mean}(X)$; Scale $\sigma = \mathrm{Std}(X)$ |
| Adapters on SPOT−VSA | Yes (Validation-selected) | Identical to native SPOT−VSA |
| PatchTST (Offline) | Yes (RevIN without affine) | Mean $\mu = \mathrm{Mean}(X)$; Scale $\sigma = \mathrm{Std}(X)$ |
| CoSPOT | Yes (Without affine) | Mean $\mu = \mathrm{Mean}(X)$; Scale $\sigma = \mathrm{Std}(X)$ |
| DLinear (Offline) | No | None (Global standardization only) |
| Continual Baselines (FSNet, OneNet, etc.) | No (As published) | None (Global standardization only) |

**Continual forecasting architectures.** FSNet (Pham et al., 2023), OneNet, and OneNet-TCN (Zhang et al., 2023) follow their published training configurations (6 warm-up epochs for FSNet, 15 for OneNet; $\eta_0 = 10^{-3}$, increasing to $3 \times 10^{-3}$ on ECL and Traffic). Replay baselines (ER and DER++) implement the FSNet TCN backbone with FIFO episodic replay buffers and logit distillation. LSTD (Cai et al., 2025) is evaluated with network depth 10 and hidden dimension 512. All continual baselines execute one gradient update per stride-$H$ window.

**Static offline forecasters.** PatchTST (Nie et al., 2022) and DLinear (Zeng et al., 2023) are trained on the warm-up split for 20 epochs with validation early stopping (patience 3, $\eta_0 = 10^{-3}$). During the online streaming evaluation, their parameters remain fixed.

**CoSPOT.** CoSPOT (Choi et al., 2026) instantiates a frozen GPT-2 backbone (6 layers, $d = 768$) with self-supervised test-time linear probing. As the released implementation evaluates only channel 0, we modified the data pipeline to evaluate all multivariate series via iterative channel-independent passes, reporting mean performance across two seeds.

**iTransformer backbone and modular adapters.** The iTransformer (Liu et al., 2024) backbone is configured following the ADAPT-Z benchmark specifications (Huang et al., 2026): 2 layers ($d = 128$) on ETT and Exchange; 3 layers ($d = 512$) on WTH and ECL; and 4 layers ($d = 512$) on Traffic. We evaluate the following modular adaptation algorithms:

- **From the ADAPT-Z codebase:** Static execution (**Ori**); online SGD across all parameters (**OGD**, $\eta = 3 \times 10^{-6}$); feature-space adaptation on the final encoder layer (**fOGD**, $\eta = 10^{-3}$); output adapter optimization (**ADCSD**, $\eta = 10^{-4}$); and meta-learned adapter modulation (**ADAPT-Z**). Updates are computed over a historical replay buffer of the 24 most recent fully revealed windows.
- **From the OnlineTSF codebase:** Full-network AdamW streaming updates (**OGD**$^{\dagger}$); context-conditioned parameter generation (**Proceed**); and selective test-time exemplar adaptation (**SOLID**), evaluated under stride-$H$ constraints.
- **DSOF:** Consists of an iTransformer teacher model coupled with a lightweight MLP student trained via residual experience replay and temporal difference loss. Because DSOF uses an intrinsically sample-wise buffer, it updates natively at stride 1 while being scored strictly on the stride-$H$ boundary (executing $H\times$ more gradient steps than competing baselines).

These adaptation techniques are transferred to `SPOT-VSA` using structurally equivalent interfaces: feature-space methods adapt the latent vector representation $Z = \mathbf{\Phi}_s(\bar{X}) \odot \mathbf{\Phi}_l$, and adapter baselines modulate the linear projection matrix $W_\Psi$.

**Bug fixes in baseline implementations.** Several implementation issues in public baseline repositories were resolved:

1. In the ADAPT-Z codebase, checkpoint paths were hardcoded to identical filenames across worker threads, corrupting concurrent executions; checkpoints were refactored to use unique per-run process hashes.
2. A diagnostic stride-1 post-evaluation pass in ADAPT-Z that accumulated forward activation graphs in memory was bypassed to prevent system out-of-memory crashes.
3. Normalization discrepancies between pre-training (`norm_style = RevIN`) and inference (`norm_style = none`) in ADAPT-Z were harmonized.
4. Missing learning rate assignments for unlisted datasets in OnlineTSF were imputed from topologically adjacent benchmarks, and commented-out pre-training initializations were restored.

### A.6.5 Component Ablation Setup

The systematic component ablation (480 configurations across four seeds on ETTh1-24, ETTm1-24, ETTh1-168, S-A-L, and S-G-L) crosses three architectural axes:

- **Chunk size:** $c \in \{H, H/2, H/4, 1\}$ (spanning single-shot to fully autoregressive).

- **Inductive bias module:** Active ($\mathbf{\Phi}_l \in \mathbb{R}^{D\times F}$ trained via $\mathcal{L}_l$) versus disabled ($\mathbf{\Phi}_l = \mathbf{1}$).
- **Binding projection:** frozen random Gaussian anchor $P$ (quasi-orthogonal rows) versus an all-ones matrix ($P = \mathbf{1}$, no binding) versus a trainable $P$.

These ablations are conducted at fixed learning rate $\eta = 10^{-3}$ with instance normalization disabled to isolate architectural effects.

### A.6.6 Latent-Size Ablation

We sweep $D \in \{50, 100, 250, 500, 1000, 2000, 5000\}$ on eight cells under the strict protocol, with the validation-selected anchor and step size of each cell, the single-shot setting ($s = 1$), one update per window and no inner iterations, over three seeds. The model has $LD + D + DH$ parameters ($+F^2$ with the channel mixer), so the sweep spans two orders of magnitude in size (e.g. 4.2k to 425k parameters on ETTh1-24). Table 13 reports the cumulative online MSE and the forgetting ratio.

Table 13: Latent-size ablation: online MSE (top) and forgetting ratio FR (bottom), mean over three seeds; $D = 500$ is the headline size. The standard deviation of the MSE is below 2% of the mean except on ETTm2-168 (up to 4%) and Exchange (up to 8%; 33% at Exchange-168, $D = 500$, marked $^*$, where one seed diverges at $\eta = 10^{-2}$); that of FR is at most 0.011 except on Exchange-24 (up to 0.08) and Exchange-168 (up to 0.40).

| | Cell | $D$=50 | 100 | 250 | 500 | 1000 | 2000 | 5000 |
|---|---|---|---|---|---|---|---|---|
| MSE | ETTh1-24 | 0.676 | 0.673 | 0.670 | 0.669 | 0.669 | 0.670 | 0.672 |
| | ETTm1-24 | 0.269 | 0.269 | 0.270 | 0.270 | 0.270 | 0.270 | 0.270 |
| | Exchange-24 | 0.118 | 0.118 | 0.119 | 0.132 | 0.128 | 0.124 | 0.132 |
| | ECL-24 | 5.997 | 5.941 | 6.007 | 6.007 | 6.206 | 6.236 | 6.239 |
| | WTH-24 | 0.350 | 0.350 | 0.350 | 0.350 | 0.350 | 0.350 | 0.350 |
| | ETTh1-168 | 1.071 | 1.080 | 1.091 | 1.107 | 1.118 | 1.130 | 1.137 |
| | Exchange-168 | 0.683 | 0.719 | 0.780 | 1.366$^*$ | 0.793 | 0.828 | 0.833 |
| | ETTm2-168 | 1.139 | 1.132 | 1.162 | 1.184 | 1.205 | 1.227 | 1.270 |
| FR | ETTh1-24 | 1.027 | 1.036 | 1.049 | 1.060 | 1.069 | 1.079 | 1.094 |
| | ETTm1-24 | 1.068 | 1.068 | 1.063 | 1.055 | 1.045 | 1.039 | 1.027 |
| | Exchange-24 | 1.183 | 1.163 | 1.148 | 1.283 | 1.275 | 1.158 | 1.174 |
| | ECL-24 | 1.287 | 1.271 | 1.256 | 1.238 | 1.228 | 1.214 | 1.210 |
| | WTH-24 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | ETTh1-168 | 1.004 | 1.004 | 1.005 | 1.008 | 1.012 | 1.018 | 1.038 |
| | Exchange-168 | 1.133 | 1.193 | 1.461 | 1.128 | 1.171 | 1.484 | 1.907 |
| | ETTm2-168 | 1.060 | 1.061 | 1.062 | 1.057 | 1.049 | 1.041 | 1.030 |

**Accuracy.** At $H = 24$ the latent size is inert over two orders of magnitude: the MSE changes by about 1% on ETTh1 and ETTm1, not at all on WTH, and by 4% on ECL, where it is the smaller sizes that are better; on Exchange-24 the sizes $D \leq 250$ are about 10% better than $D = 500$, a difference comparable to the seed spread at $\eta = 10^{-2}$. At $H = 168$ smaller is consistently better: from $D = 50$ to $D = 5000$ the MSE rises from 1.071 to 1.137 on ETTh1, from 1.139 to 1.270 on ETTm2 and from 0.683 to 0.833 on Exchange, where the headline $D = 500$ is an unstable point at $\eta = 10^{-2}$ (one divergent seed). Under the strict protocol the $D \times H$ readout of a long horizon is fit from a few dozen windows, so fewer parameters regularize. The accuracy of `SPOT-VSA` therefore does not come from the size of the latent space, and $D = 500$ is not a tuned choice: with $D = 50$, a tenth of the parameters, the MSE would change by about 1% at $H = 24$ and improve at $H = 168$.

**Forgetting.** The $1/\sqrt{D}$ cross-talk scale of Lemma A.1 would suggest that forgetting decreases with $D$. This holds on three cells (ETTm1-24: 1.068 → 1.027; ECL-24: 1.287 → 1.210; ETTm2-168: 1.060 → 1.030), but FR *increases* with $D$ on three others (ETTh1-24: 1.027 → 1.094; ETTh1-168: 1.004 → 1.038; Exchange-168: 1.13 → 1.91), is identically 1 on WTH-24 and is within noise on Exchange-24. A larger $D$ reduces cross-talk between lag latent vectors but also

enlarges the trainable readout, which moves more per update; which effect dominates depends on the stream. We therefore do not claim that $D$ is a stability knob.

**Latent size and online learning rate.** Table 14 crosses the two extreme sizes with the online learning rate $\eta$. A larger latent space moves the best $\eta$ down by half a decade to a decade (Exchange-168: from $\eta = 10^{-2}$ at $D = 50$ to the $10^{-4}$–$3\times10^{-3}$ plateau at $D = 5000$; ETTm1-24 and ETTm2-168: from $3 \times 10^{-3}$ to $10^{-3}$) and makes large $\eta$ far more harmful (at $\eta = 10^{-1}$: 213 against 2.28 on ETTh1-168), consistent with the fact that under Adam the change of the output per update grows with the number of readout coordinates. Even with each size at its own best $\eta$, however, the small model remains at least as accurate at $H = 168$ (1.071 against 1.137 on ETTh1, 1.032 against 1.063 on ETTm2, 0.683 against 0.770 on Exchange), so the advantage of small $D$ at long horizons is not only a learning-rate effect. The table also shows that the $\eta$ selected on the validation stream at $H = 24$ is not always the best on the evaluation stream (Exchange-24: $10^{-2}$ selected, $10^{-3}$ best), which is the price of selecting on a short validation stream.

Table 14: Latent size $D$ × online learning rate $\eta$: cumulative online MSE, mean over three seeds ("div": diverged). Best $\eta$ per row in bold.

| | | Online learning rate $\eta$ | | | | | |
|---|---|---|---|---|---|---|---|
| Cell | $D$ | $10^{-4}$ | $10^{-3}$ | $3\times10^{-3}$ | $10^{-2}$ | $3\times10^{-2}$ | $10^{-1}$ |
| ETTh1-24 | 50 | 0.705 | **0.676** | 0.682 | 0.708 | 0.765 | 1.01 |
| | 5000 | 0.682 | **0.672** | 0.698 | 0.754 | 1.02 | 3.68 |
| ETTm1-24 | 50 | 0.290 | 0.269 | **0.268** | 0.283 | 0.307 | 0.42 |
| | 5000 | 0.318 | **0.270** | 0.282 | 0.327 | 0.640 | 31.3 |
| WTH-24 | 50 | 0.364 | **0.350** | 0.351 | 0.364 | 0.401 | 0.62 |
| | 5000 | 0.364 | **0.350** | 0.354 | 0.372 | 0.504 | 38.8 |
| ECL-24 | 50 | 7.44 | 6.00 | **5.89** | 7.00 | 12.5 | div |
| | 5000 | 7.41 | 6.24 | **5.92** | 9.65 | 456 | div |
| Exchange-24 | 50 | 0.105 | **0.103** | 0.106 | 0.118 | 0.121 | 0.55 |
| | 5000 | 0.106 | **0.105** | 0.114 | 0.132 | 0.378 | 15.0 |
| ETTh1-168 | 50 | 1.105 | **1.071** | 1.095 | 1.193 | 1.363 | 2.28 |
| | 5000 | 1.267 | **1.137** | 1.213 | 1.640 | 2.080 | 213 |
| ETTm2-168 | 50 | 1.139 | 1.037 | **1.032** | 1.170 | 1.700 | 4.11 |
| | 5000 | 1.270 | **1.063** | 1.176 | 1.562 | 4.913 | 472 |
| Exchange-168 | 50 | 1.017 | 0.797 | 0.703 | **0.683** | 0.806 | 2.64 |
| | 5000 | 0.774 | 0.775 | **0.770** | 0.833 | 2.621 | div |

**Selection.** $D$ is not among the validation-selected hyperparameters (only the anchor and the step size are), and all headline results use the fixed value of Table 10. Table 13 is computed on the evaluation stream and cannot be used to choose $D$; selecting $D$ on the validation stream is left for future work.

### A.7 Latency and energy measurement Protocol

All platforms measure the same unit of work: one online window, i.e. forecasting the next $H$ steps, scoring the forecast and taking the gradient update on that window. After untimed warm-up windows, each configuration is timed on consecutive windows for a fixed budget of about 90 s (60 s on the A6000), and we report the median per-window latency. Power is sampled at 20 Hz by a background thread throughout the timed region, and energy per window is the mean latency multiplied by the mean power over that region. An idle baseline is recorded immediately before each run, so the marginal (dynamic) power above idle is also available. On the **RTX A6000**, power is read from the GPU's on-board sensor through NVML; it therefore covers the GPU board only, not the host CPU. Timing calls `torch.cuda.synchronize` after every window so that kernel execution, not just kernel launch, is measured, and the idle baseline is taken after a 12 s settling period because GPU power decays slowly after the previous job. On the **Jetson Nano**, power is read from the board's INA3221 monitor: the 5 V input rail provides the reported whole-board figure, and the GPU and CPU rails are logged alongside it. Clocks are locked to their maximum with `jetson_clocks` and the desktop is stopped to avoid interference. On the **ZCU104** (quad Cortex-A53, PyTorch on CPU with 4 threads, CPU frequency fixed at 1.2 GHz), power is read from the

INA226 monitor on the board's 12 V input rail; the per-rail readings of the two on-board power regulators are recorded as well. The Jetson and ZCU104 figures are therefore whole-board energy, including the static idle floor, whereas the A6000 figure is GPU-only.

### A.7.1 Online Linear Baseline

To test whether the accuracy of SPOT−VSA could be obtained with a plain linear forecaster adapted online, we run DLinear (Zeng et al., 2023) under the strict protocol with one AdamW step per window, in two variants (Table 15). *Same hyperparameters*: DLinear uses the look-back, warm-up schedule, optimizer settings (weight decay $2 \times 10^{-3}$, clipping at 1), and step size ($10^{-3}$ for warm-up and online updates) of SPOT−VSA, over four seeds. *Own recipe*: DLinear is warmed up for 20 epochs with early stopping, as for the offline model, and its online step is the best per cell on the evaluation stream over a grid reaching down to $3 \times 10^{-5}$ (three seeds); since the step is chosen on the evaluation stream, this column is an optimistic bound for the baseline and is not comparable in kind to the validation-selected SPOT−VSA column.

Table 15: Online DLinear versus SPOT−VSA under the strict protocol (cumulative online MSE; best per row in bold). "Offline": trained on the warm-up split and frozen. "Same hyperparameters": the warm-up, optimizer settings and step size of SPOT−VSA. SPOT−VSA: validation-selected, as in Table 2 (which reports RMSE).

| | | DLinear | | |
|---|---|---|---|---|
| Dataset | $H$ | Offline | Online, same hyperparameters | SPOT−VSA |
| ETTh1 | 24 | 0.754 | 0.704 | **0.670** |
| | 168 | 1.208 | 1.616 | **1.130** |
| | 336 | 1.842 | 3.008 | **1.371** |
| ETTh2 | 24 | 2.277 | 2.079 | **1.622** |
| | 168 | **8.180** | 20.32 | 10.25 |
| | 336 | **16.72** | 58.33 | 28.68 |
| ETTm1 | 24 | 0.321 | 0.658 | **0.270** |
| | 168 | 0.733 | 0.940 | **0.628** |
| | 336 | 1.085 | 1.965 | **0.940** |
| ETTm2 | 24 | 0.785 | 0.946 | **0.700** |
| | 168 | 1.266 | 1.576 | **1.185** |
| | 336 | **1.902** | 3.438 | 4.131 |
| WTH | 24 | 0.366 | 0.359 | **0.350** |
| | 168 | **0.515** | 0.745 | 0.527 |
| | 336 | **0.552** | 1.022 | 0.754 |
| Exchange | 24 | 0.170 | 0.138 | **0.129** |
| | 168 | 1.488 | 1.960 | **1.331** |
| | 336 | 7.069 | 21.10 | **1.878** |
| ECL | 24 | 130.9 | 7.361 | **5.167** |
| | 168 | 251.7 | 85.10 | **14.87** |
| | 336 | 255.2 | 1294 | **22.27** |
| Traffic | 24 | 0.727 | **0.484** | 0.500 |
| | 168 | 0.606 | 0.522 | **0.407** |

With identical hyperparameters SPOT−VSA is more accurate than the online linear model on 21 of the 23 cells (the exceptions are ETTm2-336 and Traffic-24), typically by 20–50% and by large factors at long horizons (ECL-168: 14.9 against 85.1; Exchange-336: 1.88 against 21.1; ETTh1-336: 1.37 against 3.01), where the linear model becomes unstable at this step size; its forgetting ratio reaches 2.9–3.3 on ETTh1-168 and Exchange-24, against at most 1.3 for SPOT−VSA on the same cells. The linear model needs a step 10–30× smaller than SPOT−VSA to maintain stability ($3 \times 10^{-5}$–$10^{-4}$ is best almost everywhere, and it diverges at $10^{-2}$), whereas SPOT−VSA is nearly insensitive to the step between $10^{-4}$ and $3 \times 10^{-3}$ (Table 14). Both observations agree with Theorem 2.3: under one update per window, streams on which noise dominates effective drift reward small steps and small, well-regularized models.

### A.7.2 Stability–Plasticity Experimental Setup

To validate the theoretical bounds of Section 2.8, we evaluate SPOT−VSA on synthetic AR(1) drift sequences under delayed-feedback stride-$H$ execution with gradient clipping disabled ($\kappa = \infty$). Across $D = 500$ without input normalization, we sweep:

- Prediction horizon: $H \in \{24, 48\}$;
- Chunk factor: $s \in \{1, 2, 4, 8\}$ where $c = H/s$;
- Learning rate: $\eta \in \{1 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}, 1 \times 10^{-2}\}$;
- Drift intensity: $q \in \{0$ (stationary), S-G-L (mild), S-A-L (severe), $4q, 10q\}$.

This yields 320 independent configurations over two seeds. Plasticity is quantified via cumulative online MSE; stability is measured via function-space divergence $\Delta_t$, computed once per window, i.e. after the $s$ optimizer steps of the learning pass. Continuous optimal step sizes $\eta^\star$ are computed by fitting a second-order polynomial in $\log \eta$ space around the discrete empirical minimum.

### A.7.3 Chunk-Size Sweeps on Real-World Datasets

Chunk resolution is swept on Exchange-24, ECL-24 ($c \in \{24, 12, 8, 6, 4, 3, 2, 1\}$), ETTh1-168 ($c \in \{168, 84, 56, 42, 28, 21, 14, 7, 4, 2, 1\}$), and Exchange-336 and ECL-336 ($c \in \{336, 168, 112, 84, 56, 42, 28, 21, 14, 7\}$) across three seeds under validation-selected settings ($n_{\text{inner}} = 1$). Exchange-336 is additionally evaluated at $\eta = 10^{-3}$ to decouple chunk discretization dynamics from gradient divergence at $\eta = 10^{-2}$. Under stride-336, Exchange-336 contains 16 windows; probe evaluations therefore encompass the complete sequence and measure sequence-wide stability rather than localized catastrophic forgetting.

### A.7.4 Extended Granularity Dynamics and Breakdown

Table 16 documents the full empirical granularity sweep across synthetic drift streams and diverse real-world benchmarks, reporting empirical optimal granularities $s^\star$, MSE improvements, full autoregressive inflation, functional drift growth, and resulting collapse modes.

Table 16: **Empirical granularity dynamics from Figure 4.** $s^\star$ denotes the empirical minimizer of online MSE over the search grid. (†) An interior minimum on synthetic streams emerges only when $\eta \geq 3 \times 10^{-3}$; smaller step sizes yield a monotonic optimum at $s = 1$. The stability column reports the growth ratio in $\Delta$ (synthetic) or the shift in FR (real) over the interval $s \in [1, 8]$.

| **Stream** | $H$ | $s^\star$ | $\frac{\text{MSE}(s^\star)}{\text{MSE}(1)}$ | $\frac{\text{MSE}(s{=}H)}{\text{MSE}(1)}$ | **Stability** ($s{=}1 \to 8$) | **Failure Mode** ($s \gg 1$) |
|---|---|---|---|---|---|---|
| S-A | 24 | 2–4† | 0.98 | – | $\Delta$: ×6–37 | Interference ($\gamma^{2s}$) |
| S-G | 24 | 2–4† | 0.99 | – | $\Delta$: ×5–29 | Interference ($\gamma^{2s}$) |
| ETTh1 | 168 | 3 | 0.97 | 3.98 | FR : 1.02 → 1.07 (flat) | Error compounding (stable rollout) |
| Exchange | 24 | 2 | 0.87 | div. | FR : 1.28 → 1.03 (↓) | Autoregressive divergence at $c{=}1$ |
| ECL | 24 | 2 | 0.95 | 2.01 | FR : 1.24 → 1.39; BWT : 0.04 → 0.09 | Interference, followed by divergence |

**Bound scope and empirical boundaries.** On Exchange-24, the forgetting ratio *decreases* with chunk count, moving from 1.28 at $s = 1$ to $\approx 1.0$ for $s \geq 2$. Under an aggressive step size ($\eta = 10^{-2}$), single-pass updates on full 24-step horizons cause the model to overfit local window dynamics, inducing parameter drift on historical probes. Partitioning horizons into smaller targets regularizes gradient updates and mitigates this localized drift—an effect outside the scope of our theoretical formulation. Similarly, an interior optimum $s^\star$ appears on synthetic benchmarks only at higher learning rates ($\eta \geq 3 \times 10^{-3}$). Because the synthetic chunk error $\delta(c)$ is nearly invariant ($\alpha \approx 0$), the theoretical bound predicts monotonic degradation in $s$; at elevated step sizes, executing $s$ sub-updates per window averages stochastic gradient trajectories and effectively damps the learning rate. These subtleties highlight unmodeled higher-order factors rather than contradictions of the primary bounds, aligning with the theoretical assumptions in Section 2.8.

**Online update techniques on the SPOT−VSA backbone.** To separate the effect of the online update rule from that of the model, we keep the SPOT−VSA backbone fixed and replace only its online

Table 17: Online update techniques on the SPOT−VSA backbone (strict online protocol: stride $= H$, one update per window; MSE, mean over 4 seeds unless marked). All rows start from the same warm-up checkpoint and are scored on the same windows; they differ only in how the model is updated online (ADAPT-Z and Proceed additionally fit their adaptation modules on the validation split, as released). Best per column in **bold**, second underlined.

| $H$ | Technique | ETTh1 | ETTh2 | ETTm1 | ETTm2 | WTH | Exchange | ECL | #best |
|---|---|---|---|---|---|---|---|---|---|
| 24 | Frozen (no update) | 0.737 | 2.310 | 0.350 | 0.815 | 0.375 | 0.110 | 41.58 | 0 |
| | OGD | 1.145 | 2.309 | 0.350 | 0.815 | 0.375 | 0.110 | 15.39 | 0 |
| | fOGD | 0.744 | 2.375 | 0.356 | 0.729 | 0.372 | 0.111 | 41.71 | 0 |
| | ADCSD | 0.737 | 2.310 | 0.350 | 0.791 | 0.372 | 0.110 | 41.56 | 0 |
| | ADAPT-Z | 0.707 | 1.853 | 0.292 | **0.696** | 0.361 | **0.104** | 12.55 | 2 |
| | SOLID | 0.728 | 2.301 | 0.347 | 0.788 | 0.374 | 0.110 | 41.55 | 0 |
| | Proceed | 0.699 | 1.807 | **0.266** | 0.763 | 0.352 | 0.105 | 19.66 | 1 |
| | SPOT−VSA update (ours) | **0.664** | **1.666** | 0.270 | 0.712 | **0.351** | 0.117 | **5.595** | 4 |
| 168 | Frozen (no update) | 1.440 | 9.493 | 0.673 | 1.216 | 0.561 | 0.852 | 24.34 | 0 |
| | OGD | 1.440 | 9.493 | 0.673 | 1.216 | 0.561 | 0.852 | 14.98 | 0 |
| | fOGD | 1.428 | **9.461** | 0.672 | **1.155** | 0.559 | 0.858 | 23.60 | 2 |
| | ADCSD | 1.439 | 9.493 | 0.673 | 1.215 | 0.561 | 0.852 | 24.34 | 0 |
| | ADAPT-Z | 1.146 | 10.19 | 0.658 | 1.206 | 0.546 | 0.838 | 20.14 | 0 |
| | SOLID | 1.436 | 9.492 | 0.672 | 1.214 | 0.561 | 0.851 | 24.34 | 0 |
| | Proceed | **1.069** | 10.62 | 0.651 | 1.636 | **0.520** | 0.753 | 26.03 | 2 |
| | SPOT−VSA update (ours) | 1.204 | 9.888 | **0.641** | 1.195 | 0.543 | **0.704** | **10.38** | 3 |
| 336 | Frozen (no update) | 1.241 | 15.87 | 0.806 | 1.727 | 0.632 | 2.803 | 28.55 | 0 |
| | OGD | 1.241 | 15.87 | 0.806 | 1.727 | 0.632 | 2.803 | 22.63 | 0 |
| | fOGD | 1.242 | 15.73 | 0.807 | **1.694** | 0.630 | 2.801 | 28.74 | 1 |
| | ADCSD | 1.241 | 15.87 | 0.806 | 1.727 | 0.632 | 2.803 | 28.55 | 0 |
| | ADAPT-Z | 1.432 | 20.33 | 0.954 | 1.805 | 0.616 | 2.133 | 32.48 | 0 |
| | SOLID | 1.240 | 15.87 | 0.806 | 1.726 | 0.632 | 2.801 | 28.55 | 0 |
| | Proceed | **1.153** | 15.52 | **0.769** | 2.345 | **0.556** | **1.618** | 30.89 | 4 |
| | SPOT−VSA update (ours) | 1.154 | **15.21** | 0.782 | 1.705 | 0.610 | 1.743 | **14.64** | 2 |

update with six techniques from the online forecasting literature: plain OGD, feature-space OGD (fOGD), the output-space adapter ADCSD, ADAPT-Z (Huang et al., 2026), SOLID (Chen et al., 2024), and Proceed (Zhao & Shen, 2025). Frozen denotes the model without any online update. Every technique starts from the same warm-up checkpoint, is scored on the same windows, and performs one update per scored window under the strict protocol. Table 17 reports the online MSE. Two observations stand out. (i) Techniques that adapt only a small add-on or use conservative step sizes (OGD, fOGD, ADCSD, SOLID) stay close to the frozen model on most cells, i.e., they adapt little to the incoming stream. (ii) The SPOT−VSA update, a single AdamW step on all parameters, is the most effective rule overall. It achieves the lowest MSE on four of the seven datasets at $H{=}24$ and on three of seven at $H{=}168$. It improves over the frozen model on all but one dataset at $H{=}24$ and $H{=}168$ (by up to $87\%$ on ECL), and it is the only technique that does so on every dataset at $H{=}336$. Proceed is the strongest competitor: it is best on four of seven datasets at $H{=}336$, but it falls behind the frozen model on ETTm2 and ECL at that horizon. Because the backbone is small, updating all of its parameters with one well-tuned step is both inexpensive and effective. Note that the SPOT−VSA rows in this table come from the same shared harness as the other techniques and may differ slightly from the headline results in Table 2, which use the validation-selected configuration.

### A.8 Cumulative Stream Progression

Figure 8 gives the remaining twelve dataset–horizon cells. Three patterns are worth noting beyond those of the main text. First, on ETTh2 at $H \in \{168, 336\}$ every method's curve jumps by an order of magnitude at the same stream position (around window 10), a regime change in the series rather than a property of any model; what distinguishes the methods is the level they settle at afterwards, where SPOT−VSA is lowest, i.e. it re-adapts after the shift rather than being immune to it. Second, the two cells where SPOT−VSA is clearly not the best are visible as such: on ETTm2-336 (j) its curve rises to $\approx 4$ in the first ten windows and only partly recovers, and on ECL at $H \in \{24, 168\}$ (d, h) it plateaus above the offline models and OneNet-TCN; these are the same cells the tables mark, and the curves show the loss is incurred early in the stream and not recovered, which points to the $D = 500$ representation rather than to the online updates. Third, on the low-drift, low-channel cells (ETTh1-24, WTH-24, ETTm2-24, WTH-336) all curves converge to within a few percent of each

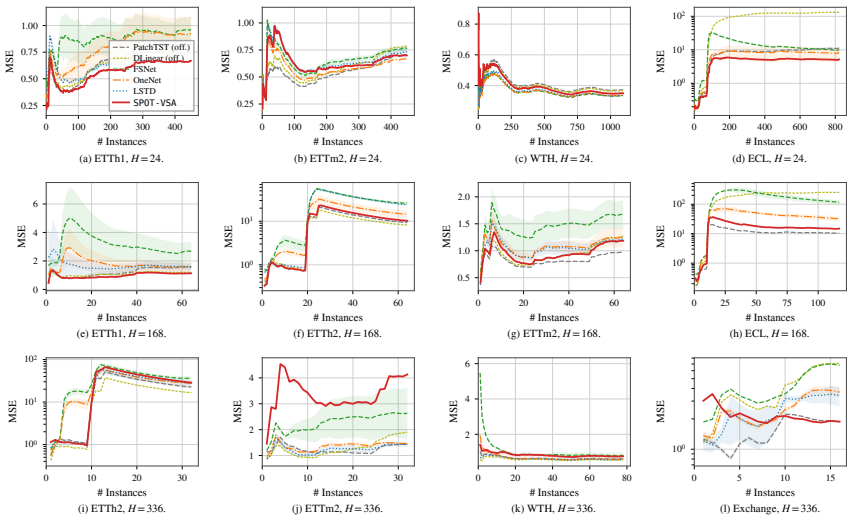



Figure 8: Cumulative average MSE for different forecast horizons $H \in \{24, 168, 336\}$ for the rest of the baselines and datasets.

other after the first hundred windows, so the differences reported in the tables there come from the transient, and the frozen offline models are competitive once it has passed. Exchange-336 (l) has only 16 windows under stride $= H$ and its curves have not converged; it is included for completeness and should not be read as a ranking.

Figure 9 repeats the cumulative curves for MAE on all 21 cells. MAE is less sensitive to the occasional large miss than MSE, so it separates two things the MSE curves conflate: a model that is usually close but sometimes far, and one that is uniformly off. Under MAE `SPOT-VSA` is the lowest or joint-lowest curve on 17 of the 21 cells, more than under MSE, and on the ETT-24 and WTH cells its advantage over the offline models, marginal in MSE, is visible as a consistent gap from the first windows onward; its errors are many small ones rather than a few large ones, the profile the smoothing behaviour would predict. The ordering among the baselines is also cleaner: FSNet is the highest curve on 13 cells and rises through the stream on the long horizons of ETTh1, ETTm1 and ECL, and OneNet sits between FSNet and LSTD nearly everywhere. Two cells change character. On ETTm2-336 (r) `SPOT-VSA` is still the worst curve, so its loss there is not an outlier effect but a uniform level error, consistent with the representation-capacity reading above. On ECL (g, n, u) the gap to the offline models shrinks under MAE relative to MSE, which says that part of the MSE deficit on that dataset comes from a small number of high-magnitude clients rather than from the bulk of the 321 channels. Exchange-336 (t) again has too few windows to rank.

**Cumulative error along the stream.** Figures 10–12 plot the cumulative online MSE of each technique on the `SPOT-VSA` backbone along the test stream (mean over four seeds, $\pm 1$ standard deviation shaded); the last point of each curve is the corresponding entry of Table 17. The curves of OGD, fOGD, ADCSD, and SOLID lie on top of the frozen model on most datasets, confirming that these rules barely adapt the backbone under their released step sizes. Most differences between the remaining techniques are decided at distribution shifts, visible as jumps shared by all curves. After such a shift, the `SPOT-VSA` update settles at a clearly lower error on ECL (all horizons), ETTh2 ($H$=24), Traffic ($H$=24 and 168), and Exchange ($H$=168), and the gap then widens as the stream continues, which is the behaviour expected from a rule that re-fits the whole model at every window. The picture is not uniform. On Exchange at $H$=24, our update drifts above the other techniques after the shift near window 120 and stays there. Proceed is lower for most of the stream on ETTm1 at $H$=24 and on ETTh1 and WTH at $H$=168 and 336, and at $H$=336 it also leads on Exchange; on ETTh1 at $H$=336 our update catches up with it only at the end (1.154 vs. 1.153). Proceed in turn degrades steadily on ETTm2 at $H$=168 and 336 once the series drifts in the second half of the stream and ends above the frozen model, whereas our update stays below the frozen model there. ADAPT-Z starts from a much higher error on ETTh1 and ETTm1 at $H$=336 and does not recover, possibly because its adapter is pre-trained on the validation split.

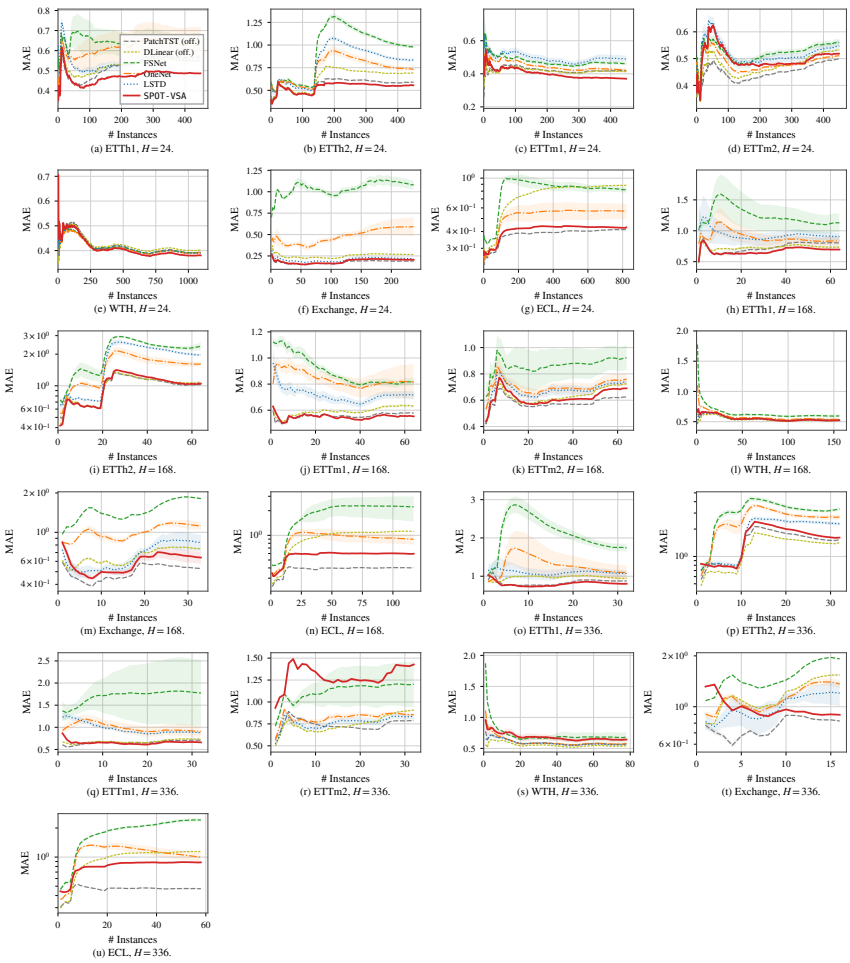



Figure 9: Cumulative average MAE for different forecast horizons $H \in \{24, 168, 336\}$ for the rest of the baselines and datasets.

### A.9 Implementation

Below we give the implementation of `SPOT-VSA` in PyTorch. It follows the released code line by line; shapes use $T$ (look-back), $F$ (channels), $D$ (hypervector dimension), $H$ (horizon) and $C$ (chunk size, $C = H$ in all headline results).

**Binding encoder.** The encoder binds a learnable weight matrix $W$ with a frozen random ID matrix $P$ (element-wise product) and normalises the result by its Frobenius norm. Only $W$ is trained; $P$ is a buffer.

Listing 1: Binding encoder $\Phi_s$ with a frozen ID matrix

```
from dataclasses import dataclass
from typing import Literal, Optional

import torch
import torch.nn as nn
from torch import Tensor


class BindingEncoder(nn.Module):
    """Phi_s: binds learnable weights W with a frozen random ID matrix P.
    """
```

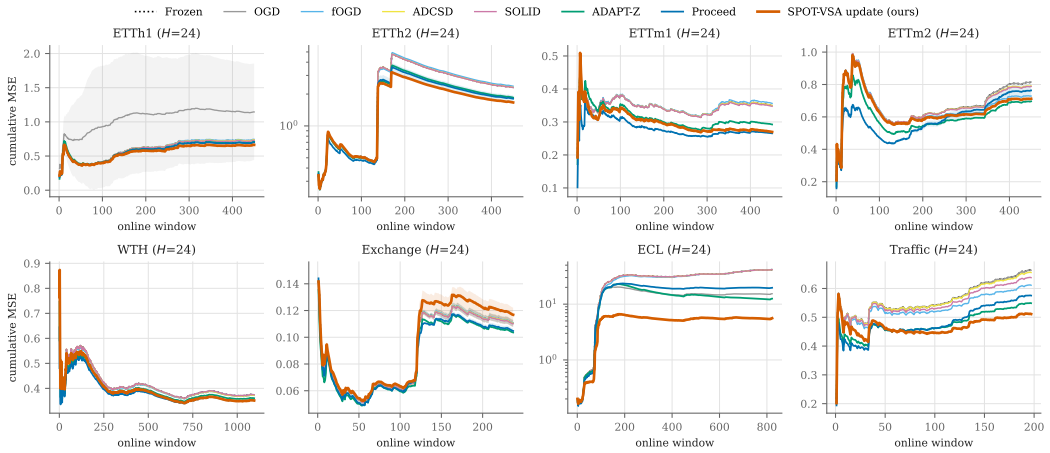



Figure 10: Cumulative online MSE of the online update techniques on the SPOT-VSA backbone, $H$=24 (strict protocol, mean $\pm$ std over 4 seeds; log scale where curves span more than a factor of five). The frozen model (dotted) is largely hidden under OGD, ADCSD, and SOLID. The wide band of OGD on ETTh1 comes from one of the four seeds diverging (MSE 2.37 vs. 0.74 for the others).

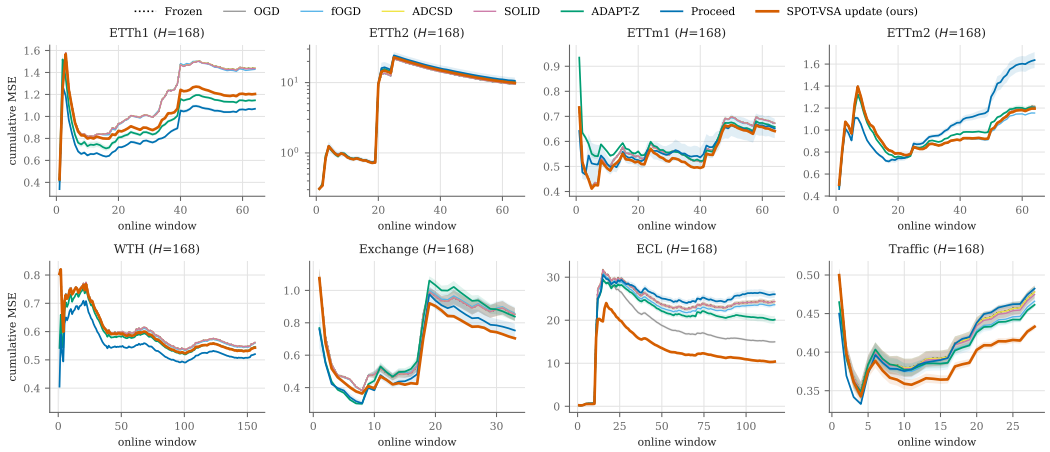



Figure 11: Cumulative online MSE of the online update techniques on the SPOT-VSA backbone, $H$=168 (same setting as Figure 10).

```python
def __init__(self, seq_len: int, hv_dim: int, id_std: float = 0.01)
-> None:
    super().__init__()
    # Frozen random ID matrix P (the hypervector "codebook").
    self.register_buffer("P", torch.randn(hv_dim, seq_len) * id_std)
    # Learnable weights W, small random init.
    self.W = nn.Parameter(torch.randn(hv_dim, seq_len) * 0.1)

def forward(self, z: Tensor) -> Tensor:
    """Map a (T, F) normalised window to (D, F) hypervectors."""
    bound = self.W * self.P                          # binding (Hadamard
product)
    bound = bound / (bound.norm() + 1e-8)            # Frobenius
normalisation
    return bound @ z                                 # project each
channel to D
```

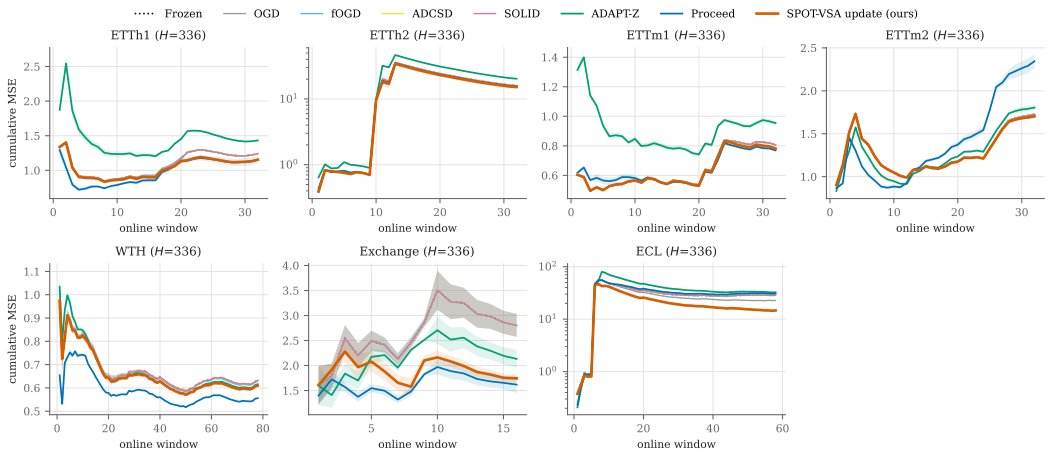



Figure 12: Cumulative online MSE of the online update techniques on the SPOT-VSA backbone, $H$=336 (same setting as Figure 10). Traffic has no $H$=336 runs (no validation windows).

**SPOT-VSA model.** Each look-back window is instance-normalised per channel, lifted into the hyperspace, gated by the learnable inductive bias (EIB) $E$, passed through a SiLU activation, optionally mixed across channels by $M$ (initialised to the identity), decoded to a chunk of $C$ steps, and de-normalised with the same statistics.

Listing 2: SPOT-VSA forecaster (one head, SiLU activation, no dropout)

```
NormMode = Literal["none", "last", "mean"]


class SPOTVSA(nn.Module):
    def __init__(
        self,
        seq_len: int,           # T
        hv_dim: int,            # D
        chunk_len: int,         # C
        n_channels: int,        # F
        instance_norm: NormMode = "last",
        channel_mix: bool = True,
    ) -> None:
        super().__init__()
        self.instance_norm = instance_norm
        self.fc1 = BindingEncoder(seq_len, hv_dim)                # Phi_s
    (encoder)
        self.fc2 = nn.Linear(hv_dim, chunk_len)                   # Psi (
    decoder)
        self.eib = nn.Parameter(torch.ones(n_channels, hv_dim))   # Phi_l
    (EIB gains)
        self.act = nn.SiLU()                                      # HD-
    level activation
        self.chan: Optional[nn.Linear] = None
        if channel_mix:                                           # M,
    identity at init
            self.chan = nn.Linear(n_channels, n_channels, bias=False)
            nn.init.eye_(self.chan.weight)

    def normalize(self, x: Tensor) -> tuple[Tensor, Optional[Tensor],
    Optional[Tensor]]:
        """Return z = (x - a) / s together with the statistics (a, s)."""
        if self.instance_norm == "none":
            return x, None, None
        if self.instance_norm == "last":
            a = x[-1:]
```

```
        else:
            a = x.mean(dim=0, keepdim=True)
        s = x.std(dim=0, keepdim=True) + 1e-5        # unbiased std per
    channel
        return (x - a) / s, a, s

    def forward(self, x: Tensor) -> Tensor:
        """Map a (T, F) look-back window to an (F, C) forecast chunk."""
        z, a, s = self.normalize(x)
        h = self.fc1(z).T * self.eib                 # (F, D), EIB-gated
    hypervectors
        h = self.act(h)                              # SiLU on the gated
    hypervectors
        if self.chan is not None:
            h = self.chan(h.T).T                     # channel mixing: h
    <- M h
        out = self.fc2(h)                            # (F, C)
        if a is not None:
            out = out * s.T + a.T                    # de-normalise
        return out
```

**Optimiser.** A single AdamW optimiser holds three parameter groups: the short-term parameters (encoder and decoder), the EIB, and the channel mixer. Weight decay $\lambda$ is applied only to the first group. During the online phase the learning rate of every group is overwritten by the online rate $\eta$.

Listing 3: Parameter groups and optimiser

```
@dataclass
class ParamGroups:
    short: list[nn.Parameter]      # encoder + decoder (L_s)
    eib: list[nn.Parameter]        # EIB (L_l)
    chan: list[nn.Parameter]       # channel mixer (L_s)
    dec: list[nn.Parameter]        # decoder only (decoder-only passes)


def build_optimizer(
    model: SPOTVSA, lr_warmup: float, lr_chan: float, weight_decay: float
) -> tuple[torch.optim.AdamW, ParamGroups]:
    groups = ParamGroups(
        short=[*model.fc1.parameters(), *model.fc2.parameters()],
        eib=[model.eib],
        chan=[] if model.chan is None else list(model.chan.parameters()),
        dec=list(model.fc2.parameters()),
    )
    opt = torch.optim.AdamW(
        [
            {"params": groups.short, "lr": lr_warmup, "weight_decay":
    weight_decay},
            {"params": groups.eib,   "lr": lr_warmup, "weight_decay":
    0.0},
            {"params": groups.chan,  "lr": lr_chan,   "weight_decay":
    0.0},
        ],
        betas=(0.9, 0.999),
        eps=1e-8,
    )
    return opt, groups


mse = nn.MSELoss()       # short-term loss L_s
mse_l = nn.MSELoss()     # long-term loss L_l (MSE in all reported runs)
CLIP_NORM = 1.0          # global gradient-norm clip
```

**Update on one window.** The window $(X, Y)$ is processed chunk by chunk; each chunk takes one AdamW step. $\mathcal{L}_s$ updates the encoder, decoder and mixer, while $\mathcal{L}_l$ reaches only the EIB. Between chunks the ground truth is fed back (teacher forcing, $p_{\text{tf}} = 1$); with $C = H$ there is a single chunk and a single step.

Listing 4: Training pass on one window (warm-up and online updates)

```
def update(
    model: SPOTVSA,
    opt: torch.optim.Optimizer,
    groups: ParamGroups,
    x0: Tensor,              # (T, F) look-back
    y: Tensor,               # (H, F) ground truth
    chunk_len: int,          # C
    dec_only: bool = False,
) -> None:
    seq_len, horizon = x0.shape[0], y.shape[0]
    x = x0
    for start in range(0, horizon, chunk_len):
        end = min(start + chunk_len, horizon)
        gt = y[start:end]                                    # (end -
    start, F)
```

```
        opt.zero_grad(set_to_none=True)
        pred = model(x).T[: end - start]                        # (end -
    start, F)

        # L_s -> encoder, decoder, mixer (EIB held out of this backward).
        loss_s = mse(pred.reshape(-1), gt.reshape(-1))
        loss_s.backward(
            retain_graph=not dec_only,
            inputs=groups.dec if dec_only else groups.short + groups.chan
    ,
        )
        # L_l -> EIB only.
        if not dec_only:
            loss_l = mse_l(pred.reshape(-1), gt.reshape(-1))
            loss_l.backward(inputs=groups.eib)

        torch.nn.utils.clip_grad_norm_(model.parameters(), CLIP_NORM)
        opt.step()                                              # one step
    per chunk
        x = torch.cat([x, gt], dim=0)[-seq_len:]                # teacher
    forcing
```

**Forecasting.** The scored forecast is produced before the ground truth is revealed; chunks are rolled out autoregressively on the model's own predictions.

Listing 5: Free-running forecast of one window

```
@torch.no_grad()
def forecast(model: SPOTVSA, x0: Tensor, horizon: int, chunk_len: int) ->
     Tensor:
    """Map a (T, F) window to an (H, F) forecast; no ground truth is used
    ."""
    seq_len, n_channels = x0.shape
    x = x0
    y_hat = x0.new_zeros(horizon, n_channels)
    for start in range(0, horizon, chunk_len):
        end = min(start + chunk_len, horizon)
        pred = model(x).T[: end - start]
        y_hat[start:end] = pred
        x = torch.cat([x, pred], dim=0)[-seq_len:]           # feed own
    prediction
    return y_hat
```

**Warm-up.** Before the stream starts, the model is trained offline on the training split with every window (stride 1).

Listing 6: Offline warm-up

```
def warmup(model, opt, groups, train: Tensor, seq_len: int, horizon: int,
           chunk_len: int, n_epochs: int) -> None:
    """train: (N_train, F), z-scored with the training-split statistics."
    ""
    model.train()
    for _ in range(n_epochs):
        for t in range(seq_len, len(train) - horizon):   # stride 1
            update(model, opt, groups,
                   train[t - seq_len : t], train[t : t + horizon],
    chunk_len)
```

**Strict online learning.** The test stream is traversed in non-overlapping windows (stride $H$): each window is forecast, scored, and then used for exactly one update pass ($n_{\text{inner}} = 1$), so every target is seen once and never before it is scored.

Listing 7: Online phase (strict protocol)

```
def run_online(model, opt, groups, stream: Tensor, seq_len: int, horizon:
     int,
               chunk_len: int, eta: float, n_inner: int = 1,
               dec_inner: int = 0) -> tuple[float, float]:
    """eta␣is␣selected␣on␣the␣validation␣split;␣dec_inner␣=␣10␣for␣ECL,␣
    else␣0."""
    for g in opt.param_groups:                    # online rate for every
    group
        g["lr"] = eta

    rmse, mae = [], []
    for t in range(seq_len, len(stream) - horizon, horizon):  # stride H
        x0, y = stream[t - seq_len : t], stream[t : t + horizon]

        # 1. Predict (y not yet revealed).
        y_hat = forecast(model, x0, horizon, chunk_len)

        # 2. Score.
        rmse.append((y_hat - y).pow(2).mean().sqrt())
        mae.append((y_hat - y).abs().mean())

        # 3. Update.
        for _ in range(n_inner):
            update(model, opt, groups, x0, y, chunk_len)
        for _ in range(dec_inner):                # decoder-only passes
            update(model, opt, groups, x0, y, chunk_len, dec_only=True)

    # Reported metrics: mean over windows.
    return torch.stack(rmse).mean().item(), torch.stack(mae).mean().item
    ()
```

**Forgetting ratio.** Retention is measured on a fixed probe: the first 16 windows of the stream are re-forecast (without updating) at checkpoints along the stream.

Listing 8: Forgetting ratio (FR)

```
N_PROBE = 16


def make_probe(stream: Tensor, seq_len: int, horizon: int) -> list[Tensor
    ]:
    stop = seq_len + N_PROBE * horizon
    return [stream[t - seq_len : t + horizon]
            for t in range(seq_len, stop, horizon)]


def probe_mse(model, probe: list[Tensor], seq_len: int,
              horizon: int, chunk_len: int) -> float:
    errs = [
        (forecast(model, w[:seq_len], horizon, chunk_len) - w[seq_len:]).
    pow(2).mean()
        for w in probe
    ]
    return torch.stack(errs).mean().item()


def forgetting_ratio(probe_curve: list[float]) -> float:
    """probe_curve:␣probe_mse()␣recorded␣at␣evenly␣spaced␣checkpoints␣of␣
    the
```

```
    online loop. FR = 1 means no forgetting."""
    return probe_curve[-1] / min(probe_curve)
```