SPS: SUFFICIENT PATCH SELECTION FOR INFORMATION-PRESERVING TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Patch-based tokenization has become a standard design choice for time series forecasting. Most existing methods treat patching as a fixed tokenization step, generating local windows and passing all resulting patches to the forecasting model regardless of their predictive utility. We instead formulate patching as a two-stage process: candidate generation followed by candidate selection. Building on this view, we propose the **Sufficient Patch Selector (SPS)**, a plug-in module for information-preserving patch selection. SPS constructs spectrally guided multi-scale candidate intervals from dominant temporal periods, then learns to retain a compact subset that preserves the predictive distribution induced by the full look-back sequence. To train SPS, we introduce an approximate posterior-preservation objective that matches the selected-patch predictive distribution to a full look-back reference, encouraging retained patches to be sufficient for prediction. We further propose Information Preservation Sufficiency (IPS), a metric that quantifies how well a patching strategy preserves full lookback predictive information. Across standard multivariate forecasting benchmarks and multiple patch-based backbones, SPS improves information preservation and achieves strong forecasting accuracy while retaining fewer patches than existing patching baselines.

1 INTRODUCTION

Long-term time-series forecasting (LTSF) is a fundamental task in domains such as energy, traffic, weather, and industrial monitoring. Given a historical lookback window, the goal is to predict future values over a long horizon, often under strong non-stationarity, multi-scale periodicity, and high cross-variable variability. Recent deep learning architectures have substantially advanced LTSF beyond classical statistical models such as ARIMA (Box and Jenkins, 1970) and earlier recurrent approaches such as LSTM (Hochreiter and Schmidhuber, 1997).

Modern forecasting models have made substantial progress by moving beyond pointwise timestep modeling and instead representing the input history as a sequence of patches (Cirstea et al., 2022; Zhang and Yan, 2023; Nie et al., 2023). The patching approach reduces the effective sequence length, providing local temporal context to each token, and has become a standard component in many competitive LTSF architectures.

Despite the success of patching in time series, it is usually treated as a fixed tokenization rule. Most patch-based forecasting methods define a patch length and stride, generate a set of local windows, and pass all generated patches to the forecasting backbone. This design is simple and effective, but it hides two distinct decisions inside one preprocessing step: which candidate patches should be generated, and which of them should actually be retained for forecasting. These two decisions are not equivalent. If the candidate set is too limited, the model may miss the temporal structures needed for accurate forecasting. If all generated candidates are retained, the representation can contain redundant windows and may spend its token budget on less informative history segments.

Some recent works have started to partially investigate this problem. Adaptive and multi-scale patching methods can enrich the candidate generation process by changing patch scales, boundaries, or periodic structure (Chen et al., 2024; Tang and Zhang, 2025; Ding et al., 2026). However, these methods still process all generated tokens, without explicitly deciding which patches are most in-

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.

formative for forecasting. Conversely, SRSNet (Wu et al., 2025) introduces explicit patch selection for constructing a compact selected representation, but its candidates are generated from fixed-scale local windows. This suggests that the patching design space is only partially explored: existing methods tend to improve either candidate generation or candidate selection, while the joint problem remains underdeveloped.

In this paper, we propose Sufficient Patch Selector (SPS), a plug-in patching framework that jointly addresses candidate generation and candidate selection. SPS first constructs a spectrally guided multi-scale candidate set, using dominant periods from the training signal to define candidate interval lengths. Then, it selects a compact subset of candidate patches under a patch budget. The goal of SPS is to retain a time series representation that is approximately sufficient for predicting future values. In other words, the selected patches should produce a predictive posterior close to the one obtained from the full history.

This idea gives SPS a different objective from ordinary token pruning. Rather than selecting patches only to reduce computation, SPS seeks a compact representation that preserves the predictive information of the full lookback window. We also introduce Information Preservation Sufficiency (IPS), an evaluation metric that compares the predictive distribution induced by a patched representation with that of a full lookback reference. IPS compares patching strategies by how much predictive information they preserve, complementing conventional accuracy metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Our contributions are summarized as follows:

- We present a unified two-stage view of patching for LTSF, separating candidate patch generation from candidate patch selection, and using this view to position existing patch-based forecasting methods.
- We propose SPS, a patching framework that constructs spectrally guided multi-scale candidate intervals and selects a compact subset using an approximate posterior-preserving objective.
- We introduce IPS, a posterior-preservation metric for comparing how much predictive information patching strategies preserve relative to a full lookback reference.
- We evaluate SPS on standard multivariate LTSF benchmarks and across patch-based forecasting backbones, examining both forecasting accuracy and IPS.

2 Rethinking Patching: A Unified View

Representing time series through local subsequences has a long history, including shapelet-based methods that identify discriminative subsequences for time series mining and classification (Ye and Keogh, 2009). In long-term time-series forecasting (LTSF), patching uses local subsequences differently: patches are arranged as token sequences for forecasting rather than selected as standalone discriminative patterns. Inspired by the tokenization strategy of Vision Transformers (Dosovitskiy et al., 2021), early patch-based forecasting models such as Triformer (Cirstea et al., 2022), Crossformer (Zhang and Yan, 2023), and PatchTST (Nie et al., 2023) replace point-wise timestep representations with patch-level representations. By grouping neighboring timesteps, these methods reduce the effective sequence length and allow forecasting models to operate on coarser temporal units. Subsequent works, including CARD (Wang et al., 2024b), retain the same basic patching principle while modifying the forecasting architecture. Meanwhile, xPatch (Stitsyuk and Choi, 2025) adopts PatchTST-style patching in a non-Transformer CNN/MLP-based framework, showing that patching is not tied to Transformer-based backbones (Vaswani et al., 2017).

Despite the success of patch-based modeling, most existing works treat patching as a fixed preprocessing operation or as an architectural detail. In this paper, we argue that patching should instead be viewed as a representation-space construction problem. Specifically, constructing an effective patched representation requires answering two distinct questions: *which candidate patches should be generated?* and *which of these candidates should be retained for forecasting?* This motivates us to propose a unified two-stage view of patching, consisting of candidate patch set generation and candidate patch selection.

2.1 TWO-STAGE FORMULATION OF PATCHING

Let $\mathbf{x} \in \mathbb{R}^L$ denote a univariate input series of historical lookback length L. A patch can be represented as a local subsequence $\mathbf{p}$:

$$\mathbf{p}_{t,\ell} = \mathbf{x}_{t:t+\ell-1}, \tag{1}$$

where t is the starting patch position and ℓ is the patch length. Conventional fixed patching constructs N patches using a predefined patch length P and stride r, producing

$$\mathcal{C}_{\text{fixed}} = \{\mathbf{x}_{1+ir:1+ir+P-1}\}_{i=0}^{N-1}, \qquad N = \left\lfloor \frac{L-P}{r} \right\rfloor + 1. \tag{2}$$

In this case, the candidate set is completely determined by only two hyperparameters: patch length P and stride r.

We formulate patching in a more general way as a two-stage process. Let $\mathbf{x} = (x_1, \ldots, x_L)$ be the full history:

$$\mathcal{C} = \mathcal{G}(\mathbf{x}), \qquad \mathbf{Z} = \mathcal{S}(\mathcal{C}), \tag{3}$$

where $\mathcal{G}$ is the candidate patch set generator, $\mathcal{C} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ is the generated candidate patch set, $\mathcal{S}$ is the candidate patch selector, and $\mathbf{Z}$ is the final patched representation passed to the forecasting model. Generator $\mathcal{G}$ determines the representation space that will be available to the model, while $\mathcal{S}$ controls which of the generated candidate patches will actually be used.

The same formulation clarifies the role of conventional fixed patching. Fixed patching can be interpreted as using a fixed-scale candidate generator followed by an identity selector that retains all generated patches:

$$\mathbf{Z} = \mathcal{S}_{\text{identity}}(\mathcal{C}_{\text{fixed}}) = \mathcal{C}_{\text{fixed}}. \tag{4}$$

Under this view, previous fixed-patching methods are not missing a selection stage. Instead, they instantiate selection as a trivial all-pass rule and retain all N candidates. This distinction matters because the retained patch set determines both the compression level and which temporal structures remain available to the forecasting model.

The two-stage view also connects patching in LTSF to broader questions in representation learning. Candidate generation is related to adaptive tokenization and multi-resolution representation learning (Mallat, 1989; Ryoo et al., 2021). Candidate selection is related to token pruning and sparse computation, where models reduce the number of processed tokens while preserving task performance (Rao et al., 2021). More generally, selecting a compact representation that preserves target-relevant information is closely related to the information bottleneck principle (Tishby et al., 1999; Alemi et al., 2017). However, in forecasting, relevance is defined by predictive information about future targets rather than by reconstruction of the observed input (Still, 2014).

2.2 EXISTING PATCHING METHODS UNDER THE UNIFIED VIEW

The proposed formulation allows existing patching methods to be organized according to how they generate candidate patches and how they utilize them. Figure 1 illustrates this unified view.

Fixed-scale generation with identity selection. Early patch-based models, including Triformer (Cirstea et al., 2022), Crossformer (Zhang and Yan, 2023), PatchTST (Nie et al., 2023), and CARD (Wang et al., 2024b) primarily rely on fixed patch construction. They segment the input history into local subsequences using predefined patching rules and process the resulting tokens with Transformer-based architectures. Similarly, xPatch (Stitsyuk and Choi, 2025) investigates the effectiveness of PatchTST-style patching in a non-Transformer CNN/MLP-based framework. These methods differ substantially in their forecasting backbones, but from the perspective of patching, they share the same high-level structure: generating a fixed-scale candidate patch set and retaining all generated patches. In the proposed taxonomy, these methods represent the fixed-generation with identity-selection corner of the design space.

Adaptive candidate generation with identity selection. A second line of work improves the candidate generation stage by modifying how patches or patch-like tokens are constructed. HDMixer (Huang et al., 2024) enriches patch construction by adjusting segment boundaries through length-extendable patching. Pathformer (Chen et al., 2024) uses multi-scale patch division and

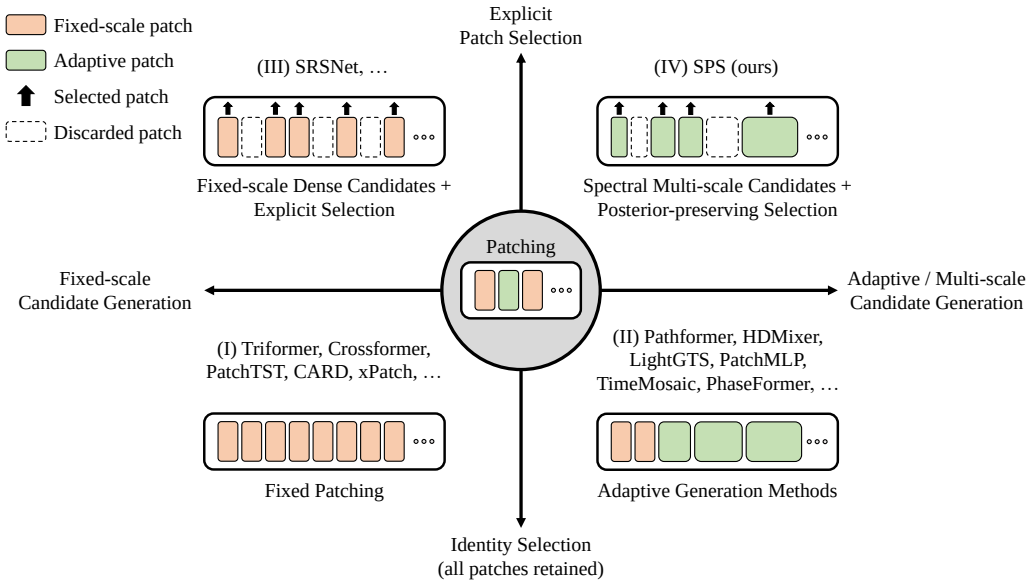


Figure 1: A unified view of patching methods for time-series forecasting. The horizontal axis represents candidate patch set generation, ranging from fixed-scale local windows to adaptive, multi-scale, or data-aware candidate sets. The vertical axis represents candidate patch selection, ranging from identity selection, where all generated patches are retained, to explicit patch-level selection.

adaptive pathways to model temporal patterns at different resolutions. LightGTS (Wang et al., 2025) introduces periodical tokenization based on cycle length to align tokens with periodic features. PatchMLP (Tang and Zhang, 2025) constructs multi-scale patch embeddings using multiple predefined patch lengths. TimeMosaic (Ding et al., 2026) further introduces region-wise adaptive patch granularity, selecting different patch scales for different input regions before token construction. PhaseFormer (Niu et al., 2026) replaces conventional patch tokens with phase tokens (values at the same offset across successive cycles), providing an alternative tokenization strategy.

These methods substantially expand the design space of candidate patch generation by changing patch scale, boundary, periodicity, or token type. However, after constructing these representations, they typically retain all generated tokens. In our unified view, they therefore advance the candidate generation dimension while leaving the selection stage as an identity (or all-pass) rule.

Fixed-scale generation with explicit selection. More recently, SRSNet (Wu et al., 2025) challenges the assumption that all generated patches should be used. It constructs a dense candidate pool by scanning fixed-length local windows with a small stride and then selects informative patches to form a compact selected representation. In our formulation, SRSNet introduces a non-trivial candidate patch selector. However, its candidate generation remains fixed-scale: the candidate family is still composed of local windows with a single predefined length, and density is increased by using a smaller stride. SRSNet therefore moves along the selection axis, but its candidate space remains tied to a single fixed patch scale.

Adaptive generation with explicit selection. The remaining quadrant combines adaptive patch generation with explicit patch selection. This is the region targeted by our proposed Sufficient Patch Selector (SPS). Instead of using a single fixed window length, SPS constructs a spectrally guided multi-scale candidate set and selects a compact subset from it. The selector is trained to preserve the predictive information of the full history under a patch budget, yielding an approximately sufficient patched representation. Thus, SPS addresses both stages of patching: adaptive candidate generation and approximate information-preserving selection. This places SPS in the quadrant where both patch construction and patch retention are explicit design choices.

2.3 Implications for Patch-Based Forecasting

The unified view suggests that improvements in patch-based forecasting can arise from two complementary sources. First, a better generator $\mathcal{G}$ can enlarge or reshape the candidate set, exposing temporal patterns at more appropriate scales. Second, a better selector $\mathcal{S}$ can reduce redundancy by retaining only informative candidates, thereby improving the compactness of the final representation. A rich candidate set without selection may introduce unnecessary redundancy, while selection over a limited fixed-scale candidate set may miss temporal structures that were never generated.

This distinction also provides a structured way to compare patching methods. Conventional fixed patching uses a simple generator and an identity selector. Adaptive patching methods improve the generator but typically retain all generated patches. SRSNet introduces explicit selection, but over a fixed-scale dense candidate family. In contrast, SPS combines spectrally guided multi-scale candidate interval generation with explicit posterior-preserving patch selection. To the best of our knowledge, SPS is the first LTSF patching method to jointly address adaptive candidate generation and explicit information-preserving patch selection within a unified patching framework.

3 Method

3.1 Problem Setup and Overview

Given a multivariate lookback window $\mathbf{X} \in \mathbb{R}^{L\times M}$, where L is the lookback length and M is the number of channels, long-term time series forecasting (LTSF) aims to predict future values $\mathbf{Y} \in \mathbb{R}^{H\times M}$, where H is the prediction horizon. The proposed Sufficient Patch Selector (SPS) follows the two-stage view introduced in Section 2: it first constructs a candidate patch set and then selects a compact subset for forecasting. Since candidate construction and selection are performed independently for each variable in our implementation, we present the method for a univariate sequence $\mathbf{x} \in \mathbb{R}^{L}$ and apply the same operation channel-wise.

SPS consists of four main components, as illustrated in Figure 2. First, the generator constructs a spectrally guided multi-scale candidate set, where interval lengths are derived from dominant periods in the training signal. Second, each variable-length interval is encoded as a fixed-dimensional token with positional information. Third, the selector scores each candidate token using both local patch content and global full history context, and is trained to preserve the predictive information of the full lookback sequence under a patch budget. Finally, the score-masked tokens are compressed through mask-aware attention. As a result, the selected representation is approximately sufficient for prediction: despite being compact, it is trained to preserve the predictive distribution induced by the full history representation.

3.2 Spectrally Guided Candidate Generation

Conventional patching fixes a patch length P and stride r before observing the data. In contrast, SPS uses dominant temporal periods as a data-aware prior over useful patch scales. These periods are estimated from the training split only, avoiding test data leakage. Let $\mathbf{x}_{\text{ref}} \in \mathbb{R}^{T}$ denote a reference signal constructed from the training data. After centering the signal and applying a Hann window to reduce boundary-induced spectral leakage, SPS computes the real Fourier spectrum:

$$\mathbf{a} = \left|\mathrm{rFFT}\left(\mathbf{w} \odot (\mathbf{x}_{\text{ref}} - \mathrm{mean}(\mathbf{x}_{\text{ref}}))\right)\right|^2, \tag{5}$$

where $\mathbf{w}$ is the Hann window and $\mathbf{a}$ denotes the spectral power at each frequency. Frequencies with high spectral power indicate recurring temporal scales. SPS converts the strongest valid spectral peaks into periods $\mathcal{P}_{\text{spec}}$, removes duplicates, and keeps at most B_{spec} period lengths:

$$\mathcal{P}_{\text{spec}} = \{\tau_1, \ldots, \tau_{B_{\text{spec}}}\}, \qquad \tau_j \in [P_{\min}, L]. \tag{6}$$

where $P_{\min}$ is the smallest patch size for excluding very short intervals. This design is motivated by spectral forecastability measures, which relate concentrated spectral density to higher intrinsic predictability (Goerg, 2013).

Each retained period τ_j defines a candidate interval scale. SPS uses a fixed candidate stride $r_{\text{sps}} = 4$ and a minimum candidate length $P_{\min} = 8$ for all datasets. These choices are inspired by a Nyquist–Shannon sampling analysis (Nyquist, 1928; Shannon, 1949) (see Appendix B for details).

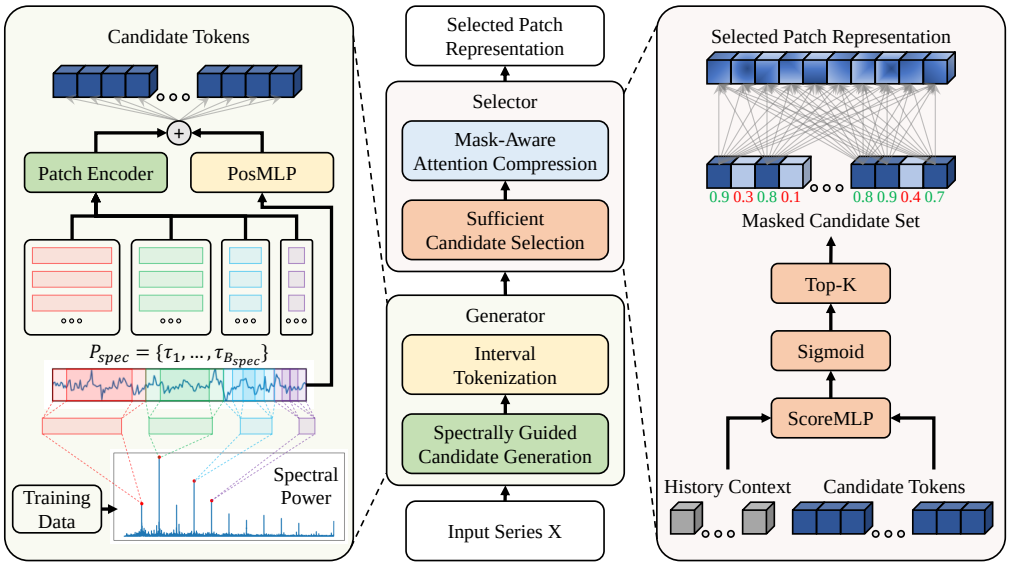


Figure 2: Overview of the proposed Sufficient Patch Selector (SPS) module. The generator constructs spectrally guided multi-scale candidates and tokenizes variable-length intervals. The selector identifies sufficient candidates using local and global context, then compresses them into a compact patch representation for forecasting.

Given the candidate stride r_{sps}, SPS scans the lookback window at regularly spaced start positions and extracts all valid intervals of length τ_j. The resulting candidate set is computed as

$$\mathcal{C}_{\text{spec}} = \mathcal{G}(\mathbf{x}, \mathcal{P}_{\text{spec}}) = \left\{ \mathbf{x}_{t:t+\tau_j-1} \;\middle|\; \tau_j \in \mathcal{P}_{\text{spec}},\ t+\tau_j-1 \le L \right\}, \tag{7}$$

where t ranges over regularly spaced candidate start indices $\{1, 1+r_{\text{sps}}, 1+2r_{\text{sps}}, \ldots\}$. After applying the candidate patch budget, the generator produces $N = |\mathcal{C}_{\text{spec}}|$ candidate patches, where $N \ge K$ and K is the final selected patch budget. This generates candidates at multiple data-aware temporal scales instead of restricting the model to a single manually chosen patch length.

3.3 Interval Tokenization

Since the candidates in $\mathcal{C}_{\text{spec}}$ have different lengths τ_j, SPS maps them into a shared D-dimensional token space with a length-aware patch encoder. For a candidate interval $\mathbf{x}_{t:t+\tau-1}$, SPS applies a patch encoder and adds an interval-position embedding:

$$\mathbf{c}_{t,\tau} = \text{PatchEnc}(W_\tau \mathbf{x}_{t:t+\tau-1}) + \text{PosMLP}(\mathbf{m}_{t,\tau}), \qquad W_\tau \in \mathbb{R}^{D\times\tau}. \tag{8}$$

where W_τ maps the variable-length intervals into $\mathbb{R}^D$, PatchEnc encodes the projected features in the shared token space, and PosMLP embeds the normalized interval metadata $\mathbf{m}_{t,\tau} = \left[\frac{t-1}{L-1}, \frac{t+\tau-2}{L-1}, \frac{\tau}{L}\right]$. The three entries of $\mathbf{m}_{t,\tau}$ represent the normalized start position, end position, and interval length, respectively. Collecting all candidate tokens yields the candidate token matrix

$$\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_N]^\top \in \mathbb{R}^{N\times D}, \tag{9}$$

where N is the number of generated candidates after applying the candidate budget.

3.4 Sufficient Patch Selector

In the selection stage, SPS scores each candidate using both its local token and a global encoding of the full lookback sequence. The global context vector $\mathbf{g} \in \mathbb{R}^D$ is computed as

$$\mathbf{g} = f_{\text{proj}}(\mathbf{x}) + f_{\text{norm}}(\text{LayerNorm}(\mathbf{x})), \tag{10}$$

where f_{proj} is a linear projection to the token dimension D, and f_{norm} is an MLP applied to the normalized history window.

Given a candidate token $\mathbf{c}_i$ and the global context $\mathbf{g}$, SPS assigns a scalar score s_i to each candidate:

$$s_i = \text{Scorer}([\mathbf{c}_i; \mathbf{g}]), \qquad i = 1, \ldots, N, \tag{11}$$

where Scorer is an MLP scorer and $[\cdot;\cdot]$ denotes concatenation.

Let K be the target number of retained candidates and $\rho = K/N$ denote the target keep ratio. SPS converts the scores into soft retention weights α_i using a keep-ratio bias:

$$\alpha_i = \sigma\left(s_i + \text{logit}(\rho)\right), \tag{12}$$

where σ is the sigmoid function.

At inference, SPS retains the top-K candidates according to these weights:

$$\mathcal{I}_K = \text{TopK}(\{\alpha_1, \ldots, \alpha_N\}, K). \tag{13}$$

During training, SPS uses a differentiable soft mask and an alignment loss $\mathcal{L}_{\text{align}}$ to approximate the hard top-K decision. The keep ratio is annealed to K/N, and the retained candidates are then aggregated by mask-aware attention into a compact selected representation

$$\mathbf{Z} \in \mathbb{R}^{K \times D}. \tag{14}$$

Details of the soft masking, alignment loss, annealing strategy, and mask-aware attention are provided in Appendix C.

3.5 Posterior Information Preservation

SPS is motivated by the principle that the selected patches should preserve the predictive posterior induced by the full history. This is a predictive notion of sufficiency rather than classical parametric sufficiency (Koopman, 1936). Instead of preserving all information about a data-generating distribution, SPS aims to retain the information in X that is relevant for predicting Y.

Theorem 1 (Posterior information preservation)**.** *Let X denote the full history, $Z = \mathcal{S}(X)$ the selected patch representation, and Y the forecasting target. Since Z is produced from X, we have the Markov structure $Y \to X \to Z$. Under log-loss, the excess Bayes risk incurred by forecasting from Z instead of X is*

$$R^*(Z) - R^*(X) = \mathbb{E}\left[D_{\text{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right)\right] = I(Y; X \mid Z), \tag{15}$$

where $R^*(W) = \inf_q \mathbb{E}[-\log q(Y \mid W)]$ is the Bayes risk under log-loss.

The proof is provided in Appendix D.

Corollary 1 (Predictive sufficiency under a patch budget)**.** *Since Z is a function of X, we have $I(Y; X) = I(Y; Z) + I(Y; X \mid Z)$. Thus, because $I(Y; X)$ is fixed with respect to the selector $\mathcal{S}$, minimizing the excess Bayes risk or posterior KL under a selected patch budget K is equivalent to maximizing the predictive information retained in Z:*

$$\min_{\mathcal{S}:|\mathcal{S}(X)|\le K} \mathbb{E}\left[D_{\text{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right)\right] \Longleftrightarrow \max_{\mathcal{S}:|\mathcal{S}(X)|\le K} I(Y; Z). \tag{16}$$

The ideal sufficient case is reached when $P(Y \mid X) = P(Y \mid Z)$, in which case the selected patch representation preserves the predictive posterior of the full history.

3.6 Training Objective

Since the true posterior $P(Y \mid X)$ is generally inaccessible, SPS uses a full-history teacher approximation $q_T(Y \mid X)$ (see Appendix E). The student forecasting model conditioned on the selected patches is denoted by $q_\theta(Y \mid Z)$. We train SPS with an approximate posterior-preservation objective

$$\mathcal{L}_{\text{KL}} = \mathbb{E}\left[D_{\text{KL}}\left(q_T(Y \mid X) \,\|\, q_\theta(Y \mid Z)\right)\right]. \tag{17}$$

In implementation, both the teacher and student predictive distributions are represented as diagonal Gaussians, so the KL term can be computed in closed form.

The full training loss is

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{align}}, \tag{18}$$

where $\mathcal{L}_{\text{pred}}$ is the supervised forecasting loss, $\mathcal{L}_{\text{KL}}$ encourages the selected-patch predictor to match the full-history posterior, and $\mathcal{L}_{\text{align}}$ stabilizes top-K selection. We use unit weights for all three terms, avoiding additional loss-balancing hyperparameters.

3.7 Information Preservation Sufficiency

Using the posterior-preservation principle, we introduce Information Preservation Sufficiency (IPS) as an evaluation metric for patching strategies. For a method m with patched representation $Z_i^{(m)}$ on sample i, IPS is defined as

$$\text{IPS}(m) = \frac{1}{n} \sum_{i=1}^{n} D_{\text{KL}} \left(q_T(Y \mid X_i) \,\|\, q_m(Y \mid Z_i^{(m)}) \right). \tag{19}$$

Here, $q_T(Y \mid X_i)$ is the full-history teacher predictive distribution, and $q_m(Y \mid Z_i^{(m)})$ is the predictive distribution induced by method m from its patched representation. To keep IPS comparable across methods, all methods are evaluated against the same shared teacher for each dataset and horizon. The shared teacher's architecture and training are detailed in Appendix E. Lower IPS indicates that the patched representation induces a predictive distribution closer to the full-history reference. Unlike Mean Squared Error (MSE) and Mean Absolute Error (MAE), IPS directly evaluates how well a patching strategy preserves predictive information relative to the full-history posterior approximation.

3.8 SPSNet Architecture

For benchmark comparison, we instantiate SPS in a lightweight MLP-based architecture called SPSNet. SPSNet first applies exponential moving average (EMA) decomposition. The trend branch applies RevIN normalization (Kim et al., 2021) and uses a separate MLP forecasting head. The seasonal branch applies SPS channel-wise, retains the K selected patch tokens, and passes them through a light token-attention layer to another MLP head. The final forecast is the sum of the trend and seasonal forecasts, refined by a lightweight channel mixer. Further details and motivation for the SPSNet design are provided in Appendix F.

SPS is a plug-in module and is not tied to SPSNet. In our experiments, it can replace the segmentation stage of existing patch-based backbones to evaluate information preservation across architectures.

4 Experiments

4.1 Experimental Details

Datasets. To conduct extensive and fair comparisons for different models, we utilize eight real-world multivariate time series datasets, which are standard forecasting benchmarks: Electricity Transformer Temperature (ETTh1, ETTh2, ETTm1, ETTm2) (Zhou et al., 2021), Weather, Electricity, Traffic (Wu et al., 2021), and Solar-energy (Lai et al., 2018) (see Appendix G for details).

Baselines. We carefully choose seven recent state-of-the-art models as our benchmark, including PatchTST (Nie et al., 2023), TimeMixer (Wang et al., 2024a), iTransformer (Liu et al., 2024), PatchMLP (Tang and Zhang, 2025), TimeKAN (Huang et al., 2025), SRSNet (Wu et al., 2025), and TimeMosaic (Ding et al., 2026).

Implementation Details. To keep consistency with previous works, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics to evaluate the performance. For all datasets, we utilize four forecasting horizons $H = \{96, 192, 336, 720\}$. To ensure a fair comparison, we follow unified settings with a fixed lookback window $L = 96$ for all datasets (see Appendix H for details).

4.2 SPSNet Results

Table 1 reports the averaged forecasting results. Compared with recent state-of-the-art models from different architectural families, SPSNet obtains the most first-place rankings across datasets and metrics. Full horizon-wise results are provided in Appendix M, both for the unified lookback $L = 96$ (Table 9) and for a longer lookback $L = 336$ (Table 10), where SPSNet again ranks first most often. Compared with patch-based baselines, SPSNet achieves better accuracy while retaining fewer patches, indicating that information-preserving patch selection can improve forecasting under a smaller patch budget.

Table 1: Averaged multivariate forecasting results with unified lookback $L = 96$. **Red** denotes the best, while blue stands for the second best. Full results are available in Table 9 of Appendix M.

Models	SPSNet (ours)		TimeMosaic (2026)		SRSNet (2025)		TimeKAN (2025)		PatchMLP (2025)		iTransformer (2024)		TimeMixer (2024)		PatchTST (2023)	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.438	**0.431**	0.451	0.445	0.450	0.435	**0.423**	0.432	0.469	0.452	0.454	0.448	0.456	0.444	0.441	0.435
ETTh2	**0.371**	**0.396**	0.385	0.405	0.372	0.399	0.387	0.406	0.404	0.423	0.383	0.407	0.381	0.406	0.383	0.406
ETTm1	**0.374**	**0.385**	0.391	0.403	0.388	0.401	0.381	0.397	0.398	0.405	0.407	0.410	0.385	0.398	0.378	0.395
ETTm2	0.277	**0.319**	0.280	0.324	0.279	0.325	0.284	0.329	0.294	0.333	0.288	0.332	**0.275**	0.322	0.285	0.329
Weather	0.252	**0.271**	0.257	0.279	0.261	0.282	**0.243**	0.273	0.252	0.279	0.258	0.278	0.247	0.276	0.248	0.275
Solar	**0.195**	**0.221**	0.246	0.283	0.260	0.284	0.276	0.316	0.258	0.288	0.233	0.262	0.236	0.292	0.224	0.261
Electricity	0.181	**0.269**	0.195	0.286	0.193	0.279	0.199	0.286	0.193	0.287	**0.178**	0.270	0.182	0.272	0.183	0.273
Traffic	0.466	**0.279**	0.488	0.325	0.497	0.315	0.580	0.374	0.493	0.333	**0.428**	0.282	0.470	0.291	0.515	0.291

4.3 SPS Results

Beyond SPSNet, we further evaluate SPS as a plug-in module for existing patch-based forecasting backbones. The IPS comparison results are reported in Table 11 of Appendix M. SPS consistently improves information preservation across patch-based models, indicating that the selected patches induce predictive distributions closer to the full-lookback reference.

Detailed results for integrating SPS into different backbones are reported in Table 6 of Appendix I. These results show that SPS is not tied to the SPSNet architecture and can improve existing patch-based models by replacing fixed patching with information-preserving selection. To further analyze SPS, we provide additional ablation, sensitivity, and efficiency studies in the appendix. Specifically, Appendix J evaluates the contribution of key SPS components, Appendix K studies the sensitivity of SPS to its main hyperparameters, and Appendix L compares SPSNet with other baselines in terms of efficiency.

5 Conclusion

In this paper, we revisited patching for long-term time series forecasting as a two-stage process of candidate generation and candidate selection. Building on this formulation, we proposed the Sufficient Patch Selector (SPS), a plug-in module that constructs spectrally guided multi-scale candidates and selects a compact subset that preserves predictive information from the full lookback sequence. Through the posterior-preservation objective and the Information Preservation Sufficiency (IPS) metric, SPS provides a way to evaluate patching strategies not only by forecasting error, but also by their ability to retain predictive information under compression.

Our results show that patching should not be treated as a fixed preprocessing choice. In real-world scenarios, time series are often non-stationary, and predictive information is not uniformly distributed across the lookback window. Adaptive patch selection can therefore reduce redundant or weakly informative tokens while preserving the information needed for accurate forecasting. More broadly, our findings suggest that information-preserving patch selection is a principled alternative to fixed patch tokenization and a promising direction for building more efficient forecasting models.

AI USE STATEMENT

In this work, we used large language models (LLMs) to assist with writing, including polishing the text for grammar and clarity. We have not used generative AI tools to generate research ideas. All AI-assisted text was reviewed before being included in the paper. The problem formulation, the proposed method, and the theoretical results were developed by the authors.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of our results. Theorem 1 and Corollary 1 are proved in Appendix D, which also states the assumption under which they hold. The components of SPS are specified in the appendix: candidate generation and its hyperparameters in Appendix B, soft masking, keep-ratio annealing, the alignment loss, and mask-aware attention in Appendix C, and the full-history teacher in Appendix E. The SPSNet architecture is described in Appendix F, and the integration of SPS into existing patch-based backbones in Appendix I. All eight datasets are publicly available; their sources, statistics, and train/validation/test splits are given in Appendix G and Table 4. Appendix H reports the hardware, optimizer, learning rates, batch sizes, lookback length, candidate and selection budgets, and the protocol used to re-run each baseline with its official implementation, together with results over five random seeds (Table 5). Full results for all forecasting horizons under both lookback settings are provided in Appendix M, and the ablation, hyperparameter-sensitivity, and efficiency protocols are described in Appendices J, K, and L. The source code, including the scripts and configurations used for all experiments, is provided in the supplementary material.

REFERENCES

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyxQzBceg.

George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970.

Peng Chen, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=lJkOCMP2aW.

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1994–2001. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/277. URL https://doi.org/10.24963/ijcai.2022/277. Main Track.

Kuiye Ding, Fanda Fan, Chunyi Hou, Zheya Wang, Lei Wang, Zhengxin Yang, and Jianfeng Zhan. Timemosaic: Temporal heterogeneity guided time series forecasting via adaptive granularity patch and segment-wise decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 20790–20798, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Georg Goerg. Forecastable component analysis. In *International conference on machine learning*, pages 64–72. PMLR, 2013.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Qihe Huang, Lei Shen, Ruixin Zhang, Jiahuan Cheng, Shouhong Ding, Zhengyang Zhou, and Yang Wang. Hdmixer: Hierarchical dependency with extendable patch for multivariate time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 12608–12616, 2024.

Songtao Huang, Zhen Zhao, Can Li, and Lei Bai. TimeKAN: KAN-based frequency decomposition learning architecture for long-term time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wTLc79YNbh.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

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.

Bernard Osgood Koopman. On distributions admitting a sufficient statistic. *Transactions of the American Mathematical society*, 39(3):399–409, 1936.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

Zhe Li, Xiangfei Qiu, Peng Chen, Yihang Wang, Hanyin Cheng, Yang Shu, Jilin Hu, Chenjuan Guo, Aoying Zhou, Christian S Jensen, et al. Tsfm-bench: A comprehensive and unified benchmark of foundation models for time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5595–5606, 2025.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

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. URL https://openreview.net/forum?id=JePfAI8fah.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Stephane G Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

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

Yiming Niu, Jinliang Deng, and Yongxin Tong. Phaseformer: From patches to phases for efficient and effective time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Lk9SqMQzhX.

Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *Advances in neural information processing systems*, 34:12786–12797, 2021.

CE Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

Susanne Still. Information bottleneck approach to predictive inference. *Entropy*, 16(2):968–989, 2014. ISSN 1099-4300. doi: 10.3390/e16020968. URL https://www.mdpi.com/1099-4300/16/2/968.

Artyom Stitsyuk and Jaesik Choi. xpatch: Dual-stream time series forecasting with exponential seasonal-trend decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20601–20609, 2025.

Peiwang Tang and Weitai Zhang. Unlocking the power of patch: Patch-based mlp for long-term time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 12640–12648, 2025.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*, 2023.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=7oLshfEIC2.

Xue Wang, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. CARD: Channel aligned robust blend transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=MJksrOhurE.

Yihang Wang, Yuying Qiu, Peng Chen, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. LightGTS: A lightweight general time series forecasting model. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Z5FJsp1U3Z.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

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

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Xingjian Wu, Xiangfei Qiu, Hanyin Cheng, Zhengyu Li, Jilin Hu, Chenjuan Guo, and Bin Yang. Enhancing time series forecasting through selective representation spaces: A patch perspective. In *NeurIPS*, 2025.

Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, 2009.

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, pages 11121–11128, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022.

A Additional Related Work

Time series forecasting (TSF) is a broad research area with many successful modeling paradigms. Due to space constraints, the main paper focuses its related work discussion on patch-based approaches, which are most directly relevant to Sufficient Patch Selector (SPS). Here, we briefly acknowledge several influential non-patching methods that have shaped the field.

Informer (Zhou et al., 2021) introduced an efficient Transformer architecture for long-sequence forecasting. Autoformer (Wu et al., 2021) incorporated decomposition and autocorrelation mechanisms, while FEDformer (Zhou et al., 2022) further explored frequency-domain modeling. ETSformer (Woo et al., 2022) integrated exponential smoothing principles into Transformer-based forecasting. Non-stationary Transformer (Liu et al., 2022) addressed distribution shift and non-stationarity in time series. MICN (Wang et al., 2023) modeled multi-scale temporal patterns using convolutional structures. TimesNet (Wu et al., 2023) represented temporal variations through multiple periods, and DLinear (Zeng et al., 2023) showed that simple linear models with decomposition can be highly competitive. iTransformer (Liu et al., 2024) inverted the standard Transformer formulation by treating variables as tokens, enabling attention to model inter-variate dependencies more directly.

B Candidate Generation Hyperparameters

The generator is controlled by four main hyperparameters: the maximum number of retained spectral periods B_{spec}, the candidate patch budget N, the candidate stride r_{sps}, and the minimum candidate interval length $P_{\min}$.

After computing the spectral power on the training split, SPS converts the strongest valid spectral peaks into candidate period lengths. The resulting periods are sorted in descending order of spectral power, so that periods corresponding to stronger recurring temporal scales are considered first. In all experiments, we keep at most $B_{\text{spec}} = 10$ spectral periods:

$$|\mathcal{P}_{\text{spec}}| \leq B_{\text{spec}}. \tag{20}$$

We set $B_{\text{spec}} = 10$ because, together with the fixed stride used in SPS, it is sufficient to generate the required candidate budget for all datasets considered in our experiments.

The candidate patch budget is fixed to $N = 80$. Candidate generation proceeds in descending spectral-power order. For each retained period τ_j, SPS scans the lookback window with stride r_{sps} and extracts valid intervals of length τ_j. The generator first creates candidates from the strongest period, then continues with the next strongest periods until $N = 80$ candidates have been collected. This procedure ensures that the candidate set is filled primarily with intervals from the most dominant temporal scales, while keeping the number of candidates fixed across datasets.

To choose a fixed candidate stride, we estimate the shortest meaningful period from the training split of each dataset. This follows the same spectral analysis used for constructing $\mathcal{P}_{\text{spec}}$, but here we focus on the shortest dominant temporal scale that should still be resolved by candidate generation. Following the sampling intuition underlying the Nyquist–Shannon principle (Nyquist, 1928; Shannon, 1949) that at least two observations are needed to resolve a cycle, we divide this shortest meaningful period by two to obtain a dataset-specific suggested stride.

Table 2 shows that the smallest suggested stride across datasets is 4. For consistency, we therefore use a fixed candidate stride $r_{\text{sps}} = 4$ in all experiments. This choice satisfies the suggested stride for every dataset and provides denser coverage for datasets with longer shortest periods. It also motivates setting $P_{\min} = 8$, so that the smallest allowed candidate interval contains at least two stride steps. This yields at least 50% overlap for the shortest allowed intervals and denser coverage for longer intervals, improving coverage of phase-shifted patterns while keeping the candidate set manageable.

Table 3 reports the periods that this procedure recovers on each benchmark. The number of recovered periods differs across datasets because generation proceeds in descending spectral-power order and stops once the candidate budget $N = 80$ is filled. A short period admits many valid start positions and therefore contributes many candidates at a fixed stride, so datasets whose dominant scales are short reach the budget after four periods (Electricity, Traffic), whereas datasets whose dominant

Table 2: Dataset-specific stride suggestions from the shortest meaningful periods estimated on the training split.

Dataset	Shortest meaningful period	Suggested stride
ETTh1	8	4
ETTh2	24	12
ETTm1	32	16
ETTm2	96	48
Electricity	24	12
Solar	72	36
Traffic	8	4
Weather	72	36

scales are long contribute few candidates each and draw on as many as ten (ETTh2). The listed periods are thus the top-ranked scales required to fill $N = 80$ rather than a fixed-size set per dataset.

Table 3: Periods recovered by the spectral generator on each benchmark. Periods are listed in descending order of spectral power on the training split.

Dataset	Sampling	Recovered periods $\mathcal{P}_{\text{spec}}$	Most selected (share)
ETTh1	1 h	24, 12, 8, 93, 90, 50, 29	12 (31%), 8 (30%), 24 (24%)
ETTh2	1 h	24, 95, 74, 83, 90, 39, 80, 51, 42, 86	24 (25%), 42 (21%), 51 (18%)
ETTm1	15 min	96, 32, 48, 94, 24, 12, 14	12 (28%), 24 (24%), 48 (19%)
ETTm2	15 min	96, 16, 24, 12, 10	12 (30%), 24 (28%), 16 (23%)
Weather	10 min	72, 48, 36, 76, 69, 66, 74, 46, 44	36 (22%), 46 (16%), 48 (15%)
Solar	10 min	72, 36, 24, 48, 92, 81, 69, 63, 29	24 (38%), 36 (13%), 48 (11%)
Electricity	1 h	24, 12, 8, 28	12 (43%), 24 (22%), 8 (21%)
Traffic	1 h	24, 12, 8, 28	8 (28%), 12 (25%), 24 (24%)

In addition, this analysis shows that some of the spectrally selected scales correspond to interpretable physical periods:

- Daily period (24 hours): 24 on hourly data (ETTh1, ETTh2, Electricity, Traffic), 96 on 15-min data (ETTm1, ETTm2)
- Half-daily period (12 hours): 12 on hourly data (ETTh1, Electricity, Traffic), 48 on 15-min data (ETTm1), 72 on 10-min data (Weather, Solar)
- Third-daily period (8 hours): 8 on hourly data (ETTh1, Electricity, Traffic), 32 on 15-min data (ETTm1), 48 on 10-min data (Weather, Solar)
- Quarter-daily period (6 hours): 24 on 15-min data (ETTm1, ETTm2), 36 on 10-min data (Weather, Solar)

C Soft Masking, Annealing, and Mask-Aware Attention

We provide the implementation details of the differentiable masking, keep-ratio annealing, and mask-aware attention used in SPS. Let $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_N]^\top \in \mathbb{R}^{N \times D}$ denote the candidate token matrix, where N is the number of generated candidates. Let K denote the selected patch budget, with $K \leq N$. The selector assigns a scalar score s_i to each candidate token.

Keep-ratio annealing. For stable training, SPS anneals the effective keep ratio from a dense initial value to the target value K/N during the first E_{anneal} training epochs. At training step t, we define an effective keep ratio ρ_t and the corresponding effective selection budget

$$K_t = \lceil \rho_t N \rceil, \qquad \rho_t \in [K/N, 1 - \delta], \tag{21}$$

where N is the number of candidate patches and K is the final selected patch budget. The value of ρ_t is gradually decreased to the target keep ratio K/N. During training, the soft mask threshold

and the hard alignment target are computed using K_t. At inference, SPS uses the final selected patch budget K. This annealing strategy allows the mask-aware attention module to initially aggregate information from more candidate patches while keeping the output compressed into K slots. As training progresses, the selection constraint becomes stronger and approaches the final top-K selection behavior.

Soft top-K_t mask. The hard TopK operation is discrete and non-differentiable. To enable end-to-end training, SPS uses a differentiable soft mask that approximates the hard top-K_t decision during training. Let $s_{(K_t)}$ denote the K_t-th largest candidate score, which acts as the current top-K_t threshold. We first compute a soft indicator

$$q_i = \sigma\left(\frac{s_i - s_{(K_t)}}{t_{\text{topk}}}\right), \qquad i = 1, \dots, N, \tag{22}$$

where σ is the sigmoid function and t_{topk} is a temperature parameter. Candidates with scores above the current top-K_t threshold receive values close to 1, while lower-scoring candidates receive values close to 0.

The soft indicators are then normalized to approximately match the current effective budget K_t:

$$m_i = \text{clip}\left(\frac{K_t q_i}{\sum_{j=1}^{N} q_j + \epsilon}, 0, 1\right), \qquad i = 1, \dots, N, \tag{23}$$

where ϵ is a small constant for numerical stability. The resulting mask $\mathbf{m} = [m_1, \dots, m_N]^\top \in [0,1]^N$ softly suppresses low-scoring candidates while approximately preserving the current selection budget.

At inference, SPS replaces the soft mask with the hard top-K binary mask

$$b_i = \mathbb{I}\left[i \in \text{TopK}(\{s_1, \dots, s_N\}, K)\right], \qquad i = 1, \dots, N. \tag{24}$$

Alignment loss. In addition to the soft training mask, SPS uses an alignment penalty to encourage the soft retention weights to match the hard selection decision. At training step t, the hard target is computed using the current effective budget K_t:

$$b_i^{(t)} = \mathbb{I}\left[i \in \text{TopK}(\{s_1, \dots, s_N\}, K_t)\right], \qquad i = 1, \dots, N. \tag{25}$$

Let $\boldsymbol{\alpha}^{(t)} \in [0,1]^N$ denote the soft retention weights produced by the selector under the current keep ratio. The alignment loss is

$$\mathcal{L}_{\text{align}} = \text{BCE}\left(\boldsymbol{\alpha}^{(t)}, \text{stopgrad}(\mathbf{b}^{(t)})\right), \tag{26}$$

where BCE denotes binary cross-entropy. The stop-gradient operator prevents the hard top-K_t targets from receiving gradients, so the loss only updates the soft selector.

Mask-aware attention pooling. After masking, SPS compresses the candidate tokens into a fixed-size selected representation. Let $\mathbf{h}_i \in \mathbb{R}^D$ denote the hidden representation of candidate i after token projection. SPS uses K learnable query vectors $\mathbf{q}_1, \dots, \mathbf{q}_K$ to aggregate the masked candidates into K output slots.

For each output slot ℓ, SPS first computes preliminary mask-aware attention weights:

$$\widetilde{A}_{\ell i} = \text{softmax}_i\left(\frac{\mathbf{q}_\ell^\top \mathbf{h}_i}{\sqrt{D}} + \log(m_i + \epsilon)\right), \qquad \ell = 1, \dots, K, \quad i = 1, \dots, N. \tag{27}$$

The term $\log(m_i + \epsilon)$ down-weights candidates with small mask values before the softmax. SPS then applies explicit masking and renormalization:

$$A_{\ell i} = \frac{m_i \widetilde{A}_{\ell i}}{\sum_{j=1}^{N} m_j \widetilde{A}_{\ell j} + \epsilon}, \qquad \ell = 1, \dots, K, \quad i = 1, \dots, N. \tag{28}$$

Combining the log-mask term in Equation 27 with the explicit masking in Equation 28 is intentional. The log-mask term down-weights low-mask candidates before the softmax, while the explicit masking and renormalization further concentrate attention on retained candidates. Thus, during training, soft masks suppress low-score candidates more strongly while still allowing gradients to flow through the selector. At inference, where $m_i \in \{0, 1\}$, the mechanism reduces to standard hard masked attention.

The output representation for slot ℓ is

$$\mathbf{z}_\ell = \sum_{i=1}^{N} A_{\ell i} \mathbf{h}_i, \qquad \ell = 1, \ldots, K. \tag{29}$$

Collecting all output slots gives the selected patch representation

$$\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_K]^\top \in \mathbb{R}^{K \times D}. \tag{30}$$

Temporal information is provided through interval-position embeddings, while mask-aware attention aggregates the selected candidates into a fixed-size representation and suppresses candidates not selected by the soft mask.

D Proof of Posterior Information Preservation

We provide the proofs of Theorem 1 and Corollary 1. Let $X = (X_1, \ldots, X_B)$ denote the full history or the full set of candidate patches, let $Z = \mathcal{S}(X)$ denote the selected patch representation, and let Y denote the forecasting target. We assume that Z is produced deterministically from X, so that $Y \to X \to Z$ forms a Markov chain. Equivalently,

$$P(Y \mid X, Z) = P(Y \mid X). \tag{31}$$

Proof of Theorem 1. For any information variable W, the log-loss risk of a predictor $q(Y \mid W)$ is

$$R(q; W) = \mathbb{E}\left[-\log q(Y \mid W)\right]. \tag{32}$$

Under log-loss, the Bayes-optimal predictor is the true conditional distribution,

$$q^*(Y \mid W) = P(Y \mid W). \tag{33}$$

Therefore, the Bayes risk is

$$R^*(W) = \mathbb{E}\left[-\log P(Y \mid W)\right] = H(Y \mid W). \tag{34}$$

Applying this identity to $W = Z$ and $W = X$ gives

$$R^*(Z) - R^*(X) = H(Y \mid Z) - H(Y \mid X). \tag{35}$$

We next show that this excess risk is equal to the expected posterior KL divergence. By definition,

$$\begin{aligned} &\mathbb{E}\left[D_{\mathrm{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right)\right] \\ &= \mathbb{E}\left[\log \frac{P(Y \mid X)}{P(Y \mid Z)}\right] \\ &= \mathbb{E}\left[\log P(Y \mid X)\right] - \mathbb{E}\left[\log P(Y \mid Z)\right]. \end{aligned} \tag{36}$$

Using

$$\mathbb{E}\left[\log P(Y \mid X)\right] = -H(Y \mid X), \qquad \mathbb{E}\left[\log P(Y \mid Z)\right] = -H(Y \mid Z), \tag{37}$$

we obtain

$$\mathbb{E}\left[D_{\mathrm{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right)\right] = H(Y \mid Z) - H(Y \mid X). \tag{38}$$

Thus,

$$R^*(Z) - R^*(X) = \mathbb{E}\left[D_{\mathrm{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right)\right]. \tag{39}$$

It remains to show that the same quantity equals $I(Y; X \mid Z)$. By the definition of conditional mutual information,

$$I(Y; X \mid Z) = H(Y \mid Z) - H(Y \mid X, Z). \tag{40}$$

Since $Z = \mathcal{S}(X)$ is a deterministic function of X, conditioning on both X and Z is equivalent to conditioning on X alone for predicting Y:

$$H(Y \mid X, Z) = H(Y \mid X). \tag{41}$$

Therefore,

$$I(Y; X \mid Z) = H(Y \mid Z) - H(Y \mid X). \tag{42}$$

Combining the identities above yields

$$R^*(Z) - R^*(X) = \mathbb{E}\left[D_{\mathrm{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right)\right] = I(Y; X \mid Z). \tag{43}$$

This completes the proof. □

Proof of Corollary 1. Since $Z = \mathcal{S}(X)$ is a deterministic function of X, we have

$$I(Y; Z \mid X) = 0. \tag{44}$$

Using the chain rule for mutual information,

$$I(Y; X, Z) = I(Y; Z) + I(Y; X \mid Z) \tag{45}$$

and also

$$I(Y; X, Z) = I(Y; X) + I(Y; Z \mid X). \tag{46}$$

Because $I(Y; Z \mid X) = 0$, it follows that

$$I(Y; X) = I(Y; Z) + I(Y; X \mid Z). \tag{47}$$

The quantity $I(Y; X)$ is fixed by the data distribution and does not depend on the selector $\mathcal{S}$. Therefore, minimizing $I(Y; X \mid Z)$ over selectors is equivalent to maximizing $I(Y; Z)$:

$$\min_{\mathcal{S}:|\mathcal{S}(X)|\le K} I(Y; X \mid Z) \iff \max_{\mathcal{S}:|\mathcal{S}(X)|\le K} I(Y; Z). \tag{48}$$

By Theorem 1, $I(Y; X \mid Z)$ is equal to the expected posterior KL divergence and to the excess Bayes risk under log-loss. Thus,

$$\min_{\mathcal{S}:|\mathcal{S}(X)|\le K} \mathbb{E}\left[D_{\mathrm{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid \mathcal{S}(X))\right)\right] \iff \max_{\mathcal{S}:|\mathcal{S}(X)|\le K} I(Y; \mathcal{S}(X)). \tag{49}$$

This completes the proof. □

The ideal predictively sufficient case is reached when

$$P(Y \mid X) = P(Y \mid Z), \tag{50}$$

which implies

$$D_{\mathrm{KL}}\left(P(Y \mid X) \,\|\, P(Y \mid Z)\right) = 0, \qquad I(Y; X \mid Z) = 0. \tag{51}$$

In this case, the selected patch representation Z preserves all predictive information about Y contained in the full history X.

E Full-History Teacher Details

The posterior preservation objective requires a reference predictive distribution conditioned on the full lookback sequence. Since the true posterior $P(Y \mid X)$ is unavailable, we approximate it with a separately trained full history teacher model $q_T(Y \mid X)$. To support this approximation, SPS-based models are equipped with Gaussian output heads: in addition to the point forecast, the model predicts a log-variance, so the output can be interpreted as a diagonal Gaussian predictive distribution. For SPSNet this is native, as the seasonal and the trend branch each predict a log-variance and their variances are summed. For the plug-in backbones, the Gaussian head reads *detached* forecasts, so the probabilistic branch cannot alter the point path, and the reported MSE of a baseline is unaffected by enabling IPS. The mean is used for standard forecasting evaluation, while the predicted log-variance is used for teacher posterior matching and IPS computation.

The teacher is a full-history reference shared by every model we compare. Concretely, we use an iTransformer trained on the full lookback sequence, without any patch selection, and the *same*

teacher is reused by every student on a given dataset and horizon. This is a deliberate design choice: IPS values are only comparable across architectures if they are measured against one common reference, and a per-student teacher would make each model's score depend on the quality of its own reference. Formally, the teacher defines

$$q_T(Y \mid X) = \mathcal{N}\left(\boldsymbol{\mu}_T(X), \mathrm{diag}(\boldsymbol{\sigma}_T^2(X))\right), \tag{52}$$

where $\boldsymbol{\mu}_T(X)$ is the teacher point forecast and $\boldsymbol{\sigma}_T^2(X)$ is the teacher-predicted diagonal variance.

The selected-patch student defines

$$q_\theta(Y \mid Z) = \mathcal{N}\left(\boldsymbol{\mu}_\theta(Z), \mathrm{diag}\left(\boldsymbol{\sigma}_\theta^2(Z)\right)\right), \tag{53}$$

where Z denotes the selected patch representation, $\boldsymbol{\mu}_\theta(Z)$ is the student point forecast, and $\boldsymbol{\sigma}_\theta^2(Z)$ is the student-predicted diagonal variance. The posterior-preservation loss is the closed-form KL divergence between these two diagonal Gaussians, and is added to the student objective with weight $\lambda_{\mathrm{KL}} = 1$.

The teacher is trained once per dataset and horizon, before the students that use it. Its objective combines a supervised point forecasting loss with a Gaussian negative log-likelihood term:

$$\mathcal{L}_T = \mathcal{L}_{\mathrm{point}}\left(\boldsymbol{\mu}_T(X), Y\right) + \mathcal{L}_{\mathrm{NLL}}\left(\boldsymbol{\mu}_T(X), \boldsymbol{\sigma}_T^2(X), Y\right), \tag{54}$$

where

$$\mathcal{L}_{\mathrm{NLL}} = \frac{1}{2}\mathbb{E}\left[\log \boldsymbol{\sigma}_T^2(X) + \frac{(Y - \boldsymbol{\mu}_T(X))^2}{\boldsymbol{\sigma}_T^2(X)}\right]. \tag{55}$$

Since the teacher is shared, its point loss $\mathcal{L}_{\mathrm{point}}$ is its own objective, MSE, as in the official iTransformer. The teacher is optimized with AdamW, and early stopping is based on the validation point forecasting loss, so that the teacher serves as a strong full-history forecasting reference rather than being optimized only for uncertainty calibration.

After training, the best teacher checkpoint is loaded and frozen. Gradients are never propagated through the teacher: for each minibatch, it produces $\boldsymbol{\mu}_T(X)$ and $\boldsymbol{\sigma}_T^2(X)$ under `torch.no_grad()`, and the student distribution $q_\theta(Y \mid Z)$ is trained to match this full-history reference through the closed-form KL divergence between diagonal Gaussians. The same frozen teacher is used for IPS evaluation, which is what makes IPS comparable across all models reported on a given dataset and horizon. At test time the forecasting model does not require the teacher, so it is used only as a training and evaluation reference for posterior information preservation.

F SPSNet Design

This section provides the architectural details and motivation for SPSNet, the lightweight MLP-based forecasting model used to evaluate SPS against benchmark models. Given an input sequence $\mathbf{X} \in \mathbb{R}^{L \times M}$, SPSNet first applies exponential moving average (EMA) decomposition (Stitsyuk and Choi, 2025):

$$\mathbf{X} = \mathbf{X}_{\mathrm{season}} + \mathbf{X}_{\mathrm{trend}}. \tag{56}$$

The two components are then processed by separate forecasting branches.

The trend branch applies RevIN normalization (Kim et al., 2021) to $\mathbf{X}_{\mathrm{trend}}$ and uses an MLP forecasting head to produce $\widehat{\mathbf{Y}}_{\mathrm{trend}}$. The seasonal branch applies SPS independently to each channel of $\mathbf{X}_{\mathrm{season}}$ and retains the K selected interval tokens in temporal order. These tokens interact through a single self-attention layer, which also attends to a context token averaged across channels. They are then flattened and passed to another MLP forecasting head, producing $\widehat{\mathbf{Y}}_{\mathrm{season}}$. We use GELU (Hendrycks and Gimpel, 2016) as the activation function in the MLP heads. The final prediction combines the two forecasts and refines them with a low-rank channel mixer Mix:

$$\widehat{\mathbf{Y}} = \mathrm{Mix}\left(\widehat{\mathbf{Y}}_{\mathrm{season}} + \widehat{\mathbf{Y}}_{\mathrm{trend}}\right). \tag{57}$$

For posterior-preservation training and IPS evaluation, the forecasting heads are augmented to output both a mean and a log-variance, defining a diagonal Gaussian predictive distribution as described in Appendix E. The mean output is used as the point forecast for standard MSE and MAE evaluation.

The use of EMA decomposition is motivated by the local-level state-space model, where the Kalman filter (Kalman, 1960) recursively updates the latent trend as

$$\hat{T}_t = (1 - K_t)\hat{T}_{t-1} + K_t x_t. \tag{58}$$

In steady state, the Kalman gain converges to a constant $K^\star$, giving

$$\hat{T}_t = (1 - K^\star)\hat{T}_{t-1} + K^\star x_t. \tag{59}$$

This has the same form as EMA,

$$\mathrm{EMA}_t = (1 - \alpha)\,\mathrm{EMA}_{t-1} + \alpha x_t, \tag{60}$$

with $\alpha = K^\star$. Thus, EMA provides a natural recursive estimator for slowly varying trend components.

RevIN is applied to the trend branch because trend dynamics are often affected by instance-wise level and scale shifts. By normalizing each trend window and applying the inverse transformation at the output, RevIN allows the trend predictor to operate in a normalized space while preserving the original level and scale information needed for forecasting.

After trend removal, the seasonal component contains more local and recurring temporal patterns. SPS is therefore applied to the seasonal branch, where its multi-scale candidate generation and sufficient patch selection can focus the patch budget on informative recurring intervals. This design separates slowly varying trend dynamics from multi-scale seasonal structure: the trend branch handles non-stationary level and scale changes, while the seasonal branch uses SPS to produce a compact, information-preserving representation.

Finally, SPS is a plug-in module and is not tied to SPSNet. In our experiments, the same selection mechanism can replace the segmentation stage of existing patch-based backbones, allowing us to evaluate whether information-preserving patch selection improves different forecasting architectures.

G Dataset Details

We conduct experiments on eight real-world multivariate time series datasets to evaluate the performance of the proposed model. The dataset summary and statistics are provided in Table 4.

- Electricity Transformer Temperature (ETT)[1]: has four subsets, where ETTh1 and ETTh2 are recorded every hour, while ETTm1 and ETTm2 are recorded every 15 minutes. Data is collected from two different electric transformers.
- Weather[2]: collects 21 meteorological indicators in Germany, such as humidity and air temperature, collected every 10 minutes from the Weather Station of the Max Planck Biogeochemistry Institute in 2020.
- Solar-energy[3]: contains the solar power production records from 137 PV plants in 2006.
- Electricity[4]: describes the hourly electricity consumption data of 321 clients from 2012 to 2014.
- Traffic[5]: records hourly road occupancy rates measured by 862 sensors on San Francisco Bay area freeways from January 2015 to December 2016.

H Implementation Details

All experiments are implemented in PyTorch (Paszke et al., 2019) and conducted on a single NVIDIA RTX PRO 5000 GPU. For the reproduced baselines, we use their official repository implementations and preserve the original architectural configurations whenever possible.

[1] https://github.com/zhouhaoyi/ETDataset
[2] https://www.bgc-jena.mpg.de/wetter
[3] https://www.nrel.gov/grid/solar-power-data.html
[4] https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014
[5] https://pems.dot.ca.gov

Table 4: Summary and statistics of benchmark datasets.

Dataset	Domain	Frequency	Lengths	Dim	Split	Description
ETTh1	Electricity	1 hour	14,400	7	6:2:2	7 indicators of Power transformer 1
ETTh2	Electricity	1 hour	14,400	7	6:2:2	7 indicators of Power transformer 2
ETTm1	Electricity	15 mins	57,600	7	6:2:2	7 indicators of Power transformer 1
ETTm2	Electricity	15 mins	57,600	7	6:2:2	7 indicators of Power transformer 2
Weather	Environment	10 mins	52,696	21	7:1:2	21 meteorological indicators for the whole year 2020
Solar	Energy	10 mins	52,560	137	6:2:2	Solar production records from 137 PV plants
Electricity	Electricity	1 hour	26,304	321	7:1:2	Hourly electricity consumption records of 321 clients
Traffic	Traffic	1 hour	17,544	862	7:1:2	Hourly road occupancy rates measured by 862 sensors

Following the evaluation settings of TFB (Qiu et al., 2024) and TSFM-Bench (Li et al., 2025), we do not use the "Drop Last" trick to ensure a fair comparison. Accordingly, we re-run TimeMixer (Wang et al., 2024a) and TimeKAN (Huang et al., 2025) under the same architectural configurations but without the "Drop Last" setting.

We use a unified lookback length of $L = 96$ across all experiments. To ensure consistency under this setting, we re-run PatchTST (Nie et al., 2023), TimeMixer (Wang et al., 2024a), PatchMLP (Tang and Zhang, 2025), TimeKAN (Huang et al., 2025), SRSNet (Wu et al., 2025), and TimeMosaic (Ding et al., 2026) using their official implementations. For iTransformer (Liu et al., 2024), we report the results from the official paper because its experimental setting matches ours and it does not use the "Drop Last" trick.

We use the AdamW optimizer (Loshchilov and Hutter, 2019). The learning rate is set to 10^{-3} for large datasets, including Solar, Electricity, and Traffic, and to 10^{-4} for the remaining datasets, including ETT and Weather. The batch size is set to 64 by default and is halved only when necessary due to memory constraints. In practice, this adjustment is required only for TimeMosaic on large datasets.

For SPS-based experiments, we set the candidate patch budget to $N = 80$ and the selected patch budget to $K = 10$. This gives SPS a smaller retained patch budget than standard fixed-patching baselines while still allowing selection from a sufficiently rich candidate set. For a fixed-patching model with patch length P and stride r, the number of generated patches is

$$N_{\text{fixed}} = \left\lfloor \frac{L - P}{r} \right\rfloor + 1. \tag{61}$$

Thus, with lookback length $L = 96$, patch length $P = 16$, and stride $r = 8$, fixed-patching methods generate and retain all $N = 11$ patches. In contrast, SPS explicitly separates candidate generation from patch retention. For SRSNet, we follow the official candidate construction setting with patch length $P = 16$ and stride $r = 1$, which produces $N = 81$ candidates and selects $K = 11$ patches. SPS uses a comparable candidate budget of $N = 80$ candidates but retains only $K = 10$ selected patches.

We also report the standard deviation of SPSNet performance over five runs with random seeds $\{2022, 2023, 2024, 2025, 2026\}$ in Table 5. The results show that SPSNet achieves stable performance across different random seeds.

I SPS WITH PATCH-BASED BACKBONES

To evaluate whether SPS is effective beyond SPSNet, we integrate the SPS module into several existing patch-based forecasting backbones. Specifically, SPS replaces the original fixed patching or segmentation stage while preserving the remaining backbone architecture. This setup allows us to isolate the effect of information-preserving patch selection across different forecasting models.

Table 5: Robustness of SPSNet performance. Mean ± standard deviation over five runs with random seeds $\{2022, 2023, 2024, 2025, 2026\}$.

Datasets	ETTh1			ETTh2			ETTm1		
Horizon	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
96	0.381±0.002	0.392±0.001	0.0614±0.007	0.288±0.001	0.338±0.001	0.1892±0.005	0.311±0.002	0.348±0.002	0.1918±0.003
192	0.431±0.003	0.425±0.001	0.0695±0.002	0.367±0.001	0.387±0.000	0.1275±0.005	0.351±0.002	0.369±0.001	0.2420±0.003
336	0.468±0.003	0.444±0.002	0.1007±0.004	0.411±0.005	0.422±0.002	0.1504±0.004	0.388±0.001	0.394±0.001	0.2443±0.002
720	0.471±0.010	0.461±0.004	0.1803±0.027	0.418±0.005	0.436±0.005	0.2706±0.026	0.445±0.006	0.429±0.002	0.2463±0.001
Datasets	ETTm2			Weather			Solar		
Horizon	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
96	0.171±0.001	0.252±0.000	0.2038±0.002	0.166±0.002	0.206±0.003	0.1849±0.005	0.174±0.005	0.206±0.005	0.2039±0.005
192	0.239±0.002	0.296±0.001	0.1623±0.005	0.214±0.001	0.246±0.001	0.1634±0.004	0.192±0.005	0.212±0.004	0.1901±0.016
336	0.301±0.001	0.336±0.000	0.1353±0.001	0.272±0.001	0.288±0.002	0.1870±0.003	0.205±0.014	0.229±0.010	0.1834±0.024
720	0.397±0.001	0.392±0.001	0.1451±0.001	0.354±0.002	0.342±0.001	0.2219±0.006	0.210±0.004	0.237±0.004	0.2428±0.018
Datasets	Electricity			Traffic					
Horizon	MSE	MAE	IPS	MSE	MAE	IPS			
96	0.152±0.002	0.241±0.001	0.1577±0.001	0.448±0.006	0.270±0.002	0.3079±0.010			
192	0.170±0.003	0.257±0.002	0.2200±0.004	0.461±0.008	0.273±0.001	0.2828±0.005			
336	0.183±0.003	0.273±0.002	0.2059±0.004	0.467±0.004	0.283±0.002	0.2428±0.004			
720	0.220±0.003	0.305±0.002	0.2256±0.007	0.489±0.003	0.290±0.001	0.0718±0.003			

Detailed results are reported in Table 6. Across backbones, SPS improves information preservation and often improves forecasting accuracy, showing that the proposed selection mechanism is not tied to a specific architecture. Across all 8 datasets, SPS improves MSE by 2.93%, MAE by 4.10%, and IPS by 61.77%.

Table 6: Averaged results of SPS across different backbones.

Datasets	ETTh1			ETTh2			ETTm1			ETTm2		
Metrics	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
SRSNet	**0.450**	0.435	0.5148	0.372	0.399	3.7434	**0.388**	0.401	0.7597	0.279	0.325	2.6402
+ SPS	0.453	**0.432**	**0.2309**	**0.366**	**0.390**	**0.8177**	0.391	**0.389**	**0.4016**	**0.274**	**0.317**	**0.6377**
Improve	**-0.67%**	**+0.69%**	**+55.15%**	**+1.61%**	**+2.26%**	**+78.16%**	**-0.77%**	**+2.99%**	**+47.14%**	**+1.79%**	**+2.46%**	**+75.85%**
PatchMLP	0.469	0.452	0.5240	0.404	0.423	3.3011	0.398	0.405	0.7714	0.294	0.333	2.7359
+ SPS	**0.440**	**0.429**	**0.2172**	**0.361**	**0.385**	**0.6586**	**0.357**	**0.369**	**0.3832**	**0.273**	**0.316**	**0.5981**
Improve	**+6.18%**	**+5.09%**	**+58.55%**	**+10.64%**	**+8.98%**	**+80.05%**	**+10.30%**	**+8.89%**	**+50.32%**	**+7.14%**	**+5.11%**	**+78.14%**
PatchTST	**0.441**	0.435	0.5034	0.383	0.406	3.3441	0.378	0.395	0.7229	0.285	0.329	2.3596
+ SPS	0.442	**0.434**	**0.1994**	**0.374**	**0.399**	**0.6325**	**0.367**	**0.382**	**0.4016**	**0.272**	**0.316**	**0.6685**
Improve	**-0.23%**	**+0.23%**	**+60.39%**	**+2.35%**	**+1.72%**	**+81.09%**	**+2.91%**	**+3.29%**	**+44.45%**	**+4.56%**	**+3.95%**	**+71.67%**
Datasets	Weather			Solar			Electricity			Traffic		
Metrics	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
SRSNet	0.261	0.282	3.2484	0.260	0.284	2.3988	0.193	0.279	0.9968	0.497	0.315	8.1137
+ SPS	**0.245**	**0.265**	**1.3232**	**0.259**	**0.277**	**0.7523**	**0.190**	**0.273**	**0.4198**	**0.495**	**0.301**	**0.8063**
Improve	**+6.13%**	**+6.03%**	**+59.27%**	**+0.38%**	**+2.46%**	**+68.64%**	**+1.55%**	**+2.15%**	**+57.89%**	**+0.40%**	**+4.44%**	**+90.06%**
PatchMLP	0.252	0.279	2.3813	0.258	0.288	2.4189	0.193	0.287	1.0050	0.493	0.333	11.4062
+ SPS	**0.236**	**0.260**	**1.2509**	**0.252**	**0.243**	**1.2954**	**0.185**	**0.279**	**0.4471**	**0.482**	**0.301**	**1.5854**
Improve	**+6.35%**	**+6.81%**	**+47.47%**	**+2.33%**	**+15.63%**	**+46.45%**	**+4.15%**	**+2.79%**	**+55.51%**	**+2.23%**	**+9.61%**	**+86.10%**
PatchTST	0.248	0.275	1.9133	**0.224**	**0.261**	2.4266	0.183	0.273	0.9874	0.515	0.291	3.3374
+ SPS	**0.242**	**0.268**	**1.1893**	0.231	0.264	**1.4319**	**0.182**	**0.271**	**0.5638**	**0.509**	**0.289**	**1.0574**
Improve	**+2.42%**	**+2.55%**	**+37.84%**	**-3.13%**	**-1.15%**	**+40.99%**	**+0.55%**	**+0.73%**	**+42.90%**	**+1.17%**	**+0.69%**	**+68.32%**

We also address the potential circularity of evaluating SPS with the proposed IPS metric. Since SPS is explicitly trained with a posterior-preservation objective related to the KL divergence measured by IPS, comparing methods only by IPS would be insufficient. Therefore, we report IPS together with MSE and MAE to verify whether improved information preservation is accompanied by competitive forecasting accuracy.

To control for potential bias in the IPS evaluation, we conduct an additional experiment in which patch-based baselines are trained with the same KL regularization term used by SPS. Specifically,

we evaluate SRSNet+KL, PatchMLP+KL, and PatchTST+KL, trained with

$$\mathcal{L} = \mathcal{L}_{\text{point}} + \mathcal{L}_{\text{KL}}, \tag{62}$$

where $\mathcal{L}_{\text{KL}}$ is the same posterior-preservation term used in the SPS objective. We then compare these KL-regularized baselines with their SPS-enhanced counterparts in Table 7.

Table 7: Averaged results of KL and SPS improvements across different backbones.

Datasets	ETTh1			ETTh2			ETTm1			ETTm2			Weather		
Metrics	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
SRSNet	**0.450**	0.435	0.5148	0.372	0.399	3.7434	0.388	0.401	0.7597	0.279	0.325	2.6402	0.261	0.282	3.2484
+ KL	0.454	0.439	0.2959	0.374	0.402	0.9212	**0.384**	0.397	0.4294	0.276	0.321	0.7026	0.246	0.273	1.8075
KL Imp	**-0.89%**	**-0.92%**	**+42.52%**	**-0.54%**	**-0.75%**	**+75.39%**	**+1.03%**	**+1.00%**	**+43.48%**	**+1.08%**	**+1.23%**	**+73.39%**	**+5.75%**	**+3.19%**	**+44.36%**
+ SPS	0.453	**0.432**	**0.2309**	**0.366**	**0.390**	**0.8177**	0.391	**0.389**	**0.4016**	**0.274**	**0.317**	**0.6377**	**0.245**	**0.265**	**1.3232**
Imp over KL	**+0.22%**	**+1.59%**	**+21.97%**	**+2.14%**	**+2.99%**	**+11.24%**	**-1.82%**	**+2.02%**	**+6.47%**	**+0.72%**	**+1.25%**	**+9.24%**	**+0.41%**	**+2.93%**	**+26.79%**
PatchMLP	0.469	0.452	0.5240	0.404	0.423	3.3011	0.398	0.405	0.7714	0.294	0.333	2.7359	0.252	0.279	2.3813
+ KL	0.457	0.444	0.3013	0.384	0.408	0.9377	0.389	0.397	0.4174	0.279	0.324	0.7097	0.242	0.271	1.7599
KL Imp	**+2.56%**	**+1.77%**	**+42.50%**	**+4.95%**	**+3.55%**	**+71.59%**	**+2.26%**	**+1.98%**	**+45.89%**	**+5.10%**	**+2.70%**	**+74.06%**	**+3.97%**	**+2.87%**	**+26.09%**
+ SPS	**0.440**	**0.429**	**0.2172**	**0.361**	**0.385**	**0.6586**	**0.357**	**0.369**	**0.3832**	**0.273**	**0.316**	**0.5981**	**0.236**	**0.260**	**1.2509**
Imp over KL	**+3.72%**	**+3.38%**	**+27.91%**	**+5.99%**	**+5.64%**	**+29.76%**	**+8.23%**	**+7.05%**	**+8.19%**	**+2.15%**	**+2.47%**	**+15.72%**	**+2.48%**	**+4.06%**	**+28.92%**
PatchTST	0.441	0.435	0.5034	0.383	0.406	3.3441	0.378	0.395	0.7229	0.285	0.329	2.3596	0.248	0.275	1.9133
+ KL	**0.438**	**0.432**	0.2524	**0.374**	0.401	0.8847	0.373	0.388	**0.3813**	0.279	0.323	0.7075	0.247	0.271	1.4194
KL Imp	**+0.68%**	**+0.69%**	**+49.86%**	**+2.35%**	**+1.23%**	**+73.54%**	**+1.32%**	**+1.77%**	**+47.25%**	**+2.11%**	**+1.82%**	**+70.02%**	**+0.40%**	**+1.45%**	**+25.81%**
+ SPS	0.442	0.434	**0.1994**	**0.374**	**0.399**	**0.6325**	**0.367**	**0.382**	0.4016	**0.272**	**0.316**	**0.6685**	**0.242**	**0.268**	**1.1893**
Imp over KL	**-0.91%**	**-0.46%**	**+21.00%**	**+0.00%**	**+0.50%**	**+28.51%**	**+1.61%**	**+1.55%**	**-5.32%**	**+2.51%**	**+2.17%**	**+5.51%**	**+2.02%**	**+1.11%**	**+16.21%**

The KL term itself improves the baseline models even without SPS: averaged over the five datasets and three backbones, KL regularization improves MSE by 2.14%, MAE by 1.57%, and IPS by 53.72% over the original backbones, indicating that posterior preservation is useful beyond IPS evaluation. However, SPS still improves both forecasting accuracy and information preservation under this controlled setting, where the baselines are trained with the same KL regularization: SPS-enhanced variants further improve MSE by 2.55%, MAE by 2.55%, and IPS by 16.81% over their KL-regularized counterparts. This shows that the gains come not only from the KL objective, but also from the proposed SPS mechanism.

J Ablation Studies

We conduct ablation studies to evaluate the contribution of the main SPS components. Specifically, we consider four variants. First, *w/o Alignment Loss* removes the alignment penalty by setting $\lambda_{\text{align}} = 0$. Second, *w/o KL Loss* removes the posterior-preservation objective by setting $\lambda_{\text{KL}} = 0$. Third, *w/o Annealing Epochs* disables gradual keep-ratio annealing by setting the number of annealing epochs to $E_{\text{anneal}} = 1$. Finally, *w/o Candidate Selection* removes the selection bottleneck by setting the selected patch budget K equal to the candidate budget N.

The averaged ablation results are reported in Table 8. Overall, the full SPSNet achieves the best performance across datasets and metrics, showing that the proposed components work together to improve both forecasting accuracy and information preservation. Removing the KL loss causes a large increase in IPS, confirming that the posterior-preservation objective is essential for learning representations that match the full-history predictive distribution. Removing the alignment loss also weakens IPS, indicating that aligning soft retention weights with the hard top-K decision stabilizes selection. The variant without annealing performs competitively on some datasets, but is less stable overall, supporting the use of gradual keep-ratio annealing.

The *w/o Candidate Selection* variant produces results close to the full model because it uses all $N = 80$ candidate tokens, effectively removing the compression bottleneck. However, this comes at the cost of higher runtime and memory usage, since the downstream module must process the full candidate set instead of the compact K-token representation. Thus, candidate selection is important not only for accuracy, but also for achieving the intended efficiency benefit of SPS. Overall, the ablation results support the design of SPS: effective patch selection benefits from posterior preservation, soft-to-hard alignment, annealing, and an explicit selection bottleneck.

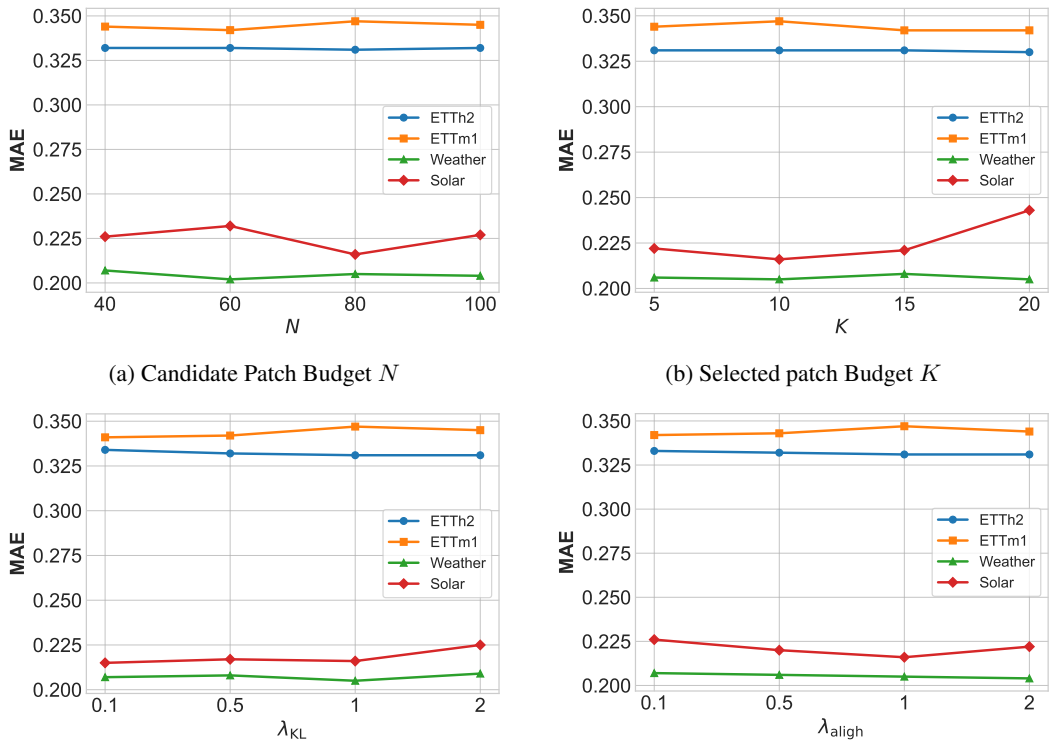
Table 8: Averaged results of Ablation Studies of key components in SPS.

Datasets	ETTh1			ETTh2			ETTm1			ETTm2			Weather		
Metrics	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
w/o Alignment Loss	0.456	0.440	0.1390	0.377	0.405	0.2274	0.391	0.388	0.2629	0.281	0.325	0.1808	0.266	0.286	0.1975
w/o KL Loss	0.443	0.432	0.4630	0.377	0.400	0.6872	0.391	0.388	0.6428	0.283	0.327	0.7146	0.258	0.275	0.8431
w/o Annealing Epochs	0.444	0.432	0.1279	0.374	0.401	0.2192	0.387	0.386	0.2635	0.281	0.324	0.1827	0.282	0.276	0.2250
w/o Candidate Selection	0.441	0.431	0.1285	0.375	0.402	0.2275	0.386	0.388	0.2543	0.279	0.322	0.1954	0.253	0.273	0.2208
SPSNet	**0.439**	**0.430**	**0.0977**	**0.371**	**0.396**	**0.1748**	**0.373**	**0.385**	**0.2239**	**0.277**	**0.319**	**0.1555**	**0.252**	**0.271**	**0.1226**

K Hyperparameter Sensitivity

We analyze the sensitivity of SPSNet to four key hyperparameters: the candidate patch budget $N \in \{40, 60, 80, 100\}$, the selected patch budget $K \in \{5, 10, 15, 20\}$, the KL loss weight $\lambda_{\text{KL}} \in \{0.1, 0.5, 1, 2\}$, and the alignment loss weight $\lambda_{\text{align}} \in \{0.1, 0.5, 1, 2\}$.

The results are shown in Figure 3. Overall, SPSNet remains stable across a broad range of hyperparameter values. Changing the candidate and selected patch budgets affects the compactness–accuracy trade-off, but the model does not rely on a narrowly tuned configuration. Similarly, varying the auxiliary loss weights produces only modest changes. Therefore, we use unit weights for all terms in the training objective, avoiding additional loss-balancing hyperparameters.



(a) Candidate Patch Budget N

(b) Selected patch Budget K

(c) KL Loss Weight λ_{KL}

(d) Alignment Loss Weight λ_{align}

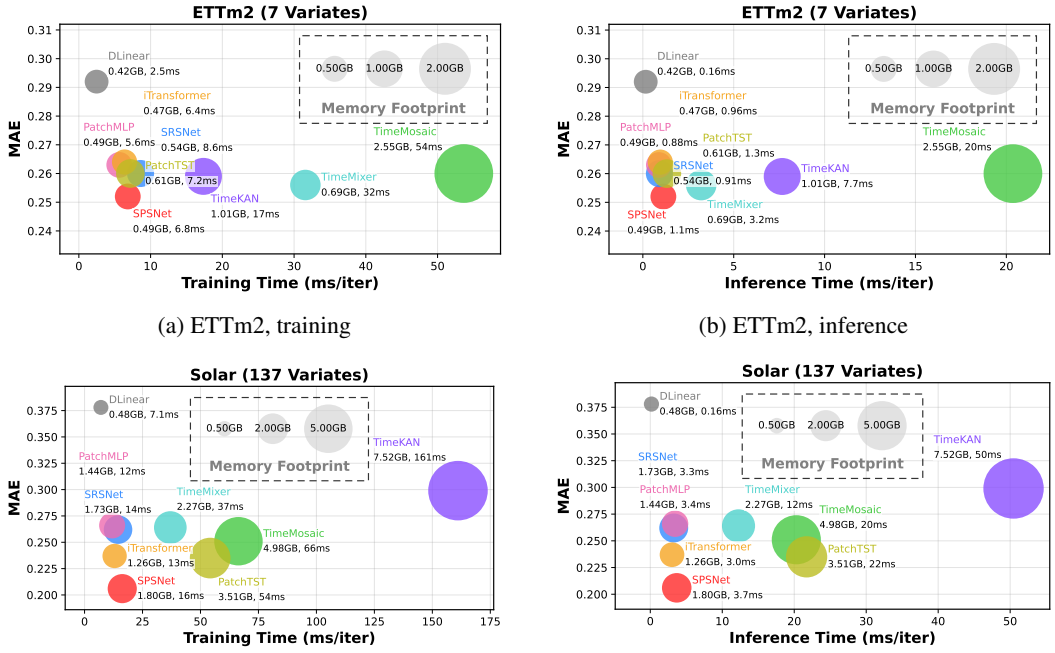
Figure 3: Hyperparameter sensitivity analysis of SPSNet.

L Model Efficiency

We compare the efficiency of SPSNet with representative forecasting baselines in terms of forecasting accuracy, training time, inference time, and GPU memory footprint. All models are run with

their official implementations under the unified setting of lookback $L = 96$ and horizon $H = 96$, with the same batch size of 64, on a single NVIDIA RTX PRO 5000 GPU. Training time is the wall-clock time of one training iteration (data loading, forward and backward pass, and optimizer step). Inference time is the latency of one forward pass on a batch of 64 samples, and memory is the peak GPU memory of the process. Figure 4 visualizes the trade-off between MAE and computational cost, where the bubble size denotes peak GPU memory usage.

SPSNet achieves the lowest MAE, while its training time, inference time, and memory footprint remain comparable to those of lightweight baselines and below those of recent models such as TimeMixer, TimeKAN, and TimeMosaic. Overall, these results show that SPSNet provides a favorable trade-off between forecasting accuracy and efficiency.



(a) ETTm2, training

(b) ETTm2, inference

(c) Solar, training

(d) Solar, inference

Figure 4: Efficiency comparison of SPSNet.

M FULL RESULTS

Table 9 demonstrates the full multivariate forecasting results following the unified lookback window $L = 96$ for all datasets.

Since some baselines were originally designed to capture longer dependencies using a longer historical window (TimeMosaic, SRSNet, PatchTST), we conducted an additional experiment with their preferred setting. Table 10 demonstrates the full multivariate forecasting results following the unified longer lookback window $L = 336$ for all datasets. TimeMosaic and SRSNet were run with the official code and no parameter change except the lookback (originally TimeMosaic L=320, SRSNet L=512; we set L=336 for all for a direct comparison). PatchTST was run at its preferred L=336 with its original parameters, without the drop-last trick. Under this setting, SPSNet obtains the best MAE on 8/8 datasets and the best MSE on 7/8, while retaining the smallest token budget of all compared methods: K=10 tokens (selected from N=320 candidates) vs 42 for TimeMosaic, 41 for SRSNet (N=321), and 42 for PatchTST — a 4× smaller representation.

Table 11 demonstrates the full multivariate forecasting results following the unified lookback window $L = 96$ for all datasets with the reported Information Preservation Sufficiency (IPS) metric. The IPS evaluation is conducted on patch-based models to compare the information preservation capability of different patching strategies.

Table 9: Full multivariate forecasting results with unified lookback window $L = 96$. Avg means the average results from all four forecasting horizons. **Red** denotes the best, while blue stands for the second best.

Models		SPSNet (ours)		TimeMosaic (2026)		SRSNet (2025)		TimeKAN (2025)		PatchMLP (2025)		iTransformer (2024)		TimeMixer (2024)		PatchTST (2023)	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.381	**0.392**	0.377	0.397	0.381	0.393	**0.370**	0.396	0.404	0.413	0.386	0.405	0.386	0.404	0.375	0.394
	192	0.431	0.425	0.435	0.432	0.436	0.424	**0.418**	**0.423**	0.456	0.441	0.441	0.436	0.445	0.437	0.430	0.428
	336	0.468	0.444	0.466	0.451	0.483	0.447	**0.447**	**0.441**	0.500	0.463	0.487	0.458	0.490	0.455	0.476	0.447
	720	0.471	**0.461**	0.527	0.501	0.499	0.476	**0.458**	0.468	0.517	0.492	0.503	0.491	0.501	0.480	0.483	0.469
	Avg	0.438	**0.431**	0.451	0.445	0.450	0.435	**0.423**	0.432	0.469	0.452	0.454	0.448	0.456	0.444	0.441	0.435
ETTh2	96	0.288	**0.338**	0.293	0.343	**0.287**	0.339	0.293	0.342	0.320	0.364	0.297	0.349	0.290	0.343	0.292	0.342
	192	0.367	**0.387**	0.381	0.398	**0.366**	0.388	0.379	0.394	0.396	0.413	0.380	0.400	0.371	0.394	0.367	0.394
	336	0.411	**0.422**	0.431	0.431	**0.409**	0.424	0.424	0.431	0.442	0.448	0.428	0.432	0.422	0.435	0.419	0.429
	720	**0.418**	**0.436**	0.436	0.449	0.426	0.445	0.450	0.456	0.456	0.465	0.427	0.445	0.439	0.451	0.452	0.458
	Avg	**0.371**	**0.396**	0.385	0.405	0.372	0.399	0.387	0.406	0.404	0.423	0.383	0.407	0.381	0.406	0.383	0.406
ETTm1	96	**0.311**	**0.348**	0.327	0.366	0.329	0.369	0.327	0.365	0.332	0.368	0.334	0.368	0.314	0.354	0.316	0.356
	192	**0.351**	**0.369**	0.373	0.389	0.365	0.384	0.360	0.384	0.374	0.390	0.377	0.391	0.370	0.387	0.359	0.381
	336	0.388	**0.394**	0.400	0.409	0.396	0.406	**0.381**	0.400	0.410	0.413	0.426	0.420	0.398	0.407	0.384	0.403
	720	**0.445**	**0.429**	0.465	0.448	0.461	0.444	0.454	0.440	0.477	0.448	0.491	0.459	0.458	0.444	0.452	0.440
	Avg	**0.374**	**0.385**	0.391	0.403	0.388	0.401	0.381	0.397	0.398	0.405	0.407	0.410	0.385	0.398	0.378	0.395
ETTm2	96	**0.171**	**0.252**	0.179	0.260	0.175	0.260	0.177	0.259	0.180	0.263	0.180	0.264	0.174	0.256	0.177	0.260
	192	**0.239**	**0.296**	0.241	0.302	0.240	0.301	0.241	0.302	0.249	0.307	0.250	0.309	0.240	0.302	0.244	0.305
	336	0.301	**0.336**	0.300	0.339	0.301	0.341	0.306	0.347	0.320	0.351	0.311	0.348	**0.296**	0.338	0.309	0.345
	720	0.397	**0.392**	0.398	0.395	0.399	0.399	0.411	0.408	0.426	0.411	0.412	0.407	**0.388**	0.393	0.410	0.405
	Avg	0.277	**0.319**	0.280	0.324	0.279	0.325	0.284	0.329	0.294	0.333	0.288	0.332	**0.275**	0.322	0.285	0.329
Weather	96	0.166	**0.206**	0.175	0.217	0.183	0.224	**0.162**	0.208	0.168	0.215	0.174	0.214	0.166	0.212	0.166	0.211
	192	0.214	**0.246**	0.219	0.255	0.227	0.260	**0.207**	0.249	0.217	0.257	0.221	0.254	0.209	0.254	0.212	0.253
	336	0.272	**0.288**	0.278	0.296	0.280	0.298	**0.263**	0.290	0.272	0.296	0.278	0.296	0.265	0.291	0.268	0.293
	720	0.354	**0.342**	0.356	0.348	0.353	0.346	**0.340**	0.343	0.352	0.347	0.358	0.347	0.347	0.347	0.344	0.343
	Avg	0.252	**0.271**	0.257	0.279	0.261	0.282	**0.243**	0.273	0.252	0.279	0.258	0.278	0.247	0.276	0.248	0.275
Solar	96	**0.174**	**0.206**	0.209	0.251	0.225	0.262	0.239	0.299	0.218	0.266	0.203	0.237	0.218	0.264	0.189	0.235
	192	**0.192**	**0.212**	0.252	0.290	0.254	0.281	0.271	0.315	0.252	0.286	0.233	0.261	0.242	0.287	0.218	0.255
	336	**0.205**	**0.229**	0.263	0.299	0.279	0.295	0.297	0.326	0.277	0.298	0.248	0.273	0.246	0.313	0.245	0.278
	720	**0.210**	**0.237**	0.261	0.291	0.282	0.297	0.298	0.323	0.285	0.302	0.249	0.275	0.239	0.303	0.242	0.275
	Avg	**0.195**	**0.221**	0.246	0.283	0.260	0.284	0.276	0.316	0.258	0.288	0.233	0.262	0.236	0.292	0.224	0.261
Electricity	96	0.152	0.241	0.165	0.260	0.167	0.255	0.176	0.265	0.157	0.254	**0.148**	**0.240**	0.153	0.245	0.156	0.247
	192	0.170	0.257	0.178	0.271	0.177	0.264	0.182	0.271	0.176	0.272	**0.162**	**0.253**	0.167	0.257	0.169	0.259
	336	0.183	0.273	0.192	0.284	0.193	0.281	0.198	0.286	0.197	0.290	**0.178**	**0.269**	0.185	0.275	0.184	0.275
	720	**0.220**	**0.305**	0.244	0.327	0.235	0.315	0.241	0.322	0.243	0.332	0.225	0.317	0.224	0.310	0.223	0.310
	Avg	0.181	**0.269**	0.195	0.286	0.193	0.279	0.199	0.286	0.193	0.287	**0.178**	0.270	0.182	0.272	0.183	0.273
Traffic	96	0.448	0.270	0.419	0.274	0.478	0.309	0.592	0.381	0.469	0.326	**0.395**	**0.268**	0.446	0.279	0.483	0.278
	192	0.461	**0.273**	0.499	0.336	0.480	0.308	0.550	0.364	0.468	0.321	**0.417**	0.276	0.460	0.284	0.503	0.283
	336	0.467	**0.283**	0.512	0.350	0.495	0.312	0.564	0.363	0.493	0.332	**0.433**	**0.283**	0.471	0.294	0.526	0.293
	720	0.489	**0.290**	0.522	0.338	0.534	0.330	0.612	0.386	0.540	0.351	**0.467**	0.302	0.502	0.305	0.547	0.308
	Avg	0.466	**0.279**	0.488	0.325	0.497	0.315	0.580	0.374	0.493	0.333	**0.428**	0.282	0.470	0.291	0.515	0.291
1^{st} Count		**11**	**26**	0	0	3	0	9	2	0	0	7	5	2	0	0	0

N VISUALIZATIONS

We provide qualitative visualizations to better understand the behavior of SPS across different forecasting backbones. Figures 5–7 compare the proposed SPS approach across SPSNet, SRSNet, PatchMLP, and PatchTST backbones. These visualizations show how SPS selects compact, informative temporal intervals while preserving the main predictive structure of the input sequence.

Table 10: Full multivariate forecasting results with unified lookback window $L = 336$. Avg means the average results from all four forecasting horizons. **Red** denotes the best, while blue stands for the second best.

Models		SPSNet (ours)		TimeMosaic (2026)		SRSNet (2025)		PatchTST (2023)	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.372	0.397	**0.360**	**0.385**	0.376	0.398	0.378	0.400
	192	0.412	0.422	**0.398**	**0.412**	0.412	0.420	0.413	0.421
	336	**0.431**	**0.430**	0.457	0.456	0.435	0.437	0.440	0.440
	720	**0.450**	**0.459**	0.454	0.470	0.472	0.471	0.456	0.471
	Avg	**0.416**	**0.427**	0.417	0.431	0.424	0.432	0.422	0.433
ETTh2	96	0.282	0.339	0.293	0.339	0.292	0.349	**0.274**	**0.337**
	192	0.353	0.383	0.359	**0.382**	0.357	0.390	**0.348**	0.384
	336	**0.374**	**0.404**	0.387	0.408	0.381	0.412	0.377	0.416
	720	0.409	**0.431**	0.428	0.442	0.451	0.467	**0.406**	0.441
	Avg	0.355	**0.389**	0.367	0.393	0.370	0.405	**0.351**	0.395
ETTm1	96	**0.283**	**0.330**	0.293	0.333	0.298	0.345	0.291	0.342
	192	**0.323**	**0.355**	0.340	0.366	0.332	0.370	0.330	0.368
	336	**0.360**	**0.377**	0.367	0.387	0.368	0.398	0.366	0.392
	720	**0.417**	**0.412**	0.440	0.423	0.428	0.433	**0.417**	0.423
	Avg	**0.346**	**0.369**	0.360	0.377	0.357	0.387	0.351	0.381
ETTm2	96	**0.161**	**0.245**	0.165	0.248	0.173	0.256	0.164	0.254
	192	**0.219**	**0.285**	0.234	0.294	0.237	0.303	**0.219**	0.291
	336	**0.273**	**0.319**	0.274	0.322	0.279	0.329	0.276	0.328
	720	**0.351**	**0.372**	0.368	0.382	0.365	0.389	0.368	0.385
	Avg	**0.251**	**0.305**	0.260	0.312	0.264	0.319	0.257	0.315
Weather	96	**0.144**	**0.188**	0.154	0.192	0.152	0.202	0.151	0.199
	192	0.195	**0.232**	**0.193**	**0.232**	0.197	0.249	0.198	0.242
	336	**0.245**	0.274	0.246	**0.272**	0.247	0.283	0.253	0.283
	720	0.327	**0.327**	0.329	0.328	**0.322**	0.335	0.324	0.334
	Avg	**0.228**	**0.255**	0.231	0.256	0.230	0.267	0.232	0.265
Solar	96	**0.163**	**0.196**	0.200	0.226	0.193	0.240	0.190	0.273
	192	**0.179**	**0.211**	0.225	0.240	0.204	0.255	0.204	0.302
	336	**0.196**	**0.222**	0.238	0.245	0.214	0.261	0.212	0.293
	720	**0.212**	**0.225**	0.241	0.251	0.218	0.265	0.221	0.310
	Avg	**0.188**	**0.214**	0.226	0.241	0.207	0.255	0.207	0.295
Electricity	96	**0.129**	**0.220**	0.139	0.236	0.141	0.238	0.178	0.268
	192	**0.146**	**0.236**	0.155	0.250	0.160	0.257	0.191	0.283
	336	**0.164**	**0.256**	0.172	0.266	0.174	0.270	0.213	0.304
	720	**0.204**	**0.290**	0.212	0.298	0.214	0.303	0.210	0.298
	Avg	**0.161**	**0.251**	0.170	0.263	0.172	0.267	0.198	0.288
Traffic	96	**0.376**	**0.245**	0.408	0.274	0.406	0.285	0.379	0.271
	192	**0.392**	**0.258**	0.423	0.280	0.420	0.294	0.394	0.277
	336	**0.401**	**0.269**	0.437	0.285	0.437	0.302	0.404	0.281
	720	**0.441**	**0.289**	0.462	0.301	0.453	0.310	0.442	0.302
	Avg	**0.403**	**0.265**	0.433	0.285	0.429	0.298	0.405	0.283
1^{st} Count		**25**	**27**	3	5	1	0	5	1

O DISCUSSION

While SPS improves information preservation and reduces redundant patch tokens, it has several limitations. First, since the true posterior $P(Y \mid X)$ is generally inaccessible, SPS relies on a full-history teacher approximation $q_T(Y \mid X)$. Although the teacher is not required at inference, and a single teacher is shared by all models on a given dataset and horizon, its quality can affect the posterior-preservation signal.

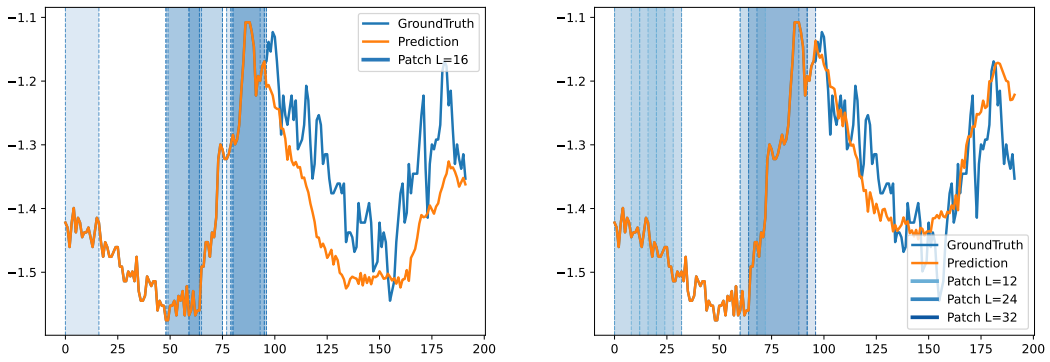
Second, the current implementation uses a fixed candidate stride and fixed candidate and selection budgets. These choices improve consistency across datasets, but they may not be optimal for all

Table 11: Full multivariate forecasting results with unified lookback $L = 96$ and reported IPS score. Avg means the average results from all four forecasting horizons. **Red** denotes the best, while blue stands for the second best.

Models		SPSNet (ours)			TimeMosaic (2026)			SRSNet (2025)			PatchMLP (2025)			PatchTST (2023)		
Metrics		MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS	MSE	MAE	IPS
ETTh1	96	0.381	**0.392**	**0.0614**	0.377	0.397	0.2254	0.381	0.393	0.5691	0.404	0.413	0.5845	**0.375**	0.394	0.5582
	192	0.431	0.425	**0.0695**	0.435	0.432	0.2896	0.436	**0.424**	0.5065	0.456	0.441	0.5108	**0.430**	0.428	0.4715
	336	0.468	**0.444**	**0.1007**	**0.466**	0.451	0.2326	0.483	0.447	0.4807	0.500	0.463	0.4940	0.476	0.447	0.4817
	720	**0.471**	**0.461**	**0.1803**	0.527	0.501	0.2950	0.499	0.476	0.5030	0.517	0.492	0.5066	0.483	0.469	0.5023
	Avg	**0.438**	**0.431**	**0.1030**	0.451	0.445	0.2607	0.450	0.435	0.5148	0.469	0.452	0.5240	0.441	0.435	0.5034
ETTh2	96	0.288	**0.338**	**0.1892**	0.293	0.343	2.0413	**0.287**	0.339	2.1756	0.320	0.364	1.5831	0.292	0.342	1.6789
	192	0.367	**0.387**	**0.1275**	0.381	0.398	5.4711	**0.366**	0.388	5.2864	0.396	0.413	3.9967	0.367	0.394	3.2865
	336	0.411	**0.422**	**0.1504**	0.431	0.431	2.6097	**0.409**	0.424	2.4092	0.442	0.448	2.3777	0.419	0.429	2.1404
	720	**0.418**	**0.436**	**0.2706**	0.436	0.449	5.6237	0.426	0.445	5.1022	0.456	0.465	5.2470	0.452	0.458	6.2706
	Avg	**0.371**	**0.396**	**0.1844**	0.385	0.405	3.9365	0.372	0.399	3.7434	0.404	0.423	3.3011	0.383	0.406	3.3441
ETTm1	96	**0.311**	**0.348**	**0.1918**	0.327	0.366	0.5695	0.329	0.369	0.8268	0.332	0.368	0.8505	0.316	0.356	0.8061
	192	**0.351**	**0.369**	**0.2420**	0.373	0.389	0.6187	0.365	0.384	0.7970	0.374	0.390	0.8069	0.359	0.381	0.7451
	336	0.388	**0.394**	**0.2443**	0.400	0.409	0.5993	0.396	0.406	0.7323	0.410	0.413	0.7333	**0.384**	0.403	0.7101
	720	**0.445**	**0.429**	**0.2463**	0.465	0.448	0.5680	0.461	0.444	0.6826	0.477	0.448	0.6949	0.452	0.440	0.6302
	Avg	**0.374**	**0.385**	**0.2311**	0.391	0.403	0.5889	0.388	0.401	0.7597	0.398	0.405	0.7714	0.378	0.395	0.7229
ETTm2	96	**0.171**	**0.252**	**0.2038**	0.179	0.260	2.6898	0.175	0.260	1.7610	0.180	0.263	1.4437	0.177	0.260	1.4910
	192	**0.239**	**0.296**	**0.1623**	0.241	0.302	16.2470	0.240	0.301	2.7331	0.249	0.307	2.2152	0.244	0.305	1.9005
	336	0.301	**0.336**	**0.1353**	**0.300**	0.339	10.2140	0.301	0.341	2.9854	0.320	0.351	3.2710	0.309	0.345	2.5759
	720	**0.397**	**0.392**	**0.1451**	0.398	0.395	15.4520	0.399	0.399	3.0814	0.426	0.411	4.0136	0.410	0.405	3.4709
	Avg	**0.277**	**0.319**	**0.1616**	0.280	0.324	11.1507	0.279	0.325	2.6402	0.294	0.333	2.7359	0.285	0.329	2.3596
Weather	96	**0.166**	**0.206**	**0.1849**	0.175	0.217	0.2114	0.183	0.224	1.9629	0.168	0.215	1.1636	**0.166**	0.211	1.0212
	192	0.214	**0.246**	**0.1634**	0.219	0.255	0.2578	0.227	0.260	2.4892	0.217	0.257	2.5861	**0.212**	0.253	1.1948
	336	0.272	**0.288**	**0.1870**	0.278	0.296	0.3188	0.280	0.298	3.7645	0.272	0.296	2.7132	**0.268**	0.293	2.3107
	720	0.354	**0.342**	**0.2219**	0.356	0.348	0.2863	0.353	0.346	4.7768	0.352	0.347	3.0621	**0.344**	0.343	3.1265
	Avg	0.252	**0.271**	**0.1893**	0.257	0.279	0.2686	0.261	0.282	3.2484	0.252	0.279	2.3813	**0.248**	0.275	1.9133
Solar	96	**0.174**	**0.206**	**0.2039**	0.209	0.251	1.4652	0.225	0.262	2.3750	0.218	0.266	2.3827	0.189	0.235	2.4659
	192	**0.192**	**0.212**	**0.1901**	0.252	0.290	1.4733	0.254	0.281	2.3903	0.252	0.286	2.4373	0.218	0.255	2.4055
	336	**0.205**	**0.229**	**0.1834**	0.263	0.299	1.5157	0.279	0.295	2.4411	0.277	0.298	2.4615	0.245	0.278	2.4598
	720	**0.210**	**0.237**	**0.2428**	0.261	0.291	1.5658	0.282	0.297	2.3888	0.285	0.302	2.3942	0.242	0.275	2.3752
	Avg	**0.195**	**0.221**	**0.2051**	0.246	0.283	1.5050	0.260	0.284	2.3988	0.258	0.288	2.4189	0.224	0.261	2.4266
Electricity	96	**0.152**	**0.241**	**0.1577**	0.165	0.260	1.0140	0.167	0.255	0.9756	0.157	0.254	1.0048	0.156	0.247	0.9881
	192	0.170	**0.257**	**0.2200**	0.178	0.271	0.9035	0.177	0.264	0.9249	0.176	0.272	0.9152	**0.169**	0.259	0.9123
	336	**0.183**	**0.273**	**0.2059**	0.192	0.284	0.7361	0.193	0.281	0.9549	0.197	0.290	0.9629	0.184	0.275	0.9580
	720	**0.220**	**0.305**	**0.2256**	0.244	0.327	0.6270	0.235	0.315	1.1316	0.243	0.332	1.1370	0.223	0.310	1.0911
	Avg	**0.181**	**0.269**	**0.2023**	0.195	0.286	0.8202	0.193	0.279	0.9968	0.193	0.287	1.0050	0.183	0.273	0.9874
Traffic	96	0.448	**0.270**	**0.3079**	**0.419**	0.274	42.6463	0.478	0.309	8.6764	0.469	0.326	17.4890	0.483	0.278	5.1987
	192	**0.461**	**0.273**	**0.2828**	0.499	0.336	60.9160	0.480	0.308	8.8918	0.468	0.321	13.1752	0.503	0.283	2.4675
	336	**0.467**	**0.283**	**0.2428**	0.512	0.350	91.2550	0.495	0.312	7.9170	0.493	0.332	7.0053	0.526	0.293	2.5117
	720	**0.489**	**0.290**	**0.0718**	0.522	0.338	120.5300	0.534	0.330	6.9696	0.540	0.351	7.9553	0.547	0.308	3.1715
	Avg	**0.466**	**0.279**	**0.2263**	0.488	0.325	78.8368	0.497	0.315	8.1137	0.493	0.333	11.4062	0.515	0.291	3.3374
1^{st} Count		**19**	**31**	**32**	3	0	0	3	1	0	0	0	0	8	0	0

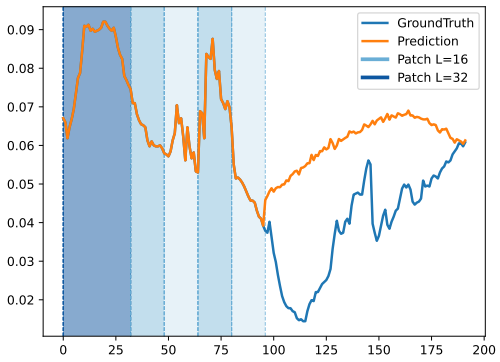
temporal resolutions or forecasting horizons. Adaptive stride and budget selection could further improve the trade-off between compactness and accuracy.

Third, SPSNet is intentionally designed as a simple MLP-based architecture. While this demonstrates that SPS can be effective without a heavy backbone, it does not fully characterize the performance potential of SPS. Future work could combine SPS with larger Transformer-based or foundation-style forecasting models to better assess its upper-bound performance.

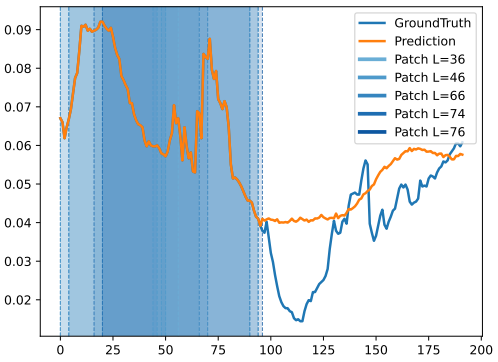


(a) SRSNet (SRS patching), ETTm1

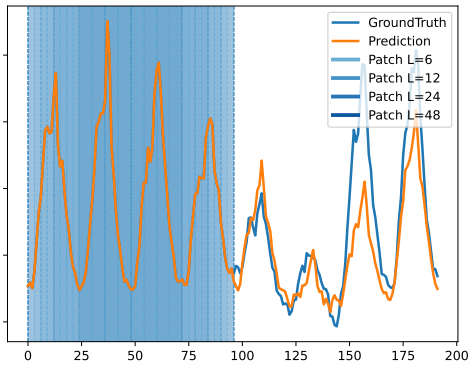
(b) SRSNet + SPS, ETTm1



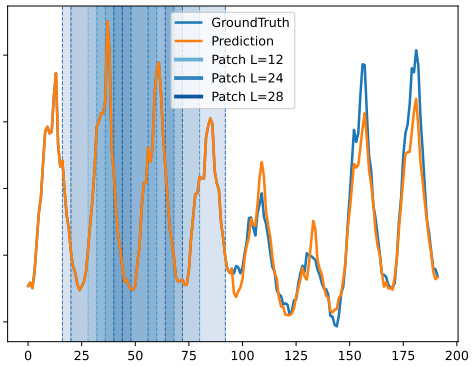
(c) TimeMosaic (adaptive patching), Weather



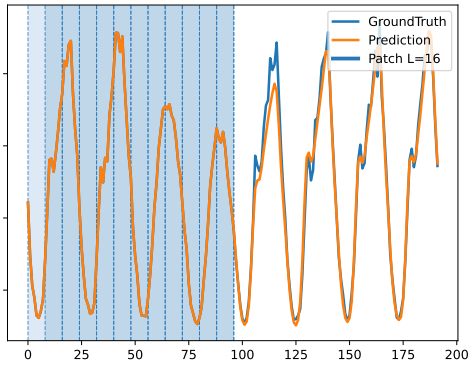
(d) TimeMosaic + SPS, Weather



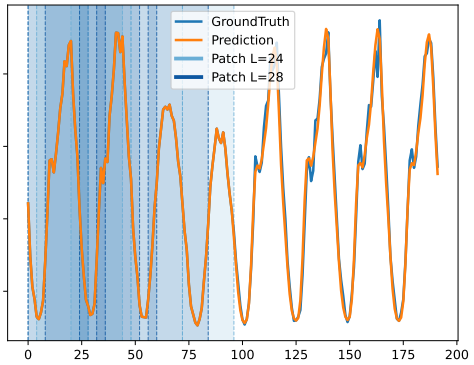
(e) PatchMLP (multi-scale patching), Electricity



(f) PatchMLP + SPS, Electricity

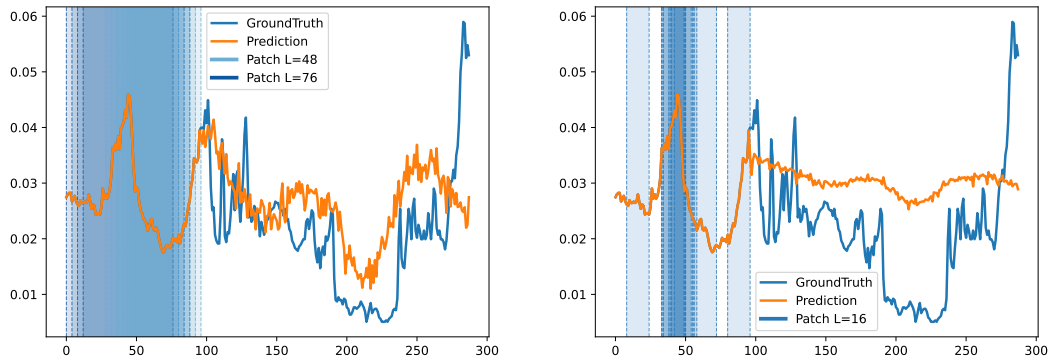


(g) PatchTST (fixed patching), Traffic



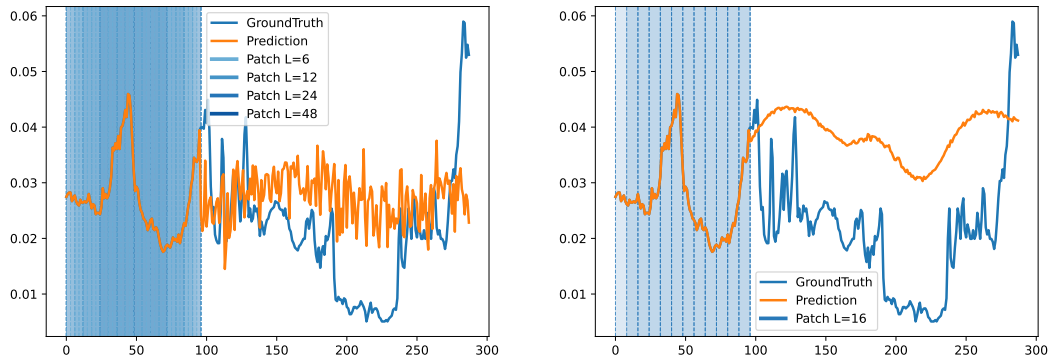
(h) PatchTST + SPS, Traffic

Figure 5: Qualitative visualizations of the SPS module with different backbones.



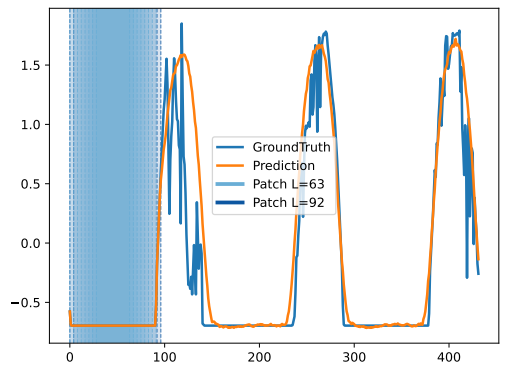
(a) SPSNet (SPS patching), Weather

(b) SRSNet (SRS patching), Weather

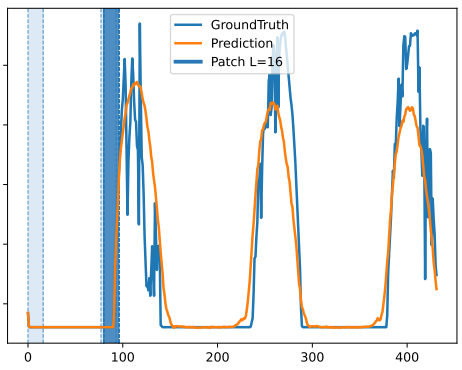


(c) PatchMLP (multi-scale patching), Weather

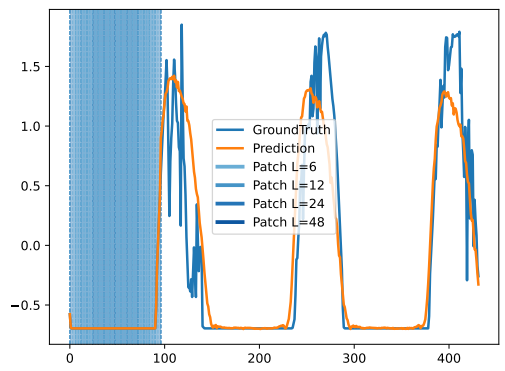
(d) PatchTST (fixed patching), Weather



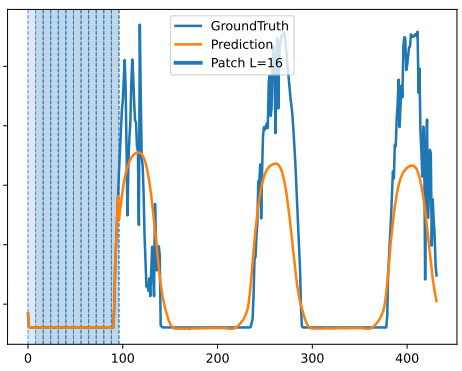
(e) SPSNet (SPS patching), Solar



(f) SRSNet (SRS patching), Solar

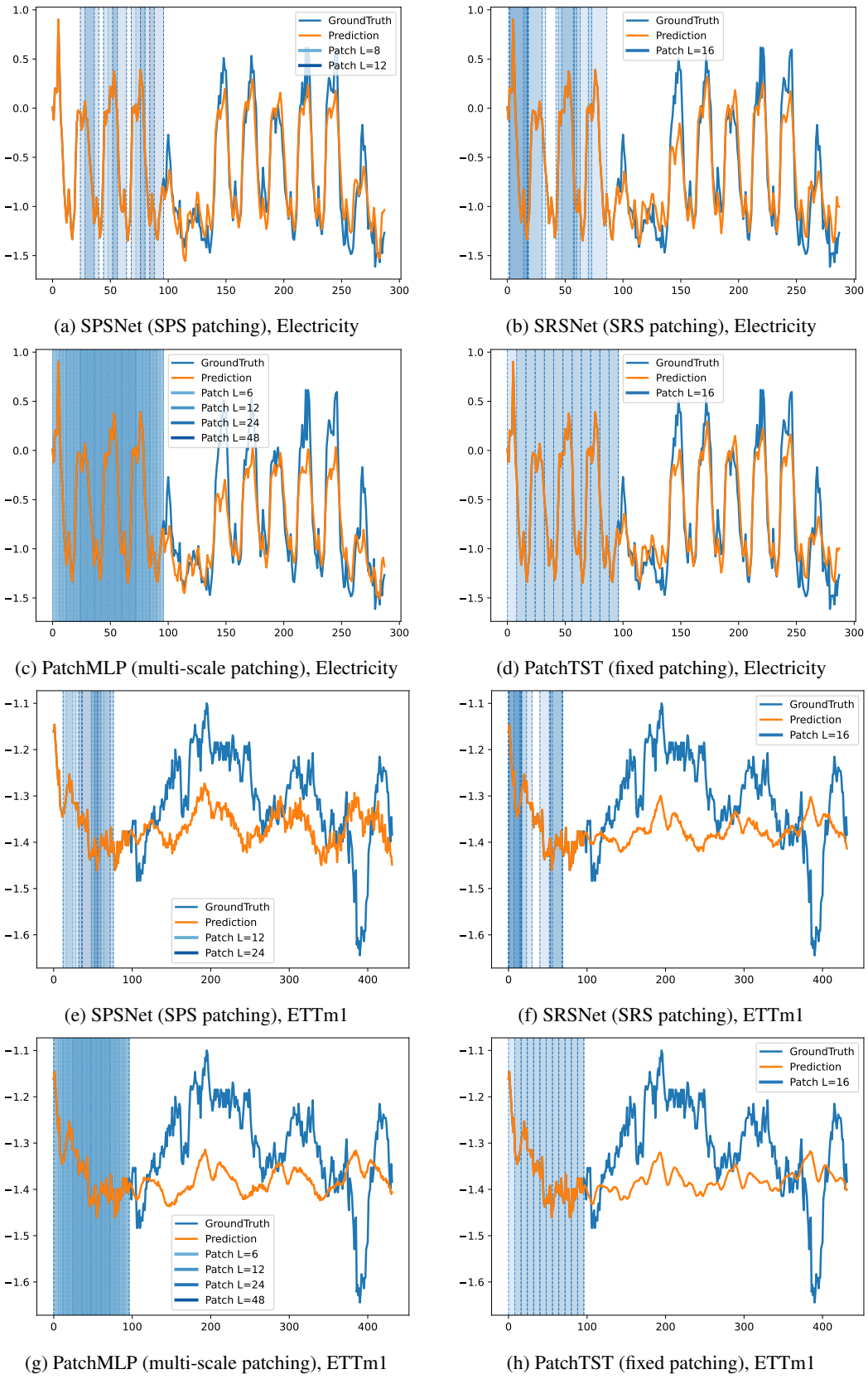


(g) PatchMLP (multi-scale patching), Solar



(h) PatchTST (fixed patching), Solar

Figure 6: Qualitative visualizations of SPSNet against other baselines.



(a) SPSNet (SPS patching), Electricity

(b) SRSNet (SRS patching), Electricity

(c) PatchMLP (multi-scale patching), Electricity

(d) PatchTST (fixed patching), Electricity

(e) SPSNet (SPS patching), ETTm1

(f) SRSNet (SRS patching), ETTm1

(g) PatchMLP (multi-scale patching), ETTm1

(h) PatchTST (fixed patching), ETTm1

Figure 7: Qualitative visualizations of SPSNet against other baselines.