energies

MDPI

Article

Partial Transfer Learning from Patch Transformer to Variate-Based Linear Forecasting Model

Le Hoang Anh [1,†], Dang Thanh Vu [2,†], Seungmin Oh [1], Gwang-Hyun Yu [1], Nguyen Bui Ngoc Han [3], Hyoung-Gook Kim [3], Jin-Sul Kim [1,*] and Jin-Young Kim [1,*]

[1] Department of Intelligent Electronics and Computer Engineering, Chonnam National University, Gwangju 61186, Republic of Korea; andrewlee1807@gmail.com (L.H.A.); osm5252kr@gmail.com (S.O.); sayney1004@gmail.com (G.-H.Y.)
[2] Research Center, AISeed Inc., Gwangju 61186, Republic of Korea; dtvu@aiseed.kr
[3] Department of Electronic Convergence Engineering, Kwangwoon University, Seoul 01897, Republic of Korea; nbngochan99@gmail.com (H.B.N.N.); hkim@kw.ac.kr (H.-G.K.)
* Correspondence: jsworld@jnu.ac.kr (J.-S.K.); beyondi@jnu.ac.kr (J.-Y.K.)
† These authors contributed equally to this work.

Abstract: Transformer-based time series forecasting models use patch tokens for temporal patterns and variate tokens to learn covariates' dependencies. While patch tokens inherently facilitate self-supervised learning, variate tokens are more suitable for linear forecasters as they help to mitigate distribution drift. However, the use of variate tokens prohibits masked model pretraining, as masking an entire series is absurd. To close this gap, we propose LSPatch-T (Long–Short Patch Transfer), a framework that transfers knowledge from short-length patch tokens into full-length variate tokens. A key implementation is that we selectively transfer a portion of the Transformer encoder to ensure the linear design of the downstream model. Additionally, we introduce a robust frequency loss to maintain consistency across different temporal ranges. The experimental results show that our approach outperforms Transformer-based baselines (Transformer, Informer, Crossformer, Autoformer, PatchTST, iTransformer) on three public datasets (ETT, Exchange, Weather), which is a promising step forward in generalizing time series forecasting models.

Keywords: multivariate time series forecasting; transfer learning; frequency analysis

Citation: Anh, L.H.; Vu, D.T.; Oh, S.; Yu, G.-H.; Han, N.B.N.; Kim, H.-G.; Kim, J.-S.; Kim, J.-Y. Partial Transfer Learning from Patch Transformer to Variate-Based Linear Forecasting Model. *Energies* **2024**, *17*, 6452. https://doi.org/10.3390/en17246452

Academic Editors: Riccardo Berta, Junfeng Liu, Luca Lazzaroni and Matteo Nardello

Received: 25 October 2024
Revised: 25 November 2024
Accepted: 19 December 2024
Published: 21 December 2024

1. Introduction

Time series data are created by performing sequential observations over time. However, a single time step in a time series lacks semantic meaning. Instead, temporal information—which encapsulates properties like continuity, periodicity, and trends—provides a deeper reflection of the underlying dynamics in time series. Nevertheless, real-world time series often involve intricate and overlapping temporal patterns, making it difficult to uncover the hidden dependencies and particularly challenging to model temporal variation.

While one-dimensional convolution kernels have been widely studied for their ability to capture temporal patterns in time series [1–4], they are inherently limited to modeling variations between adjacent time points. A common remedy is to expand the receptive field using global self-attention across temporal tokens, which has demonstrated notable improvements [5–7]. These temporal tokens, formed by multiple variates at the same timestamp, can be rolled into patch tokens by increasing the time dimension consecutively. However, while this method addresses the local constraints, it still struggles to maintain the multivariate correlations due to time-unaligned events. Recent research [8] takes an inverted approach by applying channel independence, introducing variate tokens that embed entire individual series into single tokens. This enables self-attention to effectively model covariates dependency. Figure 1 illustrates fundamental difference in constructing temporal, patch and variate tokens.

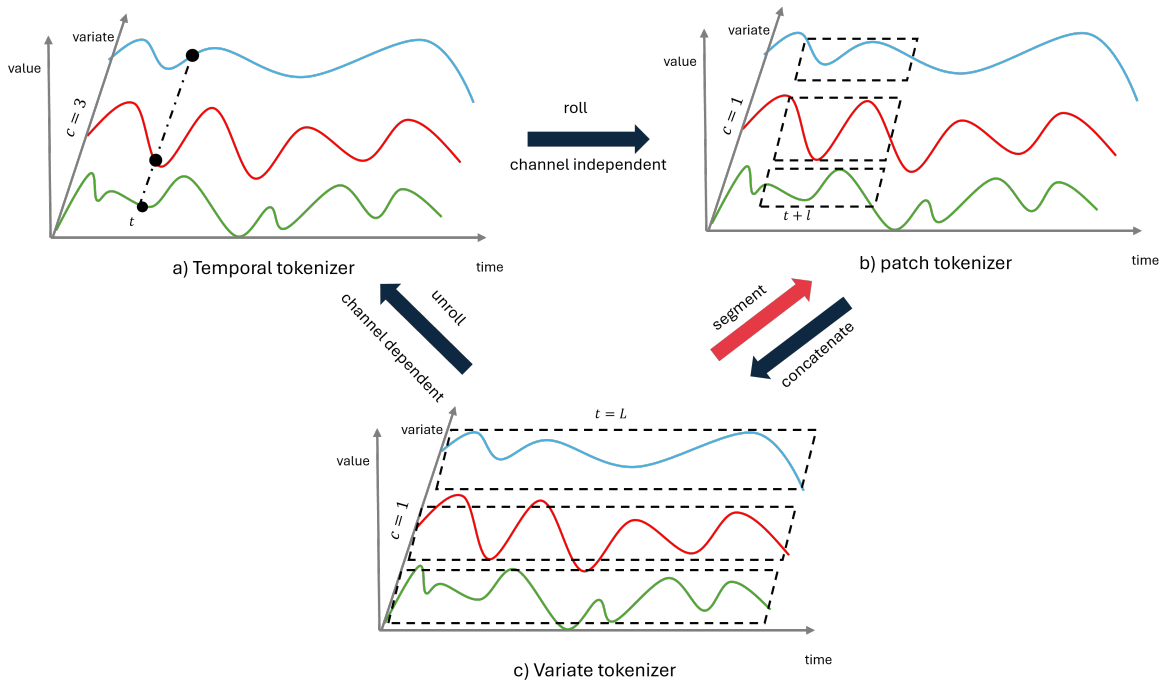


Figure 1. Tokenizations for multivariate time series (3 variables in red, green and blue) in different views: (**a**) temporal token, (**b**) variate token, and (**c**) patch token. One can obtain a new view from another by basic transformations such as roll/unroll, channel split/merge, and concatenate/segment.

Dependencies in time series arise either from data that are synchronously sampled at regular intervals or asynchronously sampled at varying points. The asynchronous nature of multivariate time series forecasting (MTSF) poses challenges due to potential delays and distinct physical measurements [9]. These issues make channel dependence less effective at capturing variate-centric representations. Therefore, modern MTSF models favor the channel independence approach [6,10], treating each multivariate time series as independent univariate series. This approach avoids the need to learn cross-variable relationships [8], allowing the model to focus solely on temporal information. This results in faster convergence and noise robustness, as noisy channels remain isolated in their own embeddings rather than contaminating others.

Furthermore, the success of recent time series forecasting models can be partly attributed to self-supervised learning, where a pretext task extracts high-level abstract representations and the learned weights will be transferred to downstream tasks [11–16]. Notably, PatchTST [6] has demonstrated that mask modeling on patch tokens significantly improves performance across various datasets. However, with variate tokens, direct application of mask signal modeling [17,18] is impractical, as masking an entire univariate sequence lacks meaning.

In this paper, we address this challenge by proposing LSPatch-T (Long–Short Patch Transferring). The novelty of this framework lies in its ability to enable a model that has been pretrained on short patch tokens to be transferable to full-length variate tokens for multivariate time series forecasting—an area that has not been explored previously. We contribute to the field in two aspects of modeling:

- **Partial transfer learning**: We examine the transferability of short patch tokens that generalize to longer lookback windows. In a trivial case, a whole series-variate token can be attained by concatenating all sub-series. These full-length variate tokens are then used for downstream forecasting, as they benefit from linear mapping, ensuring causality and mitigating distribution shifts between the historical and forecasting horizons. To achieve this, we transfer only a portion of the Transformer encoder to the downstream model, which is built of multi-layer perceptions (MLPs).

- **Spectrum analysis-based loss function**: In the frequency domain, a complex time series signal can be decomposed into distinct components. Low-frequency components capture slow-moving trends, while high-frequency components reflect rapid fluctuations or noise. We leverage frequency loss to ensure consistent representations between long and short patches, aligning with the low- and high-frequency bands.

2. Related Work

2.1. Transformers in Time Series Forecasting

Transformer [19] has gained attention for its outstanding performance in time series forecasting. Ongoing research in this area explores various innovations, such as developing new attention mechanisms and introducing intervention tokens to enhance model inputs. While recent work on MLP-based models has questioned the necessity of Transformers [20], they still excel in capturing long-range dependencies in sequential data, making them a robust choice for time series modeling. Additionally, a comprehensive theoretical study [21] demonstrated that Transformers are universal approximators of sequence-to-sequence functions, further justifying their use in time series forecasting beyond empirical results.

The point-wise dot-product self-attention in the vanilla Transformer suffers from quadratic complexity $O(L^2)$, limiting its ability to model long sequences. A number of efficient Transformer variants have been developed to reduce this complexity, often lowering it to $O(L \log L)$ by introducing sparsity biases [22–24] or approximating the self-attention matrix with a lower order [25,26]. LogSparse [22] addresses the locality-agnostic limitation by employing convolutional self-attention, where causal convolutions generate queries and keys, while Informer [26] introduces sparse attention by selecting dominant queries based on their compatible keys. Autoformer [24] employs auto-correlation-based attention to discover dependencies and aggregate representations at the sub-series level. FEDformer [25] applies attention in the frequency domain to analyze seasonal trend profiles in time series, achieving linear complexity by randomly selecting a fixed-size subset of frequencies. The success of Autoformer and FEDformer has sparked growing interest in exploring self-attention mechanisms in the frequency domain for time series modeling [27,28].

The aforementioned works primarily focus on univariate time series forecasting, which limits their applicability to real-world datasets that typically involve multivariate data. To address this, Crossformer [5] embeds both the time and variate dimensions into segments of a 2D array and introduces a two-stage attention mechanism to capture dependencies across both dimensions. Two close yet complementary models, PatchTST [6] and iTransformer [8], embody distinct approaches to multivariate time series forecasting—one based on patch tokens and the other on variate tokens. PatchTST uses a sub-series-level patch design, where time series are segmented into patches that serve as input tokens to the Transformer. In contrast, iTransformer treats each univariate series as a token, with self-attention capturing multivariate correlations and a feed-forward network learning global representations across series [29]. Both architectures emphasize the growing trend toward channel independence [10], where each channel represents a single univariate time series, sharing the same embedding across all series. Acknowledging the strengths of Transformer-based architectures in multivariate time series (MTS) forecasting—particularly in sub-series patching and channel independence—this work advances the field by bridging the design of patch tokens and variate tokens, proposing a transferring mechanism among them.

2.2. Learning Transferable Time Series Representation

Transfer learning in time series enables the learning of generalizable representations across multiple levels, including time embedding and structural latent spaces. Time2Vec [15] provides a model-agnostic vector representation for time embedding, effectively capturing both periodic and non-periodic patterns, while remaining invariant to time rescaling. Its simplicity allows it to seamlessly integrate with a wide range of models, making it a versatile tool in time series analysis.

In terms of learning structure, TS2Vec [16] employs contrastive learning in a hierarchical manner over augmented context views, distinguishing between positive and negative samples across both instance-wise and temporal dimensions. However, the augmentation strategies used in contrastive learning can introduce distortions to time series data. Additionally, forecasting can be seen as a form of denoising autoencoder, where future values are masked and then reconstructed. To leverage this concept, Ti-MAE [12] and PatchTST [6] adopt a strategy in which randomly masked sub-series are predicted during the upstream task. TempSSL [11] and SimMTM [30] combines contrastive learning on historical against target horizons and target mask modeling, allowing it to capture temporal dependencies while mitigating distribution shifts.

Universal time series forecasting poses a formidable challenge due to the intricate nature of time series signals, which are composed of multiple underlying distributions. TimeGPT-1 [31], a Transformer-based foundational model, has demonstrated potential for zero-shot transfer learning across a broad range of time series tasks. Likewise, Moment [32] provides an extensive repository of large-scale, pre-trained, open-source time series models, while TimesFM [13], a decoder-only Transformer, capitalizes on a vast corpus of time series data for pretraining. MOIRAI [33] advances the field by addressing the critical challenges of universal forecasting, such as accommodating multiple temporal frequencies and facilitating any-variate forecasting. It achieves this by incorporating multi-patch size inputs and outputs, enabling the model to adeptly navigate the heterogeneous nature of time series data.

Patch tokenization in PatchTST [6] is designed to be transferable by patchifying sub-series, enabling the application of Masked Signal Modeling. However, this approach does not accommodate the use of a linear head, which has proven to be effective in forecasting tasks. Conversely, the variate tokenization in iTransformer [8] supports a linear forecasting head but sacrifices transferability by disregarding patch context. To facilitate transfer learning with variate tokens, we introduce a dual-stream learning paradigm. The upstream phase utilizes masked modeling on sub-series to capture short-term dependencies, while the downstream phase focuses on forecasting using full-length time series data. These two streams are connected through a frequency-loss objective, ensuring coherence between short- and long-term dependencies during transfer learning. Frequency loss allows the model to handle temporal dynamics across a wide range of spectral and temporal scales.

3. LSPatch-T

This section conceptualizes our LSPatch-T, which works around two key ideas: (1) partial transfer learning and (2) spectrum analysis-based loss function. To formulate representation learning, we review the transferability and backbone network based on Transformer and MLP blocks. Then, we discuss our motivation and how to define an objective function with frequency loss. First of all, we describe our problem of interest, which is multivariate time series forecasting.

3.1. Problem Definition

A multivariate time series (MTS) is a multi-channel signal that is defined by a set of C covariates at a timestamp t, $\mathbf{x}_t = (x_1, ..., x_C)_t \in \mathbb{R}^C, C > 1$. Given a series of historical observations (lookback window) $\mathbf{X}_{t:t+L} = [\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, ..., \mathbf{x}_{t+L}] \in \mathbb{R}^{C \times L}$ with C channels and L time steps. The time series forecasting task (TSF) aims to predict the next S time horizon $\mathbf{Y}_{s+1:s+S} = [\mathbf{x}_{s+1}, \mathbf{x}_{s+2}, ..., \mathbf{x}_{s+S}] \in \mathbb{R}^{C \times S}, s = t + L$. In the formula, we need to learn a map $\mathcal{F}_\theta : \mathbf{X} \in \mathbb{R}^{C \times L} \rightarrow \mathbf{Y} \in \mathbb{R}^{C \times S}$ that predicts the multistep in future by maximizing the joint log-likelihood

$$\max_\theta \mathbb{E}_{(\mathbf{X},\mathbf{Y})} \log p(\mathbf{Y}|\mathcal{F}_\theta(\mathbf{X})) \tag{1}$$

Although most studies assume that $\mathbf{x}_t$ represents covariates recorded simultaneously at time step t, real-world datasets often exhibit time lags among variates. In the above problem, a multi-step predictive distribution can be obtained by rolling one-step predictions.

Note that, in practice, it may be more effective to focus on modeling specific values, such as summary statistics, rather than the entire probability distribution. Models that focus on these specific values are referred to as point-forecasters [34].

Instead of training the predictive model $\mathcal{F}_\theta$ from scratch in a supervised manner, we adopt a self-supervised learning method that provides a robust initialization for θ. Subsequently, θ is fine-tuned on the same dataset to optimize the performance of the forecasting task.

3.2. Partial Transfer Learning

Studies [6,12,13] have demonstrated that mask modeling methods learn representations that can be transferred to downstream forecasting tasks. Mask modeling, in short, masks a portion of patches randomly during training and learns each token's representation via predicting the values of these masks. Masking is crucial for learning abstract representation because it conceptualizes locality while avoiding trivial inferences, such as merely interpolating between neighboring time values. This approach encourages a high-level understanding of the entire sequence, rather than relying solely on adjacent data points.

The core of this study lies in transferring the weights from a pretrained model, which was trained using short-length patch tokens, to a downstream task that operates on full-length variate tokens. This extension is made possible through the shared-weight mechanism. Specifically, during pretraining, trainable weights are shared both across tokens and among covariates. Also, feed-forward layers in Transformer's blocks work as memory cells [35] that implicitly store the correlations between covariates, allowing latent features to be transferred to the downstream task.

Simply, a set of patch tokens with identical length l can be concatenated to construct a longer patch, as illustrated in Figure 2. Formally, given a subseries $\mathbf{x}_{t+kl:t+(k+1)l}$ of window size l starting at time t, and stretching from frame $t+kl$ to frame $t+(k+1)l$, the consecutive concatenation subseries $\mathbf{x}_{t:t+nl}$ that starts at time t and stretches out n frames of same window size l is defined as follows:

$$\mathbf{x}_{t:t+nl} = \begin{bmatrix} \mathbf{x}_{t:t+l} & \mathbf{x}_{t+l:t+2l} & \cdots & \mathbf{x}_{t+(n-1)l:t+nl} \end{bmatrix} \tag{2}$$

This study considers the extreme case, $n = L/l$, where short-length patch tokens are concatenated into a full-length variate token, meaning all patches in a univariate series are concatenated (the case on top of Figure 2). Furthermore, both patch tokens and variate tokens are designed following the channel independence principle, ensuring that temporal dependencies are preserved within a single univariate series through the attention mechanism. In the pretext task, the backbone network is the Transformer Encoder. However, since we concatenate n patch tokens into a single variate token, using self-attention here becomes awkward. To address this, we replace the Transformer blocks with MLP blocks. This substitution aligns with our design, as we only transfer the value-embedding matrix (excluding the key and query embeddings) to serve as the weight for the linear layer in the MLP block.

Figure 3 describes our LSPatch-T self-supervised framework, operating through two phases: pretraining with mask signal modeling (left) and the downstream forecasting task (right). In the pretraining phase, each univariate signal from the multivariate time series is segmented into patches, with a subset of these patches masked to generate visible and masked tokens. These tokens are processed through a Vanilla Transformer Encoder backbone to learn hidden representations, and a decoder head is used to reconstruct the masked patches. Each block in the Transformer Encoder consists of three main components: a self-attention module, a feed-forward network, and residual connections. This phase focuses on capturing local dependencies within signals, leveraging time loss to align temporal relationships and frequency loss to preserve spectral coherence. In the downstream phase, the patching operation is replaced by inverted variate tokens, allowing the model to handle full-length multivariate signals for forecasting tasks. A newly introduced embedding

layer processes these tokens, paired with a forecasting-specific prediction head. While the backbone architecture remains largely consistent with the Transformer Encoder to ensure transferability, the self-attention mechanism is replaced by a lightweight MLP layer for improved computational efficiency in long-term forecasting. Notably, the MLP layer is initialized with only the weights of the *value matrix* learned during the pretraining phase, which we conceptualize as partial transfer learning. Similar to pretraining, the downstream phase also employs time loss and frequency loss. This dual-phase design enables LSPatch-T to transition from learning localized short-term patterns to global long-term dependencies in forecasting.

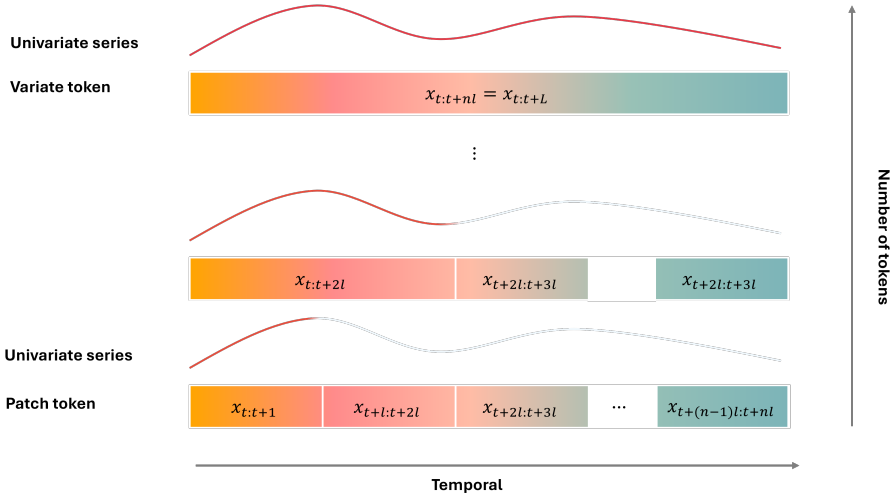


Figure 2. A full-length variate token is obtained by concatenating short-length patch tokens.

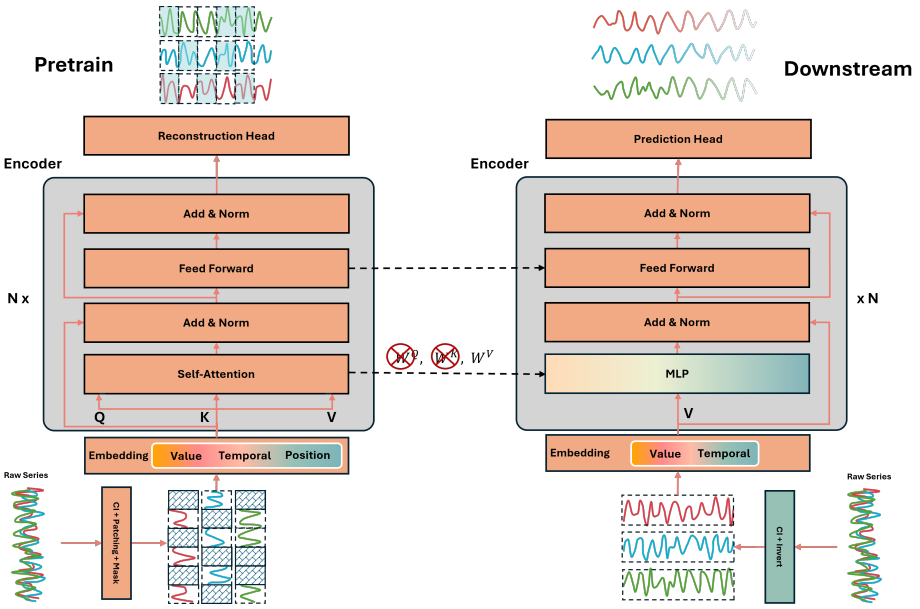


Figure 3. LSPatch-T self-supervised framework: LSPatch-T follows a two-phase approach. (1) In the pretraining phase, each univariate signal undergoes patching and masking to create visible and masked tokens. These tokens are then passed through the Transformer Encoder backbone, with a reconstruction head on top to predict the masked patches. (2) In the downstream phase, patching is replaced by inverted variate tokens, and a different embedding layer and prediction head are used for forecasting. The backbone network in the downstream phase is kept as close as possible to the Transformer encoder, with only the self-attention mechanism replaced by an MLP layer. Both the pretraining and downstream phases adopt channel independence as well as time loss and frequency loss.

In the pretext task, the use of patch tokens is essential, as the pretraining focuses on reconstructing masked information based on the visible portions, effectively capturing temporal dependencies [12,30]. However, in the downstream task, the focus shifts to predicting future values based on historical data, and when performing this task, causality and temporal continuity become more crucial [4,34,36–39]. During weight transfer, we replace both the initial embedding layer and the final projection layer. This adjustment ensures the transition between patch tokens and variate tokens takes place without violating the embedding size. The detailed implementation of the embedding process is provided in the following section.

3.3. Components

The multivariate time series is split into C univariate series, each of which is fed independently into the Transformer backbone, adhering to the channel independence setting. To prevent distribution shift, we apply Reversible Instance Normalization [40] to normalize each univariate series instance before processing.

Our framework consists of four main components: (1) patching, (2) token embedding, (3) a backbone model, and (4) a projection head. Both the pretraining and downstream phases utilize these components, but there are key differences in the configurations at certain stages, which will be highlighted below.

Patching: To put it simply, patching involves splitting a time series into several sub-series. A patch can be understood as a segment of length l from the original time series signal, $\mathbf{x}_{t:t+l}$. In time series signaling, this is equivalent to short-time analysis or the window method. The benefits of patching are twofold: first, it is compatible with modern Transformer-based architectures, which excel at learning global dependencies; second, it reduces the computational complexity of algorithms that operate on single timestamps. Furthermore, the inverse process of patching is concatenation, which is used to aggregate patch tokens into a variate token during downstream tasks.

Token embedding: Tokenization is the process of calculating token embedding of each patch, resulting in what is called a patch token or token embedding. In upstream phase, each patch is embedded using a linear projection added to a learnable temporal positional embedding to maintain its temporal order, represented as $\mathbf{z}_n = \mathbf{W}_{\text{emb}}\mathbf{x}_{t+nl:t+(n+1)l} + \mathbf{W}_{\text{pos}} \in \mathbb{R}^D$, where $\mathbf{W}_{\text{emb}} \in \mathbb{R}^{D\times l}, \mathbf{W}_{\text{pos}} \in \mathbb{R}^D$ and $n = 0...\lfloor L/l \rfloor$. However, for the downstream task, a variate token at channel c naturally preserves their order through the linear projection, $\mathbf{z} = \mathbf{W}_{\text{emb}}\mathbf{x}_{t:t+L} \in \mathbb{R}^D$, where $\mathbf{W}_{\text{emb}} \in \mathbf{R}^{D\times L}$ represents the neuron permutation [20]. Therefore, positional encoding is not required in this case. There are two key configurations we would like to emphasize. First, the embedding layer, $\mathbf{W}_{\text{emb}}$, is shared both across patches and channels. Second, we do not transfer this embedding layer from the upstream to the downstream task due to the difference in input dimensions, where $l < L$. Rather, we initialize new weights to embed the entire lookback window in the downstream stage.

Vanilla Transformer: Our chosen backbone network for pretraining is the Vanilla Transformer Encoder, which combines the two baselines, PatchTST and iTransformer [6,8]. The propagation at each Transformer block includes Multi-head Self-Attention [19] followed by linear projection and batch normalization. The attention of each head is calculated as follows:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) = \text{softmax}(\frac{\mathbf{z}_n^\top \mathbf{W}_i^q (\mathbf{z}_n^\top \mathbf{W}_i^k)^\top}{\sqrt{d}})\mathbf{z}_n^\top \mathbf{W}_i^v \tag{3}$$

where $\mathbf{W}_i^q, \mathbf{W}_i^k, \mathbf{W}_i^v \in \mathbb{R}^{D\times D}$ are projection parameters, respectively, for query, key, and value at i-th head, and $\mathbf{z}_n \in \mathbb{R}^D$ is a token embedding. The correlation among patch tokens is retained in these projection matrices through the attention mechanism, which is a crucial factor in making Transformers a standard choice for pre-trained models. However, the switch to variate tokens has an interesting effect: the number of tokens reduces to just one, meaning the operation only needs to calculate the attention score

for itself and naturally becomes an MLP layer [41,42]. Therefore, here, we initialize the MLP's parameters using the embeddings of values ($\mathbf{W}^v$) learned in the pretraining phase. Given a variate token $\mathbf{z}$, its embedding is calculated for each MLP block as follows: $\mathbf{z}^{(\text{out})} = \mathbf{W}_{\text{FF}}(\mathbf{z} + \text{Layernorm}(\mathbf{z}^\top \mathbf{W}^v))$, where $\mathbf{W}_{\text{FF}}$ have also been transferred from the pretrained model.

Projection head: The projection head sits atop the backbone model, mapping the output tokens to the final predictions. During the pretraining phase, we predict only the values of the invisible tokens that have been masked by indexing them with 0, following the Masked Image Modeling (MIM) strategy [17]. Here, the focus is on predicting the missing patches rather than reconstructing the entire sequence [18]. Prediction for a patch token, $\mathbf{z}_n$, is calculated as $\mathbf{o}_{t+nl:t+(n+1)l} = MLP(\mathbf{z}_n) \in \mathbb{R}^l$. On the other hand, our goal in the downstream task is to achieve predictions for the entire forecasting horizon for each variate. Therefore, the formula would be $\mathbf{o}_{s+1:s+S} = MLP(\mathbf{z}) \in \mathbb{R}^S, s = t + L$. In this way, the downstream model can inherit the advantage of learning seasonal components from linear mapping [29,41].

3.4. Robust Frequency Loss

We apply a standard time loss along with our proposed frequency loss in both the pretraining and fine-tuning phases.

Time loss: We employ the Mean Squared Error (MSE) to quantify the discrepancy between the predictions and the ground truth. The loss for each channel is computed, then averaged across C time series to determine the time-domain loss.

$$\mathcal{L}_{\text{time}}(\hat{\mathbf{y}}, \mathbf{y}) = \mathbb{E}_{\mathbf{x}}[\frac{1}{C}\sum_{c}^{C} ||\hat{\mathbf{y}}^{(c)} - \mathbf{y}^{(c)}||^2] \tag{4}$$

where $\mathbf{y} = \mathbf{x}_{s+1:s+S}$ and $\hat{\mathbf{y}}$ is its prediction over horizon $[t + L + 1, t + L + S]$

Frequency loss: Time series signals are often non-stationary processes, entangled by trend, seasonality (low-frequency), and noise (high-frequency) components. Several prior studies have incorporated frequency information into time series forecasting [24,25,27]. Unlike approaches such as TimesNet [28] and FEDformer [25], which focus on selecting dominant frequencies—primarily from the low-frequency band—while overlooking the high-frequency regions, our approach balances the contributions from both frequency bands. This ensures that low- and high-frequency components are given equal consideration, allowing us a broader range of temporal patterns to be captured.

$$\mathcal{L}_{\text{freq}}(\hat{\mathbf{y}}, \mathbf{y}) = \mathbb{E}_{\mathbf{x}}[\frac{1}{C}\sum_{c}^{C} \tanh(||\hat{\mathcal{Y}}^{(c)} - \mathcal{Y}^{(c)}||^2)] \tag{5}$$

where $\mathcal{Y}$ is the spectral magnitude of series $\mathbf{x}_{s+1:s+S}$ from Fourier Transform, and $\hat{\mathcal{Y}}$ is its predicted series counterparts. This study applies tanh as a scaling function, which has been investigated in [43]. Other scaling functions have been discussed in [44].

The work [33] suggests a strategy that captures varied frequency bands by using larger patch sizes for high-frequency data and vice versa. This aligns with our approach, where in the upstream phase, we use smaller patch sizes to capture low-frequency patterns. In the downstream task, we extend this to larger patch sizes to accommodate the broader context of high-frequency data. Throughout both phases, frequency loss is employed to ensure consistency between the frequency components of the ground truth and predictions.

The total training objective is the weighted sum of time loss and frequency loss

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{freq}} + (1 - \alpha)\mathcal{L}_{\text{time}} \tag{6}$$

where α is a hyperparameter that is used to control the contribution of each loss.

4. Experiments

This section begins with a brief description of the datasets used, followed by the presentation of the main experimental results and ablation studies. Finally, key discussions and insights are provided.

4.1. Dataset

We conducted experiments on six popular multivariate datasets:

- **Weather dataset** (github.com/zhouhaoyi/ETDataset, accessed on May 2024): The dataset contains 21 meteorological indicators collected every 10 min from the Weather Station of the Max Planck Biogeochemistry Institute in Germany during 2020.
- **Exchange dataset** (github.com/laiguokun/multivariate-time-series-data/tree/master/exchange_rate, accessed on May 2024): The dataset includes daily exchange rates from eight countries, spanning from 1990 to 2016.
- **ETT (Electricity transformer temperature) dataset** (www.bgc-jena.mpg.de/wetter/, accessed on May 2024): The dataset consists of seven factors related to electricity transformers, recorded from July 2016 to July 2018. It has four subsets: ETTh1/ETTh2, recorded hourly, and ETTm1/ETTm2, recorded every 15 min.

We followed the iTransformer settings for splitting the training, validation, and test sets to conduct the experiments and comparison. The details of the dataset are presented in Table 1. In the pretraining phase, we used only the training set for model training, selecting the best model for the validation but setting aside the test data. In the downstream phase, both the training and validation sets were used for model training, and the final results were recorded in the test set.

Table 1. Summary of datasets.

Dataset	Features	Time Steps	Frequency	Information
ETTh1	7	17,420	Hourly	Electronic
ETTh2	7	17,420	Hour	Electronic
ETTm1	7	69,680	15 min	Electronic
ETTm2	7	69,680	15 min	Electronic
Weather	21	52,696	10 min	Weather
Exchange	8	7,588	Daily	Economy

4.2. Settings

All experiments were conducted using the PyTorch framework on an NVIDIA H100 GPU. Here, we list common hyper-parameter settings for training. The ADAM optimizer was employed with initial learning rates of $10^{-3}, 5 \times 10^{-4}$ and 10^{-4}. A batch size of 32 was used, with the number of training epochs fixed at 10. The dimensionality of the series representation was set to 512. RevIN normalization was applied to avoid a distribution shift between training and test data [40]. Additional configurations for the pretraining phase and downstream are provided in Table 2. We make the source code available at https://github.com/synapsespectrum/LSPatch-T for reproducibility.

Table 2. Summary of configurations for pretraining and downstream.

Settings	Pretraining	Downstream
Task	Mask modeling	Forecasting
Patch size (l)	12	L
Masking ratio	40% (same as PatchTST)	no
Causal	no	yes
Batch size	64	32
Backbone	Transformer Encoder	Multi-MLP

Current literature in the field of time series forecasting commonly adopts a set of hyperparameters for the lookback window size and forecasting horizon size, which are {24, 48, 96, 168, 192, 336, 720}. In accordance with this convention, our paper applied the same settings to ensure fair comparisons. The choice of these values is closely tied to the characteristics of datasets recorded on an hourly basis. For example, 24 h corresponds to one day, 168 h to one week, and 720 h to approximately one month. Table 3 provides details of our data split, where the division is performed chronologically, progressing from past to future time steps.

Table 3. Dataset splits.

Dataset	Training	Validation	Testing
ETTh1 & ETTh2	12 months (8545)	4 months (2881)	4 months (2881)
ETTm1 & ETTm2	12 months (34465)	4 months (11521)	4 months (11521)
Weather	8.5 months (36792)	1.2 months (5271)	2.4 months (10540)
Exchange	420 months (5120)	54 months (665)	117 months (1422)

For comparison, we chose the following baselines: Vanilla Transformer [19], Informer [26], Autoformer [24], Crossformer [5], PatchTST [6], and iTransformer [8]. The results were reproduced in our own environment using the official implementations for each baseline. The architecture for each model was kept as the default unless otherwise specified. The conventional Transformer model is not built for time series forecasting tasks, so we used the version that had been studied in Crossformer. The Mean Squared Error (MSE) and Mean Absolute Error (MAE) were utilized as evaluation metrics. All experiments were repeated twice, and the best results for each metric are reported.

4.3. Main Results

Table 4 demonstrates that LSPatch-T and PatchTST are the two leading models, with LSPatch-T achieving the best performance (in both metrics) in most cases, or ranking second otherwise (*MSE: 2.11%, MAE: 1.46% improvement over PatchTST*). Three key observations emerge from these results. First, the superiority of channel independence in MTSF is reaffirmed, as PatchTST, iTransformer and LSPatch-T significantly outperform models like Crossformer (*MSE: 61.95%, MAE: 41.99%*), Autoformer (*MSE: 41.25%, MAE: 28.82%*), Informer (*MSE: 78.07%, MAE: 67.31%*), and Transformer (*MSE: 78.62%, MAE: 59.65%*), which rely on channel dependence. Second, LSPatch-T excels in long-term predictions related to the ETTh and Weather datasets, particularly in the longer forecasting horizons of 192, 336, and 720 time steps. Third, although LSPatch-T shares a similar design with iTransformer—using variate tokens and a linear layer as the forecaster—our model surpasses iTransformer (*MSE: 3.32%, MAE: 2.60% improvement*), highlighting the effectiveness of transfer learning where weights are initialized from a well-trained plateau. The results are notable because we performed unsupervised pretraining and supervised forecasting on the same dataset without additional samples, yet the two-phase training approach yielded better results than using supervised forecasting alone. We also note that in our reproduction of the experiments, iTransformer did not outperform PatchTST, as claimed in the original work.

We also note that LSPatch-T required a relatively smaller number of computations (measured in FLOPs) compared to other baselines. As shown in Figure 4, LSPatch-T and iTransformer are the two lightest models, with FLOPs of 2.87B and 2.26B, respectively, while Crossformer and Vanilla Transformer are the two heaviest models. Although we adopted the same pretraining strategy as PatchTST, in the downstream phase, we replaced the self-attention module with a linear layer, which led to a significant reduction in overall computation—approximately 77.67%.

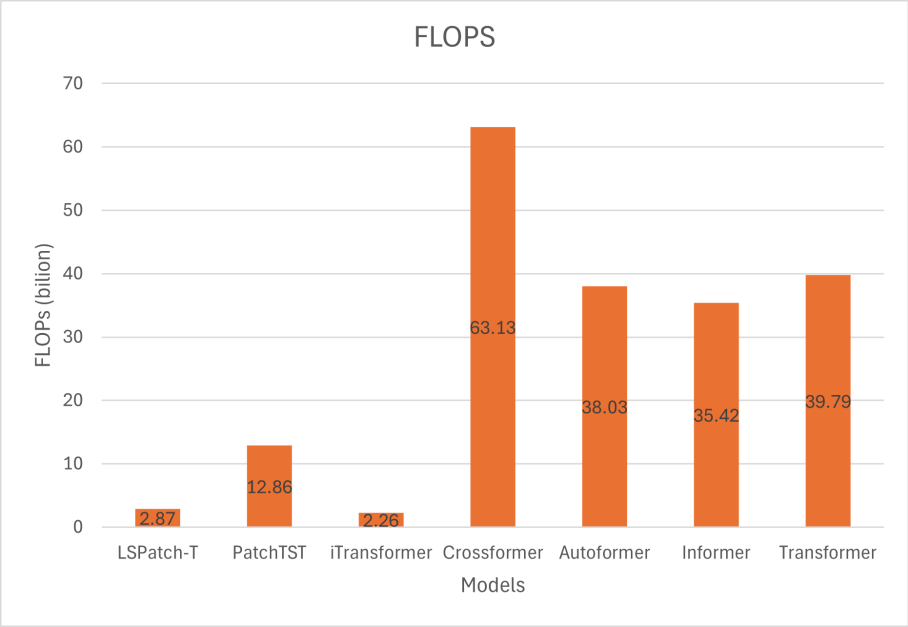


Figure 4. Number of computations measured in FLOPs of LSPatch-T compared to other baselines.

Table 4. Comparison of downstream LSPatch-T against other baselines across Weather, Exchange, and ETT datasets. The lookback window size is fixed at 96, and the forecasting horizon is {24, 48, 96, 168, 192, 336, 720}. Avg means averaged by prediction lengths. The best performance is reported in **bold**, and the second best is underlined.

Model		**LSPatch-T**		**PatchTST**		**iTransformer**		**Crossformer**		**Autoformer**		**Informer**		**Transformer**	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	24	0.105	0.139	**0.104**	0.138	0.106	**0.137**	**0.097**	0.153	0.422	0.407	0.164	0.247	0.262	0.338
	48	0.139	0.181	0.140	**0.183**	0.138	**0.177**	**0.132**	0.209	0.913	0.661	0.249	0.335	0.528	0.514
	96	**0.173**	**0.218**	0.178	0.223	0.179	**0.220**	0.163	0.241	0.484	0.488	0.283	0.354	0.762	0.632
	168	**0.208**	**0.250**	0.213	0.254	0.219	0.257	0.216	0.292	0.812	0.619	0.294	0.354	1.117	0.749
	192	**0.225**	**0.264**	0.227	0.266	0.233	0.268	**0.219**	0.295	1.088	0.736	0.302	0.358	1.302	0.812
	336	**0.284**	**0.306**	0.286	0.308	0.293	0.312	0.292	0.349	1.614	0.911	0.360	0.395	1.495	0.897
	720	**0.366**	**0.358**	0.371	0.360	0.379	0.366	0.423	0.419	1.689	0.968	0.435	0.437	1.789	0.996
	Avg	**0.214**	**0.245**	0.217	0.247	0.221	0.248	0.220	0.280	1.003	0.684	0.298	0.354	1.036	0.705
Exchange	24	0.032	0.119	0.032	**0.118**	0.033	0.122	0.380	0.377	0.070	0.194	1.288	0.856	0.500	0.528
	48	0.057	0.163	**0.055**	**0.161**	0.057	0.166	0.535	0.469	0.133	0.267	1.755	1.032	0.892	0.709
	96	**0.110**	**0.228**	0.112	0.231	0.119	0.244	0.959	0.672	0.187	0.311	1.982	1.076	1.695	0.954
	168	**0.198**	**0.312**	0.200	0.315	0.200	0.318	1.192	0.775	0.313	0.404	2.666	1.249	2.639	1.221
	192	**0.225**	**0.335**	0.228	0.338	0.231	0.343	1.293	0.784	0.356	0.429	2.944	1.304	2.996	1.297
	336	0.419	0.466	**0.416**	**0.463**	0.422	0.470	2.166	1.094	0.538	0.537	3.842	1.491	3.138	1.236
	720	1.112	0.794	1.092	0.785	**1.027**	**0.768**	2.057	1.119	1.285	0.876	4.162	1.669	3.190	1.367
	Avg	0.308	0.345	0.305	**0.344**	**0.298**	0.347	1.226	0.756	0.412	0.431	2.663	1.240	2.150	1.045
ETTh1	24	**0.290**	**0.344**	0.293	0.348	0.312	0.365	0.323	0.387	0.396	0.432	0.396	0.432	0.616	0.581
	48	**0.328**	**0.367**	0.339	0.375	0.344	0.384	0.358	0.399	0.477	0.456	0.477	0.456	0.799	0.693
	96	**0.371**	**0.392**	0.386	0.401	0.390	0.406	0.406	0.427	0.467	0.468	0.467	0.468	0.874	0.717
	168	**0.408**	**0.414**	0.423	0.428	0.429	0.429	0.471	0.474	0.535	0.489	0.535	0.489	1.105	0.852
	192	**0.421**	**0.421**	0.436	0.434	0.441	0.435	0.503	0.505	0.565	0.513	0.565	0.513	0.866	0.700
	336	**0.461**	**0.443**	0.509	0.467	0.476	0.452	0.647	0.596	0.488	0.480	0.488	0.480	1.185	0.876
	720	**0.478**	**0.475**	0.575	0.510	0.491	0.483	0.748	0.664	0.571	0.541	0.571	0.541	1.171	0.860
	Avg	**0.394**	**0.408**	0.423	0.423	0.412	0.422	0.494	0.493	0.500	0.483	0.500	0.483	0.945	0.754

Table 4. *Cont.*

Model		LSPatch-T		PatchTST		iTransformer		Crossformer		Autoformer		Informer		Transformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	24	**0.172**	**0.261**	0.175	0.266	0.185	0.269	0.239	0.331	0.301	0.373	0.805	0.720	0.865	0.723
	48	0.231	0.302	**0.227**	**0.299**	0.238	0.311	0.765	0.598	0.300	0.365	1.433	0.946	1.168	0.878
	96	**0.288**	**0.339**	0.290	0.342	0.297	0.345	0.814	0.705	0.377	0.411	2.433	1.235	1.842	1.114
	168	0.357	0.380	**0.347**	**0.377**	0.358	0.384	1.517	0.875	0.419	0.432	5.786	1.981	4.165	1.607
	192	**0.368**	**0.391**	0.371	0.394	0.375	0.397	1.769	1.075	0.442	0.443	4.498	1.731	5.909	1.976
	336	**0.410**	**0.421**	0.411	0.427	0.433	0.435	2.652	1.379	0.483	0.484	3.907	1.634	3.674	1.544
	720	**0.425**	**0.442**	0.435	0.451	0.458	0.459	2.780	1.427	0.462	0.477	3.455	1.585	2.320	1.289
	Avg	**0.322**	**0.362**	0.322	0.365	0.335	0.371	1.505	0.913	0.398	0.426	3.188	1.405	2.849	1.304
ETTm1	24	**0.211**	**0.282**	0.214	0.286	0.225	0.296	0.229	0.314	0.448	0.453	0.321	0.368	0.367	0.407
	48	**0.284**	**0.332**	0.298	0.346	0.294	0.342	0.315	0.364	0.433	0.449	0.475	0.456	0.386	0.414
	96	**0.321**	**0.358**	0.334	0.373	0.349	0.381	0.369	0.420	0.443	0.457	0.592	0.554	0.536	0.523
	168	**0.358**	**0.379**	0.367	0.389	0.374	0.392	0.431	0.450	0.517	0.484	0.684	0.608	0.626	0.579
	192	**0.364**	**0.382**	0.371	0.390	0.381	0.395	0.475	0.501	0.518	0.483	0.734	0.620	0.670	0.596
	336	**0.395**	**0.405**	0.403	0.408	0.438	0.429	0.609	0.578	0.547	0.503	0.934	0.739	0.850	0.706
	720	**0.457**	**0.442**	0.458	0.446	0.505	0.466	0.845	0.709	0.501	0.491	1.037	0.778	1.047	0.781
	Avg	**0.341**	**0.369**	0.349	0.377	0.367	0.386	0.468	0.477	0.487	0.474	0.682	0.589	0.640	0.572
ETTm2	24	0.101	0.194	**0.100**	**0.193**	0.104	0.201	0.135	0.248	0.138	0.250	0.185	0.315	0.137	0.263
	48	0.137	0.229	**0.135**	0.229	0.138	0.231	0.228	0.329	0.192	0.291	0.263	0.373	0.246	0.381
	96	**0.176**	0.258	0.177	**0.257**	0.183	0.266	0.314	0.395	0.236	0.321	0.385	0.488	0.398	0.458
	168	**0.230**	**0.293**	0.234	0.297	0.240	0.302	0.442	0.467	0.289	0.346	0.552	0.577	0.515	0.536
	192	**0.240**	**0.299**	0.241	0.302	0.253	0.313	0.633	0.579	0.275	0.332	0.636	0.617	0.750	0.644
	336	**0.299**	**0.337**	0.305	0.343	0.311	0.347	0.974	0.691	0.330	0.367	1.206	0.839	1.251	0.841
	720	**0.397**	**0.395**	0.406	0.403	0.406	0.403	3.100	1.179	0.428	0.423	3.093	1.339	2.470	1.183
	Avg	**0.226**	**0.286**	0.228	0.289	0.234	0.295	0.832	0.555	0.270	0.333	0.903	0.650	0.824	0.615

4.4. Effectiveness of Lookback Window

Although it is used as a default setting in various baselines, the choice of a fixed lookback window size of 96 (as selected in the abovementioned experiments) does not consistently yield better performance across all datasets. To clarify this point, we evaluate the robustness of our model with respect to lookback window size, as shown in Figure 5. LSPatch-T is compared in the MSE metric against PatchTST, iTransformer, and Crossformer on the ETTh1 and Weather datasets. The results demonstrate that LSPatch-T achieves the lowest MSE across all lookback windows for both datasets. Notably, among the four baselines, only LSPatch-T strictly follows the principle that a longer input horizon leads to better performance, benefiting from the increased temporal information. It is also important to highlight that there is a loss plateau starting at a lookback window of 192, where further increasing the window yields' diminishing returns. This comparison provides insights into how each model handles long-term dependencies in time series .

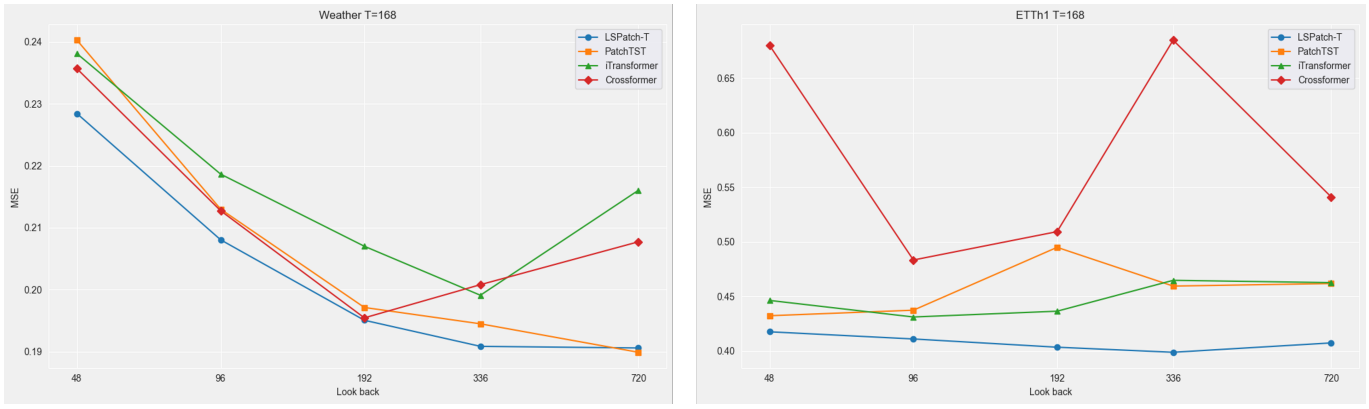
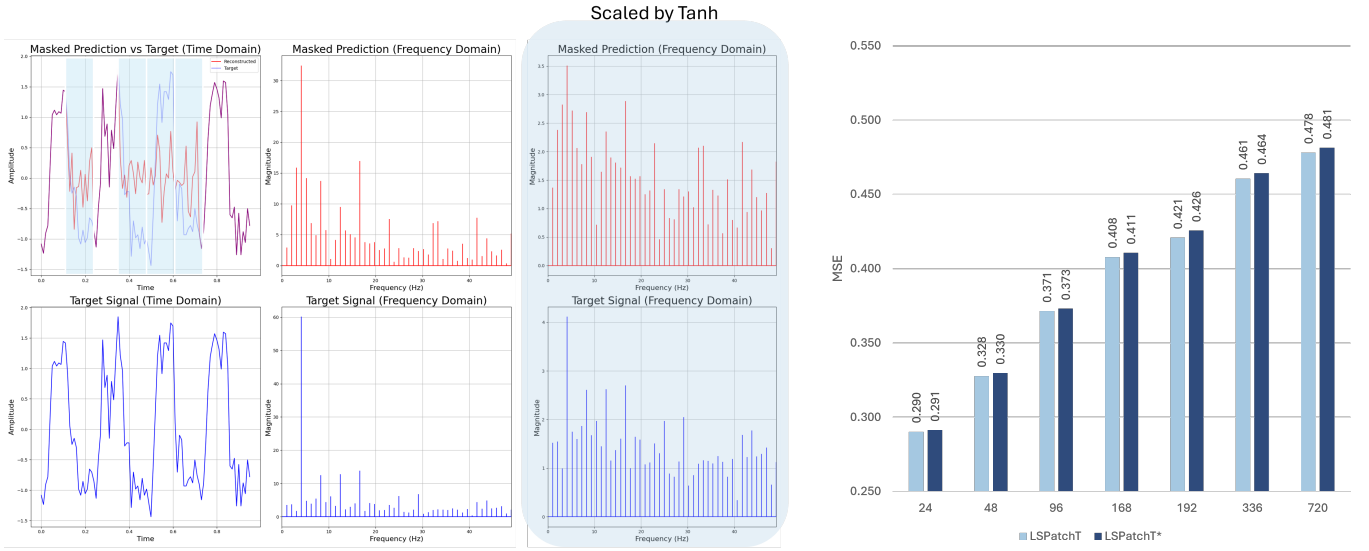


Figure 5. Impact of input horizon on forecasting performances. The forecasting horizon is fixed at 168 and the lookback window is set to {48, 96, 192, 336, 720}.

4.5. Effectiveness of Frequency Loss

We study the effectiveness of frequency loss in Figure 6. Although the forecasting performance shows only a slight improvement (*0.66% on average*) compared to using time loss alone, the improvement is consistent across all forecasting horizons, as seen in Figure 6b. This offers the promising benefit of incorporating frequency analysis when dealing with time series data. Additionally, Figure 6a illustrates the advantages of scaling in the frequency domain. From an optimization perspective, the L_2 objective tends to smooth the regression by averaging across the frequency spectrum, giving dominant frequencies more weight while reducing the contribution of others. Scaling acts as an equalizer, suppressing dominant frequencies and amplifying the less prominent ones, thereby encouraging finer reconstruction. While this method has been explored to some extent in computer vision [43,44], its application in time series analysis is relatively new [25,28], and important frequencies are often selected heuristically. This study takes a step further by balancing the contributions of all frequencies, rather than selecting specific ones.



a) Time series in frequency domain and its scaling b) Forecasting performance

Figure 6. Impact of frequency loss function. (**a**) Frequency domain analysis. (**b**) Forecasting performances on ETTh1 dataset, the LSPatch-T* is a model that can be trained only with time loss.

4.6. Cross-Data Transferability

We conducted pretraining on one dataset and fine-tuned the model on another to assess the performance of LSPatch-T in cross-dataset forecasting tasks. As shown in Table 5, ETTh2 → ETTh1 denotes pretraining on ETTh2 and transferring to ETTh1. As anticipated, cross-domain forecasting results are slightly lower than in-domain predictions. In particular, ETTh2 → ETTh1 achieves a *0.77%* improvement in MSE and *0.86%* in MAE, while Weather → ETTh1 and Exchange → ETTh1 show improvements of *0.23%*(MSE), *0.42%* (MAE) and *0.50%* (MSE), *0.75%* (MAE), respectively. Notably, the improvements are more significant when forecasting longer horizons, as demonstrated in the case with a horizon of 720. Nonetheless, we observed a slightly inferior performance on shorter horizons, which suggests that concatenating more patch tokens into a variate token enhances transferability.

Table 5. Transferability across the data domain (→). The improvement is marked with ↑, while the decline is marked with ↓.

Model			Transferred LSPatch-T		LSPatch-T		Performance	
Source	**Target**	**Horizon**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
In-domain	ETTh2 ↓ ETTh1	96	0.374	0.392	0.371	0.392	↓0.73%	↓0.02%
		192	0.426	0.421	0.421	0.421	↓1.26%	↓0.12%
		336	0.463	0.440	0.461	0.443	↓0.47%	↑0.62%
		720	0.455	0.462	0.478	0.475	↑4.92%	↑2.69%
		Avg	0.429	0.429	0.433	0.432	↑0.77%	↑0.86%
Cross-domain	Weather ↓ ETTh1	96	0.373	0.392	0.371	0.392	↓0.52%	↓0.06%
		192	0.425	0.422	0.421	0.421	↓0.97%	↓0.14%
		336	0.462	0.440	0.461	0.443	↓0.30%	↑0.67%
		720	0.467	0.469	0.478	0.475	↑2.40%	↑1.10%
		Avg	0.432	0.431	0.433	0.432	↑0.23%	↑0.42%
	Exchange ↓ ETTh1	96	0.375	0.391	0.371	0.392	↓0.90%	↑0.04%
		192	0.427	0.421	0.421	0.421	↓1.44%	↓0.12%
		336	0.464	0.441	0.461	0.443	↓0.77%	↑0.36%
		720	0.456	0.463	0.478	0.475	↑4.53%	↑2.48%
		Avg	0.431	0.429	0.433	0.432	↑0.50%	↑0.75%

4.7. Discussion

We have presented key results in the previous sections. In summary, LSPatch-T has demonstrated improvements in multivariate time series forecasting by leveraging self-supervised learning and views of token representation. However, there are several limitations to this relatively simple setup that we would like to address:

1. **Incomplete utilization of patch dependencies**: One obvious drawback of the model is that variate tokens may not fully exploit the patch dependencies learned during pretraining. The embedding layers used in pretraining and downstream tasks are derived from different projections, leading to separate embedding spaces. This discrepancy can increase the training time required for the downstream model to adapt to the patch token embedding space. A potential solution is to initialize the downstream embedding layer by replicating the pre-trained weights for each concatenated patch, providing better alignment between the two phases. Additionally, extending the framework to incorporate a dynamic range of window sizes would be a valuable direction for future research, but one should carefully investigate the treatment for time misalignment arisen with this design.
2. **Theoretical framework of variate tokens**: Our results demonstrate the promise of transfer learning with variate tokens, which is compromised by recent studies [8,29]. However, the theoretical foundation of variate tokens in the context of transfer learning has not been fully explored here. Future work will focus on situating this within a representation learning framework, with particular focus on

examining the structure of the embedding space where variate tokens can substitute patch tokens.

3. **Effectiveness of frequency Loss**: The ablation study indicates that combining frequency loss with time loss boost performance. In this work, the proposed frequency loss acts as a regularizer, ensuring balance across different spectral ranges, yet it may not be effective for datasets where seasonal patterns are less prominent. This limitation arises when frequency loss is calculated on individual short-term series and the long-term trends in the data are neglected. A proper improvement could involve using contrastive learning to feed multiviews (multi-segments) of the signal into the frequency analysis.

5. Conclusions

Pretraining with short-length patch tokens and channel independence enhances performance during downstream fine-tuning on the same dataset when using full-length variate tokens. Empirical studies focused on three public datasets demonstrate the effectiveness of our approach in two key areas: long-range forecasting and multivariate transfer learning. In summary, downstream forecasting with our LSPatch-T leads to 2.11% and 3.32% deductions in the average MSE compared to PatchTST and iTransformer, respectively. However, our work is not without limitations. Regarding our framework, which only investigates the trivial case of concatenation, inherently limiting its ability to capture local context, the extension to dynamic range of window sizes should be considered as a future research direction. Furthermore, several key aspects, such as developing a theoretical framework for embedding space or incorporating contrastive learning, have yet to be addressed. We hope this piece of work inspires researchers to further explore transfer learning in the context of time series analysis.

Author Contributions: Conceptualization, L.H.A. and D.T.V.; methodology, L.H.A. and S.O.; software, L.H.A. and N.B.N.H.; validation, L.H.A., S.O., N.B.N.H. and G.-H.Y.; formal analysis, D.T.V. and H.-G.K.; investigation, L.H.A. and S.O.; resources, S.O., J.-S.K. and G.-H.Y.; data curation, L.H.A.; writing—original draft preparation, D.T.V. and N.B.N.H.; writing—review and editing, D.T.V., H.-G.K., J.-S.K. and J.-Y.K.; visualization, L.H.A. and N.B.N.H.; supervision, J.-S.K., J.-Y.K. and G.-H.Y.; project administration, J.-Y.K. and G.-H.Y.; funding acquisition, J.-S.K., J.-Y.K. and G.-H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (IITP-2024-RS-2022-00156287, 50). And this work was partly supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (RS-2021-II212068, Artificial Intelligence Innovation Hub).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data is publicly available. ETT: github.com/zhouhaoyi/ETDataset; Exchange: github.com/laiguokun/multivariate-time-series-data/tree/master/exchange_rate; Weather: www.bgc-jena.mpg.de/wetter/ (accessed on May 2024).

Conflicts of Interest: Author Dang Thanh Vu is employed by AISeed Inc. However, the company did not influence the design, execution, data interpretation, writting or funding of the manuscript. Other authors declare no conflicts of interest.

References

1. Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **2020**, *36*, 1181–1191. [CrossRef]
2. Anh, L.H.; Yu, G.H.; Vu, D.T.; Kim, J.S.; Lee, J.I.; Yoon, J.C.; Kim, J.Y. Stride-TCN for Energy Consumption Forecasting and Its Optimization. *Appl. Sci.* **2022**, *12*, 9422. [CrossRef]

3. Anh, L.H.; Yu, G.H.; Vu, D.T.; Kim, H.G.; Kim, J.Y. DelayNet: Enhancing Temporal Feature Extraction for Electronic Consumption Forecasting with Delayed Dilated Convolution. *Energies* **2023**, *16*, 7662. [CrossRef]
4. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
5. Zhang, Y.; Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In Proceedings of the The Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
6. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv* **2022**, arXiv:2211.14730.
7. Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; Eickhoff, C. A transformer-based framework for multivariate time series representation learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 2114–2124.
8. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. itransformer: Inverted transformers are effective for time series forecasting. *arXiv* **2023**, arXiv:2310.06625.
9. Zhao, L.; Shen, Y. Rethinking Channel Dependence for Multivariate Time Series Forecasting: Learning from Leading Indicators. *arXiv* **2024**, arXiv:2401.17548.
10. Han, L.; Ye, H.J.; Zhan, D.C. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 7129–7142. [CrossRef]
11. Zhao, S.; Li, Z.; Zhou, X. Rethinking Self-Supervised Learning for Time Series Forecasting: A Temporal Perspective. *Knowl.-Based Syst.* **2024**, *305*, 112652. [CrossRef]
12. Li, Z.; Rao, Z.; Pan, L.; Wang, P.; Xu, Z. Ti-mae: Self-supervised masked time series autoencoders. *arXiv* **2023**, arXiv:2301.08871.
13. Das, A.; Kong, W.; Sen, R.; Zhou, Y. A decoder-only foundation model for time-series forecasting. *arXiv* **2023**, arXiv:2310.10688.
14. Gruver, N.; Finzi, M.; Qiu, S.; Wilson, A.G. Large language models are zero-shot time series forecasters. In Proceedings of the Annual Conference on Neural Information Processing Systems 2023, New Orleans, LA, USA, 10–16 December 2023; Volume 36.
15. Kazemi, S.M.; Goel, R.; Eghbali, S.; Ramanan, J.; Sahota, J.; Thakur, S.; Wu, S.; Smyth, C.; Poupart, P.; Brubaker, M. Time2vec: Learning a vector representation of time. *arXiv* **2019**, arXiv:1907.05321.
16. Yue, Z.; Wang, Y.; Duan, J.; Yang, T.; Huang, C.; Tong, Y.; Xu, B. Ts2vec: Towards universal representation of time series. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 8980–8987.
17. Xie, Z.; Zhang, Z.; Cao, Y.; Lin, Y.; Bao, J.; Yao, Z.; Dai, Q.; Hu, H. Simmim: A simple framework for masked image modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 9653–9663.
18. He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; Girshick, R. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 16000–16009.
19. Vaswani, A. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
20. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are transformers effective for time series forecasting? In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 24 January 2023; Volume 37, pp. 11121–11128.
21. Yun, C.; Bhojanapalli, S.; Rawat, A.S.; Reddi, S.J.; Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *arXiv* **2019**, arXiv:1912.10077.
22. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
23. Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A.X.; Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
24. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
25. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Proceedings of the International conference on machine learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 27268–27286.
26. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11106–11115.
27. Yi, K.; Zhang, Q.; Fan, W.; Wang, S.; Wang, P.; He, H.; An, N.; Lian, D.; Cao, L.; Niu, Z. Frequency-domain MLPs are more effective learners in time series forecasting. In Proceedings of the Annual Conference on Neural Information Processing Systems 2023, New Orleans, LA, USA, 10–16 December 2023; Volume 36.
28. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv* **2022**, arXiv:2210.02186.
29. Li, Z.; Qi, S.; Li, Y.; Xu, Z. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv* **2023**, arXiv:2305.10721.

30. Dong, J.; Wu, H.; Zhang, H.; Zhang, L.; Wang, J.; Long, M. Simmtm: A simple pre-training framework for masked time-series modeling. In Proceedings of the Annual Conference on Neural Information Processing Systems 2023, New Orleans, LA, USA, 10–16 December 2023; Volume 36.
31. Garza, A.; Challu, C.; Mergenthaler-Canseco, M. TimeGPT-1. *arXiv* **2023**, arXiv:2310.03589.
32. Goswami, M.; Szafer, K.; Choudhry, A.; Cai, Y.; Li, S.; Dubrawski, A. Moment: A family of open time-series foundation models. *arXiv* **2024**, arXiv:2402.03885.
33. Woo, G.; Liu, C.; Kumar, A.; Xiong, C.; Savarese, S.; Sahoo, D. Unified training of universal time series forecasting transformers. *arXiv* **2024**, arXiv:2402.02592.
34. Benidis, K.; Rangapuram, S.S.; Flunkert, V.; Wang, Y.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
35. Geva, M.; Schuster, R.; Berant, J.; Levy, O. Transformer feed-forward layers are key-value memories. *arXiv* **2020**, arXiv:2012.14913.
36. Lai, G.; Chang, W.C.; Yang, Y.; Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In Proceedings of the The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 95–104.
37. Cao, D.; Wang, Y.; Duan, J.; Zhang, C.; Zhu, X.; Huang, C.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; et al. Spectral temporal graph neural network for multivariate time-series forecasting. In Proceedings of the 34th Conference on Neural Information Processing Systems, Virtual, 6–12 December 2020; Volume 33, pp. 17766–17778.
38. Rangapuram, S.S.; Seeger, M.W.; Gasthaus, J.; Stella, L.; Wang, Y.; Januschowski, T. Deep state space models for time series forecasting. In Proceedings of the 31th Annual Conference on Neural Information Processing Systems 2018, Montreal, QC, Canada, 3–8 December 2018; Volume 31.
39. Wang, Z.; Kong, F.; Feng, S.; Wang, M.; Zhao, H.; Wang, D.; Zhang, Y. Is Mamba Effective for Time Series Forecasting? *arXiv* **2024**, arXiv:2403.11144. [CrossRef]
40. Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.H.; Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
41. Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; Kalagnanam, J. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 459–469.
42. Tolstikhin, I.O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24261–24272.
43. Lee, J.; Vu, D.T.; Yu, G.; Kim, J.; Kim, K.; Kim, J. SDSL: Spectral Distance Scaling Loss Pretraining SwinUNETR for 3D Medical Image Segmentation. *IEEE Access* **2024**, *12*, 126693–126706. [CrossRef]
44. Jiang, L.; Dai, B.; Wu, W.; Loy, C.C. Focal frequency loss for image reconstruction and synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 13919–13929.