xPatch: Dual-Stream Time Series Forecasting with Exponential Seasonal-Trend Decomposition

Artyom Stitsyuk [1], **Jaesik Choi** [1,2]

[1]Korea Advanced Institute of Science and Technology (KAIST), South Korea
[2]INEEJI, South Korea
{stitsyuk, jaesik.choi}@kaist.ac.kr

Abstract

In recent years, the application of transformer-based models in time-series forecasting has received significant attention. While often demonstrating promising results, the transformer architecture encounters challenges in fully exploiting the temporal relations within time series data due to its attention mechanism. In this work, we design e**X**ponential **Patch** (xPatch for short), a novel dual-stream architecture that utilizes exponential decomposition. Inspired by the classical exponential smoothing approaches, xPatch introduces the innovative seasonal-trend exponential decomposition module. Additionally, we propose a dual-flow architecture that consists of an MLP-based linear stream and a CNN-based non-linear stream. This model investigates the benefits of employing patching and channel-independence techniques within a non-transformer model. Finally, we develop a robust arctangent loss function and a sigmoid learning rate adjustment scheme, which prevent overfitting and boost forecasting performance.

Code — https://github.com/stitsyuk/xPatch

1 Introduction

Long-term time series forecasting (LTSF) is one of the fundamental tasks in time series analysis. The task is focused on predicting future values over an extended period, based on historical data. With the advent of deep learning models, they have recently demonstrated superior performance in LTSF compared to traditional approaches such as ARIMA (Box et al. 2015) and LSTM (Bahdanau, Cho, and Bengio 2015).

Transformer-based models (Vaswani et al. 2017) have revolutionized the LTSF task, enabling powerful AI systems to achieve state-of-the-art performance. The transformer architecture is considered highly successful in capturing semantic correlations among elements in long sequences. Recent works have primarily focused on adapting transformers to the LTSF task and addressing such limitations of the vanilla transformer as quadratic time and memory complexity (Li et al. 2019; Zhou et al. 2021; Wen et al. 2022).

The self-attention mechanism employed in transformers is permutation-invariant. Although techniques like positional encoding can partially retain ordering information, preserving temporal information remains a challenge for transformer-based models. This limitation can adversely affect the performance of the LTSF task dealing with a continuous set of points. As a result, the effectiveness of transformers in the LTSF task has been challenged by a simple linear approach utilizing a Multi-Layer Perceptron (MLP) network (Zeng et al. 2023). Surprisingly, a simple linear model named DLinear has surpassed the state-of-the-art forecasting performance of all previous transformer-based models, raising a fundamental question: "Are Transformers effective for long-term time series forecasting?".

Due to the non-stationary nature of real-world systems, time series data usually contain complex temporal patterns. To handle this complexity and non-stationarity (Liu et al. 2022), many recent LTSF models have adopted a paradigm of decomposing inputs. They use a seasonal-trend decomposition to capture linear trend features and non-linear seasonal variations. For handling time series trend features, certain transformer-based models, including Autoformer (Wu et al. 2021) and FEDformer (Zhou et al. 2022), incorporate seasonal-trend data decomposition. By partitioning the signal into two components, each with distinct function behavior, it becomes more feasible to capture semantic features from each component and make separate predictions.

Both Autoformer and FEDformer focus on refining the transformer architecture by introducing an auto-correlation mechanism and a frequency-enhanced method while decomposing the signal using a simple average pooling method. This technique requires padding at both ends, essentially repeating the last and first values. Consequently, we argue that this approach introduces a bias towards the initial and final values, potentially altering the behavior of trend values.

We propose a simple yet effective decomposition technique based on a generally applicable time series smoothing method named Exponential Moving Average (EMA) (Gardner Jr 1985). The proposed strategy assigns exponentially decreasing weights over time, facilitating more efficient feature learning from the decomposed data. The resulting exponentially smoothed sequence represents the trend, while the residual difference encapsulates the seasonality.

Currently, the state-of-the-art models for the LTSF task are transformer-based architectures CARD (Wang et al. 2024b) and PatchTST (Nie et al. 2023). These models rely on channel-independence and segmentation of time se-

ries into patches, which are used as input tokens for the transformer. However, we assume that the permutation-invariance of the attention mechanism in transformers may impede the model from attaining the optimal forecasting performance. Therefore, we are aiming to explore channel-independence and patching approaches within a non-transformer architecture, proposing the xPatch model.

In this study, we introduce the utilization of the exponential seasonal-trend decomposition technique. Furthermore, we propose a robust arctangent loss with weight decay and a novel learning rate adjustment strategy that improves training adaptability. Additionally, we present the xPatch, a novel dual-flow network architecture that integrates Convolutional Neural Networks (CNNs), Multi-Layer Perceptrons (MLPs), patching, channel-independence, exponential seasonal-trend decomposition, and dual stream prediction.

We summarized our main contributions as follows:

- We propose a novel method for seasonal-trend decomposition that utilizes an Exponential Moving Average (EMA).
- We introduce the dual-flow network and investigate the patching and channel-independence approaches within the CNN-based backbone.
- We develop a robust arctangent loss and a novel sigmoid learning rate adjustment scheme with a warm-up that results in smoother training.

2 Related Work

Informer (Zhou et al. 2021) is the first well-known transformer-based model designed for the LTSF task. It employs ProbSparse self-attention and a generative style decoder for addressing quadratic time and memory complexity. Notably, this work also contributes to the field by curating data and introducing the Electricity Transformer Temperature (ETT) benchmark dataset that is now commonly used for LTSF experiments by most of the models.

TimesNet (Wu et al. 2023) utilizes Fourier Transform to decompose time series into multiple components with varying period lengths, enhancing its focus on temporal variation modeling. The official repository provides a forecasting protocol with standardized hyperparameter settings and fairly implemented baselines.

To address the issue of non-stationarity in time series data, several models employ series decomposition to better capture complex temporal patterns. Autoformer (Wu et al. 2021) and FEDformer (Zhou et al. 2022) are two recent transformer-based solutions for the LTSF task, leveraging auto-correlation mechanism and frequency-enhanced structure, respectively. Both models incorporate seasonal-trend decomposition within each neural block to enhance the predictability of time-series data. Specifically, they apply a moving average kernel to the input sequence with padding at both ends, extracting the trend component. The difference between the original time series and the extracted trend component is identified as the seasonal component.

DLinear (Zeng et al. 2023) is a recent one-layer linear model that uses seasonal-trend decomposition as a preprocessing step. Initially, the model decomposes the raw data into trend and seasonal components using a moving average technique. Two linear layers are then applied independently to each of these components. The resulting features are subsequently aggregated to generate the final prediction.

MICN (Wang et al. 2023) is a recent CNN-based solution that employs multi-scale hybrid seasonal-trend decomposition. After decomposing the input series into seasonal and trend components, the model integrates both global and local contexts to enhance forecasting accuracy.

TimeMixer (Wang et al. 2024a) is an MLP-based approach that employs a decomposable multiscale-mixing method. The model uses the same series decomposition block from Autoformer (Wu et al. 2021) to break down multiscale time series into multiple seasonal and trend components. By leveraging the multiscale past information obtained after seasonal and trend mixing, the model predicts future values.

ETSformer (Woo et al. 2022) and CARD (Wang et al. 2024b) are two transformer-based architectures that incorporate the exponential smoothing approach. ETSformer introduces Exponential Smoothing Attention (ESA), while CARD applies exponential smoothing to the query and key tokens before the token blending module within one prediction head of the attention mechanism. In contrast to these models, the proposed xPatch architecture employs Exponential Moving Average (EMA) decomposition to separate the time series into trend and seasonal components, which are then processed separately.

Crossformer (Zhang and Yan 2022) and PatchTST (Nie et al. 2023) are transformer-based models that introduce a segmentation technique to LTSF. PatchTST divides time series data into subseries-level patches that serve as input tokens for the transformer. This approach is motivated by the vision transformer (Dosovitskiy et al. 2021) and designed for LTSF with channel-independence. Currently, PatchTST is recognized as the state-of-the-art solution for multivariate long-term forecasting. In our proposed xPatch model, we also incorporate patching and channel-independence approaches. Given that xPatch is a CNN-based approach, we investigate whether the superior performance of PatchTST can be attributed to its patching and channel-independence modules rather than its transformer architecture. To explore this, we examine if a CNN-based model can achieve improved results by leveraging these techniques.

MobileNet (Howard et al. 2017) and ConvMixer (Trockman and Kolter 2022) are notable models designed for Computer Vision (CV) tasks that demonstrate the advantages of depthwise separable convolutions. In the proposed xPatch approach, we incorporate depthwise separable convolution as the non-linear stream of the dual-flow network.

3 Proposed Method

In multivariate time series forecasting, given the observation of the historical L values $x = (x_1, x_2, ..., x_L)$, the task is to predict the future T timesteps $\hat{x} = (x_{L+1}, x_{L+2}, ..., x_{L+T})$. Each x_t value at timestep t is multivariate, representing a vector of M variables. Therefore, the multivariate lookback series is denoted as $x \in \mathbb{R}^{M \times L}$ and the multivariate prediction is represented by $\hat{x} \in \mathbb{R}^{M \times T}$.

3.1 Seasonal-Trend Decomposition

Seasonal-trend decomposition facilitates the learning of complex temporal patterns by separating the time series signal into trend and seasonal components. Trend features generally represent the long-term direction of the data, which can be linear or smoothly varying. In contrast, seasonal components capture repeating patterns or cycles that occur at regular intervals and are often non-linear due to the complexities and variations in periodic behavior. The model first learns the features of these components individually and then combines them to generate the final forecast.

Simple Moving Average (SMA) is the decomposition approach utilized in Autoformer (Wu et al. 2021), FEDformer (Zhou et al. 2022), DLinear (Zeng et al. 2023), MICN (Wang et al. 2023), and TimeMixer (Wang et al. 2024a) models. SMA is defined as the unweighted mean of the previous k data points.

Moving average mean point s_t of the k entries with t being moving step, n being dataset length, and $X = x_1, x_2, ..., x_n$ being data points is calculated as:

$$\begin{aligned} s_t &= \frac{x_t + x_{t+1} + ... + x_{t+k-1}}{k} = \frac{1}{k}\sum_{i=t}^{t+k-1} x_i \\ X_T &= \text{AvgPool}(\text{Padding}(X)) \\ X_S &= X - X_T \end{aligned} \tag{1}$$

where AvgPool$(\cdot)$ denotes moving average with the padding operation, while X_T and X_S correspond to trend and seasonality components. Padding is employed to maintain the length of the time series unchanged after performing average pooling. Figure 1 illustrates an example of SMA decomposition.

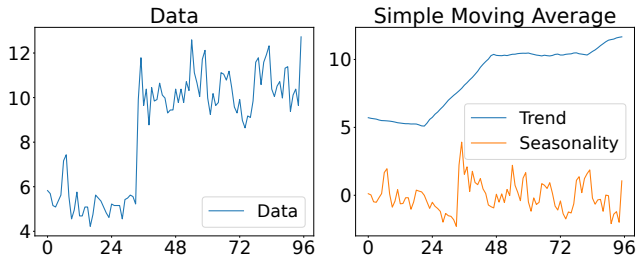

Figure 1: Example of SMA decomposition with kernel k = 25 on a 96-length sample from the ETTh1 dataset.

Firstly, we argue that the average pooling operation results in the loss of significant trend features (see Appendix B). Additionally, alignment requires padding on both ends of the series, which can distort the sequence at the head and tail.

Secondly, the primary goal of decomposition is to enhance the interpretability of both decomposed signals. This entails improving the clarity of the trend and seasonality components while enriching them with more distinct features for learning. However, SMA produces an overly simplistic trend signal with limited diverse features and a complex seasonality pattern. As a result, we investigate an alternative decomposition method to address this issue.

Exponential Moving Average (EMA) (Gardner Jr 1985) is an exponential smoothing method that assigns greater weight to more recent data points while smoothing out older data. This exponential weighting scheme allows EMA to respond more promptly to changes in the underlying trends of the time series, without the need for padding repeated values.

EMA point s_t of data x_t beginning at time $t = 0$ is represented by:

$$\begin{aligned} s_0 &= x_0 \\ s_t &= \alpha x_t + (1-\alpha)s_{t-1},\ t > 0 \\ X_T &= \text{EMA}(X) \\ X_S &= X - X_T \end{aligned} \tag{2}$$

where α is the smoothing factor, $0 < \alpha < 1$, EMA$(\cdot)$ denotes exponential moving average, while X_T and X_S correspond to trend and seasonality components. Figure 2 shows an example of EMA decomposition.

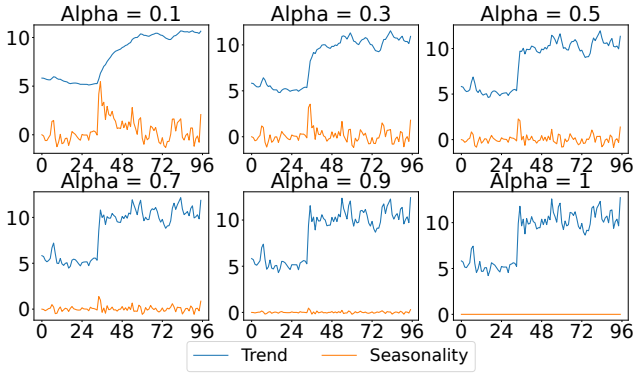

Figure 2: Example of EMA decomposition with $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9, 1\}$ on a 96-length sample from the ETTh1 dataset.

The exponential method offers greater control over the behavior of both trend and seasonality components. Given that data can exhibit diverse patterns, including stationary and non-stationary characteristics with varying periods and behaviors, the adaptability of exponential decomposition provides advantages in feature extraction (see Appendix B). Compared to SMA, EMA presents a more flexible approach to decomposition, as it adjusts its weighting scheme based on the exponential decay of data points. This adaptability allows EMA to capture changing trends more effectively, making it particularly suitable for time series with dynamic and evolving patterns (see Appendix C).

3.2 Model Architecture

Channel-Independence. The multivariate time series $x = (x_1, x_2, ..., x_L)$ is divided into M univariate sequences $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_L^{(i)})$, where $x^{(i)} \in \mathbb{R}^L$ and L is lookback of recent historical data points. Each of these univariate series is then individually fed into the backbone model, which consequently generates a prediction sequence $\hat{x}^{(i)} = (\hat{x}_{L+1}^{(i)}, \hat{x}_{L+2}^{(i)}, ..., \hat{x}_{L+T}^{(i)})$, where $\hat{x}^{(i)} \in \mathbb{R}^T$ and T

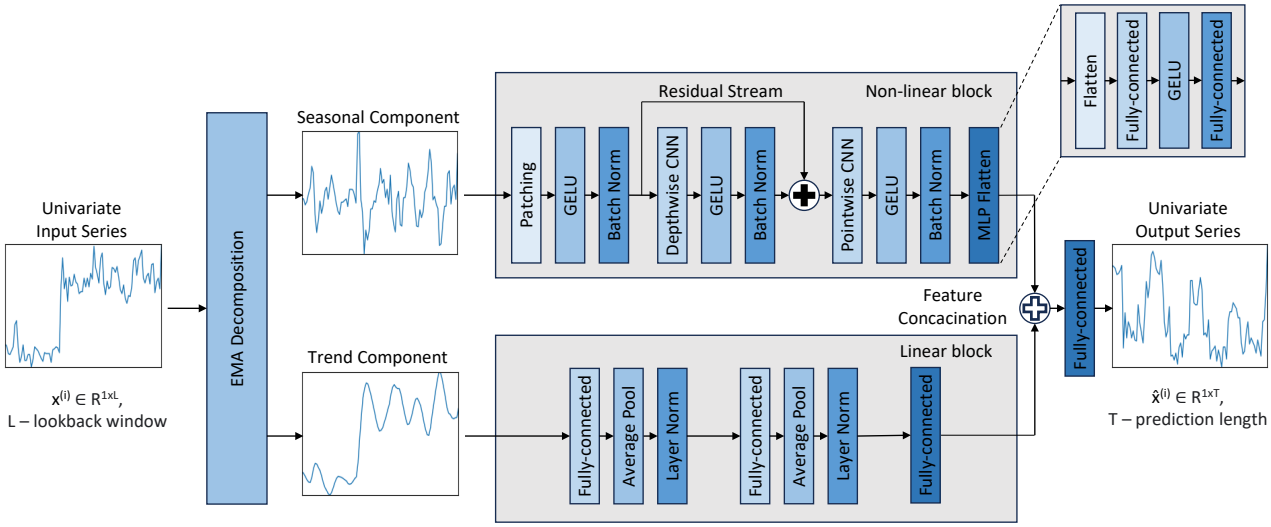

Figure 3: xPatch Model Overview. Every univariate series is passed through exponential decomposition. Consequently, the trend and seasonal components are processed through the dual flow network.

is future steps observations. This partitioning approach has proven to work well in both linear models and transformers (Zeng et al. 2023; Nie et al. 2023; Han, Ye, and Zhan 2023).

Exponential Decomposition. Using the EMA method, we decompose each univariate series into trend and seasonality components, which are then processed separately by the dual-flow architecture. After processing, the learned trend and seasonal features are aggregated and passed to the final output layer to comprise the final prediction as illustrated in Figure 3. Details on optimization and ablation studies of EMA are available in Appendix D, E.

Dual Flow Net. As the main backbone, we employ two distinct flows to analyze trend and seasonality: linear and non-linear streams. The trend component is processed through the linear MLP-based stream, while the seasonal component is handled by the non-linear CNN-based block.

Seasonality represents periodic fluctuations around a constant level, meaning that the statistical properties of these fluctuations, such as mean and variance, remain stable over time, meaning that the seasonal component is stationary. In contrast, the trend reflects long-term progression with either increasing or decreasing behavior and a changing mean, which makes the trend component non-stationary.

To summarize, in most cases, the seasonal component is non-linear and stationary, while the trend component is linear and non-stationary. However, some datasets might exhibit unusual behavior, such as a stationary trend. Therefore, the dual-stream architecture is designed to enhance the model's adaptability to both stationary and non-stationary data. For the exploration of the dual-flow architecture, see Appendix F.

Linear Stream. The linear stream is an MLP-based network that includes average pooling and layer normalization, intentionally omitting activation functions to emphasize linear features.

The decomposed data $x^{(i)}$ is processed through two linear blocks, each consisting of a fully connected layer followed by average pooling with a kernel $k = 2$ for feature smoothing and layer normalization for training stability. Each linear layer and average pooling operation contribute to dimensionality reduction, encouraging the network to compress feature representations to fit the available space effectively. This reduction in the number of features, combined with the absence of activation functions and a bottleneck architecture, aims to retain only the most significant linear features of the smoothed trend.

$$x^{(i)} = \text{LayerNorm}(\text{AvgPool}(\text{Linear}(x^{(i)}), k = 2)) \quad (3)$$

The final expansion layer takes the bottleneck representation and upscales it to the prediction length.

$$\hat{x}^{(i)}_{lin} = \text{Linear}(x^{(i)}) \quad (4)$$

Patching. Patching is a technique inspired by the vision transformer (Dosovitskiy et al. 2021) and was first introduced in the context of LTSF by PatchTST (Nie et al. 2023). This method unfolds each univariate time series using a sliding window. We incorporate patching into the non-linear block to emphasize repetitive seasonal features. By using patching, the model can better focus on these repetitive patterns, effectively capturing their inter-pattern dependencies more effectively.

The patch length is denoted as P, and the non-overlapping region between two consecutive patches is referred to as stride S. We apply patching in the non-linear stream to each normalized univariate decomposed sequence $x^{(i)} \in \mathbb{R}^L$, which generates a sequence of N 2D patches $x^{(i)}_p \in \mathbb{R}^{N \times P}$. The number of patches is calculated as $N = \lfloor \frac{L-P}{S} \rfloor + 2$. In our implementation, for a fair comparison with PatchTST and CARD, we adopt their setup for patch embedding, setting $P = 16$ and $S = 8$.

Non-linear Stream. The non-linear stream is a CNN-based network that introduces non-linearity through activa-

tion functions. By applying convolutions on top of patching, the CNN-based stream captures spatio-temporal patterns and inter-patch correlations, focusing on the non-linear features of the seasonal signal.

First, the patched data $x_p^{(i)} \in \mathbb{R}^{N \times P}$ is embedded for increasing the number of features with activation function σ and batch normalization (Ioffe and Szegedy 2015). Since the seasonal variations have many zero values, we employ GELU (Hendrycks and Gimpel 2016) as an activation function for its smooth transition around zero and non-linearity. The resulting embedded shape is denoted as $x_p^{N \times P^2}$.

$$x_p^{N \times P^2} = \text{BatchNorm}(\sigma(\text{Embed}(x_p^{(i)}))) \tag{5}$$

Following embedding, the data is processed through depthwise separable convolution. This method splits the computation into two steps: depthwise convolution applies a single convolutional filter per input channel, and pointwise convolution creates a linear combination of the output of the depthwise convolution, with an additional residual stream between them.

Given that the xPatch architecture leverages channel-independence, it was determined to employ patching to increment the number of dimensions, enabling patches to function as channels in the data $x_p^{N \times P^2}$. Consequently, rather than relying on inter-channel feature representations, we utilize channel-independent inter-patch representations. This approach aims to capture comprehensive semantic information that may not be available at the point level and allows to focus on non-linear features.

For depthwise convolution, we employ grouped convolution with the number of groups g equal to the number of patches N, a large kernel size k equal to the patch length P, and a convolution stride s equal to the patch length P.

$$x_p^{N \times P} = \text{Conv}_{N \to N}(x_p^{N \times P^2}, k = P, s = P, g = N) \tag{6}$$

$$x_p^{N \times P} = \text{BatchNorm}(\sigma(x_p^{N \times P})) \tag{7}$$

Depthwise convolution applies a single convolutional filter per input channel, generating N feature maps, each corresponding to a specific patch. This approach enables the model to capture temporal features with group convolution that is consistent for periodic patches.

Subsequently, the data is updated with a linear residual connection spanning the depthwise convolution. Although depthwise convolution captures temporal relations between periodic patterns, it may not effectively capture inter-patch feature correlations. Therefore, the sequence is further processed through the pointwise convolution layer with the number of groups $g = 1$, a small kernel size $k = 1$, and a convolution stride $s = 1$.

$$x_p^{N \times P} = \text{DepthwiseConv}(x_p^{N \times P^2}) + x_p^{N \times P^2} \tag{8}$$

$$x_p^{N \times P} = \text{Conv}_{N \to N}(x_p^{N \times P}, k = 1, s = 1, g = 1) \tag{9}$$

$$x_p^{N \times P} = \text{BatchNorm}(\sigma(x_p^{N \times P})) \tag{10}$$

Pointwise convolution creates a linear combination of the output and aggregates features across different patches without skipping elements.

These predictions are then processed through the MLP flatten layer. This layer is designed in a similar style to PatchTST: the first linear layer doubles the hidden dimension, while the second linear layer projects it back with a GELU activation function between them.

$$\hat{x}_{nonlin}^{(i)} = \text{Linear}(\sigma(\text{Linear}(\text{Flatten}(x_p^{N \times P})))) \tag{11}$$

Finally, linear features (4) and non-linear features (11) are concatenated and fed into the final linear layer, which merges linear and non-linear features for the output prediction.

$$\hat{x}^{(i)} = \text{Linear}(\text{concat}(\hat{x}_{lin}^{(i)}, \hat{x}_{nonlin}^{(i)})) \tag{12}$$

We concatenate the linear and non-linear features from the two flows, representing learned representations from the MLP and CNN streams. This mechanism enables the model to dynamically weigh the significance of both linear and non-linear features in the final prediction, providing adaptability to diverse patterns in time series data.

3.3 Loss Function

Mean Squared Error (MSE) loss is a training loss scheme commonly used by LTSF models. The MSE loss $\mathcal{L}_{\text{MSE}}$ between the predicted univariate sequence $\hat{x}_{1:T}^{(i)}$ and the ground truth observations $x_{1:T}^{(i)}$, where T is future prediction length, is denoted as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{T} \sum_{i=1}^{T} ||\hat{x}_{1:T}^{(i)} - x_{1:T}^{(i)}||_2^2 \tag{13}$$

The recent transformer-based model CARD (Wang et al. 2024b) introduced a novel signal decay-based loss function, where they scale down the far-future Mean Absolute Error (MAE) loss to address the high variance. MAE was chosen since it is more resilient to outliers than MSE.

$$\mathcal{L}_{\text{CARD}} = \frac{1}{T} \sum_{i=1}^{T} i^{-\frac{1}{2}} ||\hat{x}_{1:T}^{(i)} - x_{1:T}^{(i)}|| \tag{14}$$

where i corresponds to the prediction point in the future. This training scheme was proven by CARD to be efficient and to increase the performance of existing models.

To identify a more effective scaling loss coefficient, we extend Equation (14) to a universally applicable MAE scalable loss function:

$$\mathcal{L} = \frac{1}{T} \sum_{i=1}^{T} \rho(i) ||\hat{x}_{1:T}^{(i)} - x_{1:T}^{(i)}|| \tag{15}$$

where $\rho(i)$ represents the scaling coefficient. Thus, the $\mathcal{L}_{\text{CARD}}$ loss defined in Equation (14) emerges as a specific instance of the scalable loss function delineated in Equation (15), with $\rho(i) = i^{-\frac{1}{2}}$.

We find that the scaling coefficient $\rho_{CARD}(i) = i^{-\frac{1}{2}}$ exhibits a too rapid decrease rate for our task. Therefore, we propose a novel arctangent loss $\mathcal{L}_{arctan}$, which features a

Models	**xPatch (ours)**		CARD (2024)		TimeMixer (2024)		iTransformer (2024)		RLinear (2023)		PatchTST (2023)		MICN (2023)		DLinear (2023)		TimesNet (2023)		ETSformer (2022)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.428**	**0.419**	0.442	0.429	0.447	0.44	0.454	0.448	0.438	0.427	0.45	0.441	0.559	0.535	0.456	0.452	0.458	0.450	0.542	0.510
ETTh2	**0.319**	**0.361**	0.368	0.390	0.365	0.395	0.383	0.407	0.362	0.394	0.365	0.394	0.588	0.525	0.559	0.515	0.414	0.427	0.439	0.452
ETTm1	**0.377**	0.384	0.382	**0.383**	0.381	0.396	0.407	0.410	0.409	0.401	0.383	0.394	0.392	0.414	0.403	0.407	0.400	0.406	0.429	0.425
ETTm2	**0.267**	**0.313**	0.272	0.317	0.275	0.323	0.288	0.332	0.286	0.328	0.284	0.327	0.328	0.382	0.350	0.401	0.291	0.333	0.293	0.342
Weather	**0.232**	**0.261**	0.239	0.265	0.240	0.272	0.258	0.278	0.269	0.289	0.257	0.280	0.243	0.299	0.265	0.317	0.259	0.287	0.271	0.334
Traffic	0.499	**0.279**	0.453	0.282	0.485	0.298	**0.428**	0.282	0.623	0.372	0.467	0.292	0.542	0.316	0.625	0.383	0.620	0.336	0.621	0.396
Electricity	0.179	0.264	**0.168**	**0.258**	0.182	0.273	0.178	0.270	0.214	0.291	0.190	0.275	0.187	0.295	0.212	0.300	0.193	0.295	0.208	0.323
Exchange	0.375	0.408	0.360	0.402	0.408	0.422	0.360	0.403	0.380	0.410	0.364	**0.400**	**0.315**	0.404	0.354	0.414	0.416	0.443	0.410	0.427
Solar	0.239	**0.236**	0.237	0.239	**0.216**	0.280	0.233	0.262	0.369	0.357	0.254	0.289	0.283	0.358	0.327	0.398	0.301	0.319	0.603	0.615
ILI	**1.442**	**0.725**	1.916	0.842	1.708	0.820	2.918	1.154	2.452	0.978	1.626	0.804	2.664	1.086	2.616	1.090	2.139	0.931	2.497	1.004

Table 1: Averaged long-term forecasting results with unified lookback window $L = 36$ for the ILI dataset, and $L = 96$ for all other datasets. All results are averaged from 4 different prediction lengths: $T = \{24, 36, 48, 60\}$ for the ILI dataset, and $T = \{96, 192, 336, 720\}$ for all other datasets, respectively. The best model is **boldface** and the second best is underlined. See Table 13 in Appendix K for the full results.

slower increase rate compared to the exponential functions analyzed in CARD (Wang et al. 2024b):

$$\mathcal{L}_{arctan} = \frac{1}{T}\sum_{i=1}^{T} \rho_{arctan}(i)||\hat{x}_{1:T}^{(i)} - x_{1:T}^{(i)}|| \tag{16}$$

$$\rho_{arctan}(i) = -\arctan(i) + \frac{\pi}{4} + 1 \tag{17}$$

Mathematical proofs, ablation studies on state-of-the-art models employing the arctangent loss, and the arctangent function's scaling analysis can be found in Appendix G.

3.4 Learning Rate Adjustment Scheme

Most recent LTSF models (Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022; Woo et al. 2022; Wu et al. 2023; Zeng et al. 2023; Li et al. 2023; Liu et al. 2024) adapt standard learning rate adjustment technique. Learning rate α_t at epoch t with initial learning rate α_0 is calculated as:

$$\alpha_t = \alpha_{t-1} * 0.5^{t-1}, \text{ for } t \geq 1 \tag{18}$$

This strategy results in a decreasing learning rate with each successive epoch. Such a rapidly decreasing scheme was effective since the models were trained with a small number of epochs, usually limited to 10.

PatchTST (Nie et al. 2023) introduced a long training approach with an upper limit of 100 epochs and a new learning rate adjustment schedule:

$$\alpha_t = \alpha_0, \text{ for } t < 3, \tag{19}$$

$$\alpha_t = \alpha_{t-1} * 0.9^{t-3}, \text{ for } t \geq 3 \tag{20}$$

Consequently, CARD (Wang et al. 2024b) developed a new linear warm-up of the model with subsequent cosine learning rate decay. Learning rate α_t at epoch t with initial learning rate α_0, number of warmup epochs w, and upper limit of 100 epochs is calculated as:

$$\alpha_t = \alpha_{t-1} * \frac{t}{w}, \text{ for } t < w, \tag{21}$$

$$\alpha_t = 0.5\alpha(1 + cos(\pi * \frac{(t-w)}{100-w})), \text{ for } t \geq w \tag{22}$$

We introduce a novel sigmoid learning rate adjustment scheme. The learning rate α_t at epoch t, with an initial learning rate α_0, logistic growth rate k, decreasing curve smoothing rate s, and warm-up coefficient w, is calculated as follows:

$$\alpha_t = \frac{\alpha_0}{1 + e^{-k(t-w)}} - \frac{\alpha_0}{1 + e^{-\frac{k}{s}(t-sw)}} \tag{23}$$

Mathematical proofs, ablation studies on state-of-the-art models using the sigmoid learning rate adjustment approach, and hyperparameters selection are available in Appendix H.

4 Experiments

Datasets. We conduct extensive experiments on nine real-world multivariate time series datasets, including Electricity Transform Temperature (ETTh1, ETTh2, ETTm1, ETTm2) (Zhou et al. 2021), Weather, Traffic, Electricity, Exchange-rate, ILI (Wu et al. 2021), and Solar-energy (Lai et al. 2018).

Evaluation Metrics. Following previous works, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics to assess the performance.

Implementation Details. All the experiments are implemented in PyTorch (Paszke et al. 2019), and conducted on a single Quadro RTX 6000 GPU.

Baselines. We choose the last state-of-the-art LTSF models, including Autoformer (2021) (Wu et al. 2021), FEDformer (2022) (Zhou et al. 2022), ETSformer (2022) (Woo et al. 2022), TimesNet (2023) (Wu et al. 2023), DLinear (2023) (Zeng et al. 2023), RLinear (2023) (Li et al. 2023), MICN (2023) (Wang et al. 2023), PatchTST (2023) (Nie et al. 2023), iTransformer (2024) (Liu et al. 2024), TimeMixer (2024) (Wang et al. 2024a), and CARD (2024) (Wang et al. 2024b) as baselines for our experiments.

Unified Experimental Settings. To ensure a fair comparison, we conduct 2 types of experiments. The first experiment uses unified settings based on the forecasting protocol proposed by TimesNet (Wu et al. 2023): a lookback length $L = 36$, prediction lengths $T = \{24, 36, 48, 60\}$ for the ILI dataset, and $L = 96$, $T = \{96, 192, 336, 720\}$ for all other datasets. The averaged results are reported in Table 1.

Models	**xPatch (ours)**		CARD (2024)		TimeMixer (2024)		iTransformer (2024)		RLinear (2023)		PatchTST (2023)		MICN (2023)		DLinear (2023)		TimesNet (2023)		ETSformer (2022)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.391**	**0.412**	0.401	0.422	0.411	0.423	0.501	0.492	0.413	0.427	0.413	0.434	0.440	0.462	0.423	0.437	0.458	0.450	0.542	0.510
ETTh2	**0.299**	**0.351**	0.321	0.373	0.316	0.384	0.385	0.417	0.328	0.382	0.331	0.381	0.403	0.437	0.431	0.447	0.414	0.427	0.439	0.452
ETTm1	**0.341**	**0.368**	0.350	**0.368**	0.348	0.376	0.373	0.404	0.359	0.378	0.353	0.382	0.387	0.411	0.357	0.379	0.400	0.406	0.429	0.425
ETTm2	**0.242**	**0.300**	0.255	0.310	0.256	0.316	0.274	0.335	0.253	0.313	0.256	0.317	0.284	0.340	0.267	0.332	0.291	0.333	0.293	0.342
Weather	**0.211**	**0.247**	0.220	0.248	0.222	0.262	0.271	0.297	0.242	0.278	0.226	0.264	0.243	0.299	0.246	0.300	0.259	0.287	0.271	0.334
Traffic	0.392	**0.248**	0.381	0.251	0.388	0.263	**0.378**	0.270	0.417	0.283	0.391	0.264	0.542	0.316	0.434	0.295	0.620	0.336	0.621	0.396
Electricity	**0.153**	**0.245**	0.157	0.251	0.156	0.247	0.161	0.257	0.164	0.257	0.159	0.253	0.187	0.295	0.166	0.264	0.193	0.295	0.208	0.323
Exchange	0.366	0.404	0.360	0.402	0.471	0.452	0.458	0.469	0.423	0.427	0.405	0.426	0.315	0.404	**0.297**	**0.378**	0.416	0.443	0.410	0.427
Solar	0.194	**0.214**	0.198	0.225	**0.192**	0.244	0.197	0.262	0.235	0.266	0.256	0.298	0.213	0.266	0.329	0.400	0.244	0.334	0.603	0.615
ILI	**1.281**	**0.688**	1.916	0.842	1.971	0.924	2.947	1.193	1.803	0.874	1.480	0.807	2.567	1.056	2.169	1.041	2.139	0.931	2.497	1.004

Table 2: Averaged long-term forecasting results under hyperparameter searching. All results are averaged from 4 different prediction lengths: $T = \{24, 36, 48, 60\}$ for the ILI dataset, and $T = \{96, 192, 336, 720\}$ for all other datasets, respectively. The best model is **boldface** and the second best is underlined. See Table 14 in Appendix K for the full results.

To handle data heterogeneity and distribution shift, we apply reversible instance normalization (Kim et al. 2021). In Appendix J, we examine the impact of instance normalization on the forecasting results of xPatch and other state-of-the-art models, comparing their performance with and without the RevIN module.

Hyperparameter Search. In the second experiment, we aim to determine the upper bounds of the compared models and conduct a hyperparameter search. We evaluate all models to see if they benefit from longer historical data to identify the optimal lookback length for each, as detailed in Appendix I. For the models that benefit from a longer input length, namely xPatch, CARD, TimeMixer, iTransformer, RLinear, PatchTST, and DLinear, we perform a hyperparameter search similar to TimeMixer (Wang et al. 2024a). The averaged results are reported in Table 2.

All implementations are derived from the models' official repository code, maintaining the same configurations. It is also important to note that we strictly adhere to the settings specified in the official implementations, including the number of epochs (100 for CARD and PatchTST, 15 for RLinear) and the learning rate adjustment strategy.

Results. In the unified experimental settings, xPatch achieves the best averaged performance on 60% of the datasets using the MSE metric and 70% of the datasets using the MAE metric. Compared to CARD, xPatch surpasses it by 2.46% in MSE and 2.34% in MAE. Compared to TimeMixer, xPatch surpasses it by 3.34% in MSE and 6.34% in MAE. Compared to PatchTST, xPatch surpasses it by 4.76% in MSE and 6.20% in MAE.

In the hyperparameter search settings, xPatch achieves the best averaged performance on 70% of the datasets using the MSE metric and 90% of the datasets using the MAE metric. Compared to CARD, xPatch surpasses it by 5.29% in MSE and 3.81% in MAE. Compared to TimeMixer, xPatch surpasses it by 7.45% in MSE and 7.85% in MAE. Compared to PatchTST, xPatch surpasses it by 7.87% in MSE and 8.59% in MAE.

Computational Cost. While it is true that the proposed dual-flow architecture incurs higher computational costs compared to single-stream CNN and MLP models, it is important to note that convolution and linear operations are initially not as computationally expensive as transformer-based solutions. The overall increase in computational costs remains relatively small, as shown in Table 3. Moreover, the enhanced performance of the introduced dual-stream architecture outweighs these additional computational costs.

Method	Training time	Inference time
MLP-stream	0.948 msec	0.540 msec
CNN-stream	1.811 msec	0.963 msec
xPatch	3.099 msec	1.303 msec
CARD	14.877 msec	7.162 msec
TimeMixer	13.174 msec	8.848 msec
iTransformer	6.290 msec	2.743 msec
PatchTST	6.618 msec	2.917 msec
DLinear	0.420 msec	0.310 msec

Table 3: The average per step running and inference time maintaining the same settings for all benchmarks.

5 Conclusion

This study introduces xPatch, a novel dual-flow architecture for long-term time series forecasting (LTSF). xPatch combines the strengths of both Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) to achieve superior performance. Our findings demonstrate that the integration of an Exponential Moving Average (EMA) seasonal-trend decomposition module effectively captures underlying trends and enhances forecasting accuracy. The dual-stream network further enhances xPatch's adaptability by dynamically weighing the importance of linear and non-linear features for diverse time series patterns. Additionally, this study introduces a robust arctangent loss function and a novel sigmoid learning rate adjustment approach, both of which consistently improve the performance of existing models. By investigating patching and channel-independence within a CNN-based backbone, xPatch offers a compelling alternative to transformer-based architectures, achieving superior performance while maintaining computational efficiency.

Acknowledgements

We would like to thank Enver Menadjiev, Kyowoon Lee, Jihyeon Seong, Jiyeon Han and the anonymous reviewers for their valuable comments.

This work was supported by NAVER, Institute of Information & Communications Technology Planning & Evaluation (IITP), and the Korean Ministry of Science and ICT (MSIT) under the grant agreement No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST); No. RS-2022-II220984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation; No.RS-2022-II220184, Development and Study of AI Technologies to Inexpensively Conform to Evolving Policy on Ethics.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Box, G. E.; Jenkins, G. M.; Reinsel, G. C.; and Ljung, G. M. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Dickey, D. A.; and Fuller, W. A. 1979. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a): 427–431.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*.

Gardner Jr, E. S. 1985. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1): 1–28.

Han, L.; Ye, H.-J.; and Zhan, D.-C. 2023. The Capacity and Robustness Trade-off: Revisiting the Channel Independent Strategy for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2304.05206*.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. pmlr.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, 95–104.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.

Li, Z.; Qi, S.; Li, Y.; and Xu, Z. 2023. Revisiting Long-term Time Series Forecasting: An Investigation on Linear Mapping. *arXiv preprint arXiv:2305.10721*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35: 9881–9893.

Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Trockman, A.; and Kolter, J. Z. 2022. Patches Are All You Need? *Trans. Mach. Learn. Res.*

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, H.; Peng, J.; Huang, F.; Wang, J.; Chen, J.; and Xiao, Y. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*.

Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and Zhou, J. 2024a. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*.

Wang, X.; Zhou, T.; Wen, Q.; Gao, J.; Ding, B.; and Jin, R. 2024b. CARD: Channel Aligned Robust Blend Transformer for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; and Hoi, S. 2022. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *International Conference on Learning Representations*.

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

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

Zhang, Y.; and Yan, J. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 27268–27286. PMLR.