# FTDL: Enhancing Time Series Forecasting Through Frequency-Domain and Temporal-Domain Learning

**Anonymous submission**


## Abstract

Multivariate time series forecasting has seen rapid progress, with recent models leveraging attention mechanisms, patch-based processing, and variable correlation learning. However, it appears that many of these approaches do not fully leverage the fundamental characteristics of time series—namely, its inherent dual structure across temporal and frequency domains. In this work, we revisit this foundation and propose a simple yet effective framework, termed *FTDL* (Frequency-Temporal Domain Learning), which explicitly extracts and integrates prediction-relevant features from both domains. Unlike prior models that rely heavily on architectural complexity, our method emphasizes interpretability and efficiency by decomposing the learning process into two complementary views: temporal continuity and spectral composition. Our experimental results reveal that the frequency domain component predominantly captures high-frequency patterns, while the temporal domain component focuses on low-frequency trends, highlighting their complementary roles in modeling time series data. Through extensive experiments on real-world datasets, we demonstrate that *FTDL* achieves competitive or superior accuracy compared to state-of-the-art models, while maintaining a lightweight and intuitive design.


## 1 Introduction

Time series forecasting is a foundational technique in domains such as finance, economics, weather prediction, and industrial operations. Accurate forecasts enable informed decision-making, efficient resource allocation, and strategic planning. The success of time series forecasting depends on the presence of consistent and interpretable patterns—such as trends, seasonality, and bounded oscillations—that persist over time. Extracting such features from historical data is essential.

However, the increasing availability of large-scale and high-dimensional datasets has introduced new challenges. Traditional statistical methods, including autoregressive integrated moving average (ARIMA) and exponential smoothing, often struggle to capture the nonlinear dependencies and complex dynamics present in modern time series (Box et al. 2015; Hyndman and Athanasopoulos 2018). In response, recent research has focused on transformer-based architectures and their derivatives, which have demonstrated strong performance in modeling long-range dependencies and handling multivariate inputs (Liu et al. 2021; Wu et al. 2021; Zhang and Yan 2022; Zhou et al. 2022; Hugging Face 2023; Wu et al. 2022; Nie et al. 2023; Zeng et al. 2023; Ni et al. 2023; Chen et al. 2023; Ekambaram et al. 2023; Wang et al. 2025). Despite their effectiveness, these models often exhibit substantial architectural complexity, which leads to increased computational and memory costs. This raises practical concerns regarding scalability, training time, and deployment efficiency in real-world forecasting applications (Zeng et al. 2023; Hugging Face 2023; Liu et al. 2023).

To address these concerns, we propose a simple yet effective approach that emphasizes domain-aware design over architectural novelty. Our method, Frequency-Temporal Domain Learning (FTDL), is motivated by the observation that time series data can be meaningfully represented in both the temporal and frequency domains. While the temporal view captures trends and fluctuations over time, the frequency domain reveals periodic structures and underlying components that may be less apparent in the time domain.

Figure 1 illustrates a synthetic time series in both representations. The temporal domain shows a waveform with trend and noise, whereas the frequency domain highlights distinct periodic components and low-frequency trends. This dual perspective suggests that combining insights from both domains can enhance feature extraction for forecasting.

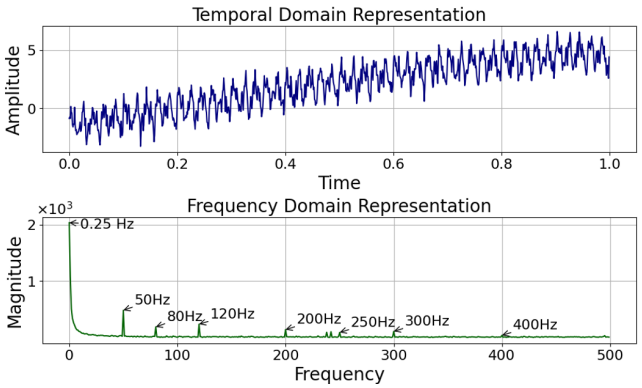



Figure 1: A synthetic time series illustrated in both the temporal and frequency domains, showing distinct patterns in each representation.

This dual perspective also relates to a known limita-

tion in neural networks called *spectral bias*, where multilayer perceptrons (MLPs) tend to prioritize learning low-frequency patterns over high-frequency ones (Tancik et al. 2020; Uteuliyeva et al. 2020; Basri et al. 2020; Luo et al. 2019). This bias can hinder the accurate modeling of sharp transitions or periodic fluctuations in time series data. To mitigate this, we focus on frequency-domain representations to help the model better capture and utilize high-frequency components.

Building on this motivation, and inspired by the work of (Zeng et al. 2023), which shows that even simple linear models can serve as competitive baselines with strong interpretability, we take a different perspective from transformer-based forecasters. Rather than relying on architectural complexity, our approach emphasizes the intrinsic structure of time series data, including decomposition into trend, seasonal, and residual components.

By leveraging these structural insights, we aim to design a model that is both interpretable and effective. Our contributions are summarized as follows:

- **Frequency-domain Representation** We investigate multiple frequency-domain formats—including magnitude, phase, and complex values—and identify effective representations for time series forecasting. By incorporating complex-valued neural networks (CVNNs), our model captures rich spectral features while maintaining interpretability and computational efficiency.
- **Dual-domain Feature Integration** We design and evaluate strategies for integrating temporal and frequency-domain features. This dual-domain approach enables the model to adapt to diverse data characteristics and demonstrates the complementary strengths of both domains in improving predictive performance.
- **Model Simplicity** Our model architecture is intentionally lightweight, incorporating standard components such as fully connected layers (FCs), multilayer perceptrons (MLPs), and skip connections. These choices ensure a balance between simplicity and expressiveness, allowing for efficient training and deployment.
- **Feature Effectiveness Analysis** We conduct extensive ablation studies to assess the individual and joint contributions of both domain features. By comparing model variants that isolate each domain, we provide insights into their respective roles and validate the effectiveness of dual-domain learning.

We benchmark our method on a diverse set of publicly available datasets, demonstrating its robustness and generalizability across a wide range of forecasting scenarios.

## 2 Preliminary

**Problem Definition** We begin by defining the forecasting problem: given a collection of multivariate time series samples $\mathcal{X} = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^{D\times L}\}$, where each sample $\mathbf{x}$ represents a multivariate time series with a look-back window of length $L$, and each time step $t$ corresponds to a $D$-dimensional vector $\mathbf{x}_t = (x_t^1, \ldots, x_t^D)$. Our goal is to forecast the next $T$ time steps for each $\mathbf{x}$, denoted as $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ where $\mathbf{y}_t = (y_t^1, \ldots, y_t^D)$. The corresponding set of targets is $\mathcal{Y} = \{\mathbf{y} \mid \mathbf{y} \in \mathbb{R}^{D\times T}\}$. The problem can be formally described as learning a mapping function $\mathbb{F}_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by $\theta$ that accurately predicts future time steps based on historical observations.

**Instance Normalization** For the given data, we apply a commonly used instance normalization technique by default to both the inputs and outputs of our model. This technique is designed to mitigate distribution shifts between training and testing data (Ulyanov, Vedaldi, and Lempitsky 2016). To avoid trivializing the description, we omit this step from our model illustration.

## 3 Methodology

We construct our model with two fundamental components: frequency domain learning (FDL) and temporal domain learning (TDL), which are designed to extract prediction-relevant features from the frequency and temporal domains, respectively. We employ the Fast Fourier Transform (FFT) and its inverse (iFFT) to facilitate transformations between them. Additionally, skip connections are incorporated to enhance the model's expressiveness and improve its ability to capture complex patterns.

### 3.1 Frequency Domain Learning

In frequency domain learning, we first transform the input sequence $\mathbf{x} \in \mathbb{R}^{D\times L}$ into its frequency components using the Fourier Transform. For each dimension $d \in \{1, \ldots, D\}$, the frequency component $f_k^d$ is computed as:

$$f_k^d = \sum_{t=1}^{L} x_t^d \cdot e^{-j\frac{2\pi}{L}kt}, \quad k = 1, 2, \ldots, L$$

Using Euler's formula $e^{-j\theta} = \cos(\theta) - j\sin(\theta)$, this can be rewritten in the standard complex form $a + jb$ as:

$$f_k^d = \left(\sum_{t=1}^{L} x_t^d \cos\left(\frac{2\pi}{L}kt\right)\right) - j\left(\sum_{t=1}^{L} x_t^d \sin\left(\frac{2\pi}{L}kt\right)\right)$$

This complex form provides both amplitude and phase information for each frequency, separating the real and imaginary parts. It serves as the target representation for analysis in the frequency domain.

We employ a CVNN to operate directly on the complex-valued frequency representations for prediction. The CVNN is designed to handle both real and imaginary components jointly and learn meaningful correlations between magnitude and phase. Let the input of the CVNN be $\mathbf{z} = \mathbf{a} + j\mathbf{b} \in \mathbb{C}$, the CVNN learns a transformation $\mathcal{G} : \mathbb{C} \to \mathbb{C}$, such that $\mathbf{z}' = \mathcal{G}(\mathbf{z}) = \mathbf{c} + j\mathbf{d}$, where $\mathbf{c}$ and $\mathbf{d}$ are the learned real and imaginary components of the prediction-relevant representation. This formulation allows the model to maintain phase coherence and exploit spectral features more effectively than real-valued models. After learning, the complex representation $\mathbf{z}' = \mathbf{c} + j\mathbf{d}$ can be transformed back into the temporal domain for final prediction. Figure 2 illustrates our approach to frequency domain learning.

Considering the need to control model complexity, we construct our CVNN using standard and lightweight architectural components. We adopt a classical encoder-decoder

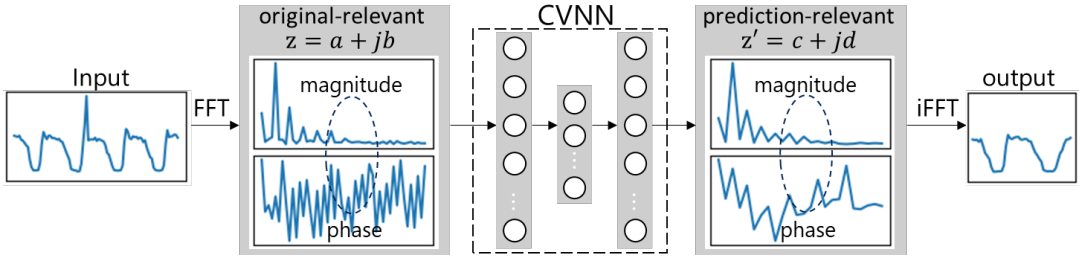



Figure 2: Learning prediction-relevant features from frequency domain.

design to preserve complex-valued features and maintain structural symmetry. The encoder consists of a complex-valued MLP, denoted as $\mathrm{CMLP}_1$, applied after the FFT, followed by a fully connected linear layer $\mathrm{CFC}_1$. This structure embeds the complex-valued input representation $z = a + jb$ into a hidden space, preserving both magnitude and phase information. The decoder mirrors the encoder with a symmetric architecture: it begins with a fully connected linear layer $\mathrm{CFC}_2$, followed by another complex-valued MLP, denoted $\mathrm{CMLP}_2$, and concludes with an inverse FFT (iFFT) to transform back into the temporal domain. We note that the last layer of $\mathrm{CMLP}_1$ (and $\mathrm{MLP}_1$) employs ReLU as its activation function to introduce non-linearity and enhance representational capacity. The complex-valued ReLU is implemented as (Trabelsi et al. 2017):

$$\mathrm{CReLU}(z) = \mathrm{ReLU}(\Re(z)) + j \cdot \mathrm{ReLU}(\Im(z))$$

To enhance information flow and model stability, we incorporate a skip connection to bridge over two fully connections $\mathrm{CFC}_1$ and $\mathrm{CFC}_2$. This allows the model to preserve low-level frequency features across the transformation pipeline. The dimensions of the hidden features between layers $\mathrm{CFC}_1$ and $\mathrm{CFC}_2$ can be adjusted as hyperparameter based on the characteristics of the input data.

After that, the learned complex representation $\mathbf{z}' = \mathbf{c} + j\mathbf{d}$ can be transformed back into the temporal domain using the iFFT. We note that this representation maintains the same dimension $D$ on the variable axis, but obtains a new length $L'$ on the time axis. The length $L'$ can either match the original input length $L$ or be set to a target length $T$, depending on the application requirements. The learned representations can be summarized as:

$$\mathbf{Y}_{\text{freq}} \in \mathbb{R}^{D \times L'}$$

and, the FDL architecture is summarized as:

$$\mathbf{X} \to \mathrm{FFT} \to \mathrm{CMLP}_1 \to \mathrm{CFC}_1 \to \mathrm{CFC}_2 \to \mathrm{CMLP}_2 \to \mathrm{iFFT} \to \mathbf{Y}_{\text{freq}}$$

## 3.2 Temporal Domain Learning

Most existing methods have focused on learning in the temporal domain. In this study, we adopt a simplified network architecture that mirrors our FDL—a real-valued version of FDL that excludes both the FFT and iFFT. The TDL architecture follows the form:

$$\mathbf{X} \to \mathrm{MLP}_1 \to \mathrm{FC}_1 \to \mathrm{FC}_2 \to \mathrm{MLP}_2 \to \mathbf{Y}_{\text{temp}}$$

Here, $\mathrm{MLP}_1$ and $\mathrm{MLP}_2$ are real-valued multilayer perceptrons, and $\mathrm{FC}_1$, $\mathrm{FC}_2$ are fully connected layers. The output $\mathbf{Y}_{\text{temp}}$ represents the prediction in the temporal domain.

## 3.3 Combination of FDL and TDL

Based on above description, we know that both the FDL and TDL components can function independently, each possessing a sound structure for prediction. FDL embodies our core idea of approaching time series prediction from a frequency-based perspective, while TDL can be regarded as a variation of linear-based methods, as discussed in prior work (Zeng et al. 2023). These two components are not only capable of operating separately but can also be integrated into a unified model to leverage the strengths of both.

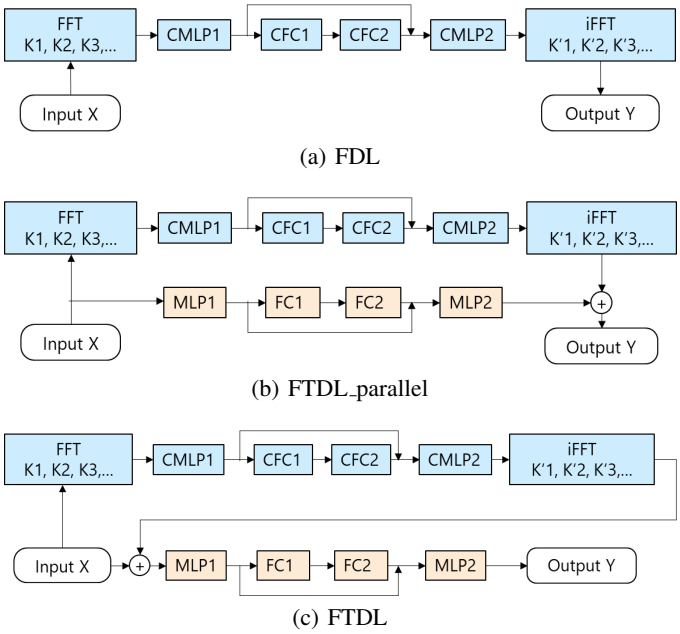



Figure 3: Three implementations of our model.

We construct and validate three implementations of our model, as shown in Figure 3. Figure 3(a) illustrates a simple model consisting solely of FDL. Figure 3(b) presents a model that combines both FDL and TDL in parallel. In this case, we explicitly set $L' = T$. We refer to this model as FTDL_parallel. Figure 3(c) shows a sequential combina-

tion of FDL and TDL. Since the output of FDL serves as the input to TDL, the representational power of TDL may be affected by distortions introduced by FDL. To mitigate this, we introduce a skip connection that directly links the original input to the output of FDL. Specifically, the output of FDL is adjusted to match the original input dimensions $L' = L$, and the two are added together to form the combined representation used as input to TDL. We refer to this architecture as FTDL.

### 3.4 Loss Functions

We chose Mean Absolute Error (MAE) as our loss function to minimize the discrepancy between the predictions $Y$ and the ground truth $\hat{Y}$. An investigation about other commonly used functions is given in Appendix C.1.

$$\mathcal{L} = \mathrm{MAE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{NTD} \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{d=1}^{D} \left| Y_{n,t,d} - \hat{Y}_{n,t,d} \right|$$

## 4 Experiments

### 4.1 Experimental Settings

**Dataset Overview** We evaluate the performance of our method on 11 datasets commonly used in recent research. These datasets are categorized into two groups based on characteristics such as baseline models and forecasting horizons. The first group consists of seven datasets: ETTh1, ETTh2, ETTm1, ETTm2, Traffic, Electricity, and Weather, used for forecasting with prediction lengths $T \in \{96, 192, 336, 720\}$. The second group includes four PeMS datasets: PEMS03, PEMS04, PEMS07, and PEMS08, used for forecasting with $T \in \{12, 24, 48, 96\}$.

For brevity, detailed descriptions of each dataset are deferred to Appendix B.1, and additional experiments on a third group of datasets are reported in Appendix D.1.

**Evaluation Protocol** We compare our method with strong baselines selected for the two dataset groups (see Appendix A for details).

For the first group, we evaluate against transformer-based models—TimeXer (Wang et al. 2025), iTransformer (Liu et al. 2023), PatchTST (Nie et al. 2023)—as well as the TCN-based ModernTCN (Luo and Wang 2024) and the frequency-based FilterNet (Yi et al. 2024). For the second group, we compare with iTransformer (Liu et al. 2023), PatchTST (Nie et al. 2023), Autoformer (Wu et al. 2021), FEDformer (Zhou et al. 2022), and the TCN-based SCINet (Liu et al. 2022). As our reproduction experiments show that many baselines perform better in their original papers, we adopt the reported results for the first group. For the second group, we use results from (Liu et al. 2023), which includes comprehensive experiments on these datasets.

Following prior work, we evaluate using MAE and MSE. Metric computation details are provided in Appendix C.2.

**Parameter Settings** Our method uses an input length of $L = 512$, with learning rates of $0.0006$ and $0.0003$ for the first and second groups, respectively. We fix the hidden feature size at $d_{ff} = 128$ across all datasets. Training runs for up to 200 epochs, with early stopping after 20 epochs without validation improvement. We use the Adam optimizer (Kingma and Ba 2014), and all experiments are conducted in PyTorch 2.6.0 with CUDA 12.4 on a single NVIDIA Quadro RTX 6000 GPU.

### 4.2 Main Experimental Results

**Results on the First Group of Datasets** Table 1 summarizes the forecasting results on the first group of datasets, comparing our methods with the baselines. As shown in the table, our methods—FDL, FTDL_parallel, and FTDL—consistently achieve strong performance across different datasets. The count row at the bottom reports the number of first- and second-best results based on MAE and MSE for each dataset. Overall, our methods achieve the majority of top-two rankings, with FTDL_parallel leading (6 best MSE and 14 best MAE, plus 8 and 8 second-best). Among our three implementations, FDL excels on ETT datasets, while FTDL_parallel dominates other datasets, slightly outperforming FTDL. Among baselines, ModernTCN is the most competitive (13 best MSE and 6 best MAE, plus 4 and 2 second-best), followed by PatchTST, while others show limited impact.

**Results on the Second Group of Datasets** Table 2 shows the forecasting results on the second group of datasets. We can see that our methods achieved almost the first- and second-best performance across four datasets. Significantly, FTDL outperformed the others and obtained the majority of the best results. The baselines iTransformer and SCINet performed reasonably well, but still lagged behind ours.

**Instance-Level Performance** We also evaluate instance-level performance by comparing forecasts with ground truth. Figure 4 compares FTDL, PatchTST, and TimeXer on the Traffic dataset for the 96-step prediction. Each row presents the waveform, power spectrum, and an error heatmap. The error heatmap visualizes frequency-wise differences between the ground truth and predicted results, highlighting where the model deviates most in the frequency domain and offering insight into spectral accuracy. From the heatmaps, the superiority of FTDL is clearly observed: the error map for FTDL remains consistently dark, indicating minimal deviation across frequencies, while PatchTST and TimeXer exhibit brighter regions that reflect larger spectral errors. Other comparisons on instances from additional datasets are provided in Appendix D.2.

### 4.3 Model Analysis

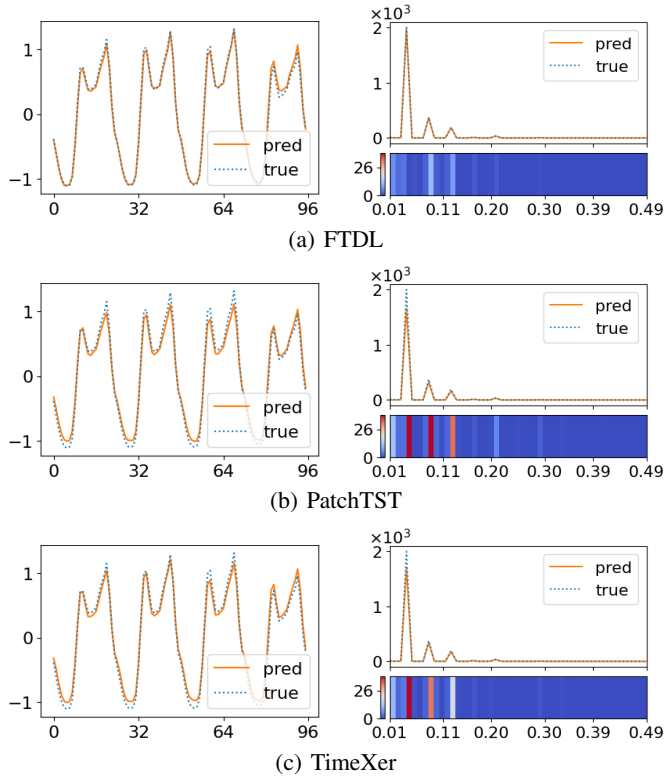
**Efficiency** We compare the efficiency under two resource consumption scenarios: a 720-step prediction on the Traffic dataset and a 96-step prediction on the PEMS07 dataset. Table 3 summarizes the results in terms of training time per epoch (in seconds), unless otherwise specified, and total parameter count. The input length is set to 512 for FTDL and ModernTCN, and to 96 for the other methods. Due to varying memory requirements, batch sizes are tuned accordingly for each method. From the results, it is evident that FTDL and iTransformer significantly outperform others in both metrics. This aligns with their architectural differences: iTransformer applies attention only to variable

| Data | T | FDL | | FTDL_parallel (Ours) | | FTDL | | TimeXer (2025) | | ModernTCN (2024) | | TexFilter (2024) | | iTransformer (2023) | | PatchTST (2023) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | 96 | **0.358** | **0.388** | 0.365 | 0.392 | 0.377 | 0.401 | 0.382 | 0.403 | 0.368 | 0.394 | 0.382 | 0.402 | 0.386 | 0.405 | 0.370 | 0.400 |
| | 192 | 0.406 | 0.426 | **0.405** | 0.418 | 0.413 | 0.42 | 0.429 | 0.435 | **0.405** | **0.413** | 0.430 | 0.429 | 0.441 | 0.436 | 0.413 | 0.429 |
| | 336 | 0.429 | 0.441 | 0.434 | 0.444 | 0.465 | 0.459 | 0.468 | 0.448 | **0.391** | **0.412** | 0.472 | 0.451 | 0.487 | 0.458 | 0.422 | 0.440 |
| | 720 | 0.454 | 0.470 | 0.454 | 0.468 | 0.490 | 0.480 | 0.469 | **0.461** | 0.450 | **0.461** | 0.481 | 0.473 | 0.503 | 0.491 | **0.447** | 0.468 |
| ETTm1 | 96 | 0.287 | **0.335** | **0.286** | **0.335** | 0.290 | 0.342 | 0.318 | 0.356 | 0.292 | 0.346 | 0.321 | 0.361 | 0.334 | 0.368 | 0.293 | 0.346 |
| | 192 | 0.331 | **0.360** | **0.330** | 0.367 | 0.331 | 0.364 | 0.362 | 0.383 | 0.332 | 0.368 | 0.367 | 0.387 | 0.377 | 0.391 | 0.333 | 0.370 |
| | 336 | 0.369 | **0.383** | 0.366 | 0.384 | 0.366 | 0.384 | 0.395 | 0.407 | **0.365** | 0.391 | 0.401 | 0.409 | 0.426 | 0.420 | 0.369 | 0.392 |
| | 720 | 0.422 | **0.411** | 0.423 | 0.414 | 0.434 | 0.420 | 0.452 | 0.441 | **0.416** | 0.417 | 0.477 | 0.448 | 0.491 | 0.459 | **0.416** | 0.420 |
| ETTh2 | 96 | 0.272 | **0.332** | 0.277 | 0.336 | 0.281 | 0.340 | 0.286 | 0.338 | **0.263** | **0.332** | 0.293 | 0.343 | 0.297 | 0.349 | 0.274 | 0.337 |
| | 192 | 0.343 | 0.378 | 0.349 | 0.382 | 0.352 | 0.385 | 0.363 | 0.389 | **0.320** | **0.374** | 0.374 | 0.396 | 0.380 | 0.400 | 0.341 | 0.382 |
| | 336 | 0.379 | 0.406 | 0.384 | 0.410 | 0.387 | 0.414 | 0.414 | 0.423 | **0.313** | **0.376** | 0.417 | 0.430 | 0.428 | 0.432 | 0.329 | 0.384 |
| | 720 | 0.405 | 0.433 | 0.411 | 0.438 | 0.415 | 0.441 | 0.408 | 0.432 | 0.392 | 0.433 | 0.449 | 0.460 | 0.427 | 0.445 | **0.379** | **0.422** |
| ETTm2 | 96 | **0.161** | **0.245** | 0.163 | 0.248 | 0.164 | 0.246 | 0.171 | 0.256 | 0.166 | 0.256 | 0.175 | 0.258 | 0.180 | 0.264 | 0.166 | 0.256 |
| | 192 | **0.218** | **0.284** | 0.222 | 0.286 | 0.220 | 0.290 | 0.237 | 0.299 | 0.222 | 0.293 | 0.240 | 0.301 | 0.250 | 0.309 | 0.223 | 0.296 |
| | 336 | 0.276 | 0.322 | **0.270** | **0.320** | 0.274 | 0.328 | 0.296 | 0.338 | 0.272 | 0.324 | 0.311 | 0.347 | 0.311 | 0.348 | 0.274 | 0.329 |
| | 720 | 0.359 | 0.377 | 0.354 | **0.376** | 0.363 | 0.381 | 0.392 | 0.394 | **0.351** | 0.381 | 0.414 | 0.405 | 0.412 | 0.407 | 0.362 | 0.385 |
| Traffic | 96 | 0.379 | 0.257 | 0.363 | **0.237** | **0.357** | 0.249 | 0.428 | 0.271 | 0.368 | 0.253 | 0.430 | 0.294 | 0.395 | 0.268 | 0.360 | 0.249 |
| | 192 | 0.396 | 0.271 | 0.385 | **0.252** | **0.379** | 0.261 | 0.448 | 0.282 | **0.379** | 0.261 | 0.452 | 0.307 | 0.417 | 0.276 | **0.379** | 0.256 |
| | 336 | 0.410 | 0.282 | 0.395 | **0.259** | 0.393 | 0.266 | 0.473 | 0.289 | 0.397 | 0.270 | 0.470 | 0.316 | 0.433 | 0.283 | **0.392** | 0.264 |
| | 720 | 0.441 | 0.298 | 0.433 | **0.280** | 0.435 | 0.287 | 0.516 | 0.307 | 0.440 | 0.296 | 0.498 | 0.323 | 0.467 | 0.302 | **0.432** | 0.286 |
| Electricity | 96 | 0.130 | 0.222 | **0.128** | **0.217** | 0.131 | 0.225 | 0.140 | 0.242 | 0.129 | 0.226 | 0.147 | 0.245 | 0.148 | 0.240 | 0.129 | 0.222 |
| | 192 | 0.148 | 0.240 | 0.148 | **0.237** | 0.151 | 0.242 | 0.157 | 0.256 | **0.143** | 0.239 | 0.160 | 0.250 | 0.162 | 0.253 | 0.147 | 0.240 |
| | 336 | 0.165 | 0.256 | 0.164 | **0.254** | 0.167 | 0.259 | 0.176 | 0.275 | **0.161** | 0.259 | 0.173 | 0.267 | 0.178 | 0.269 | 0.163 | 0.259 |
| | 720 | 0.203 | 0.290 | 0.200 | **0.283** | 0.198 | 0.286 | 0.211 | 0.306 | **0.191** | 0.286 | 0.210 | 0.309 | 0.225 | 0.317 | 0.197 | 0.290 |
| Weather | 96 | 0.150 | 0.194 | **0.144** | **0.187** | 0.148 | 0.194 | 0.157 | 0.205 | 0.149 | 0.200 | 0.162 | 0.207 | 0.174 | 0.214 | 0.149 | 0.198 |
| | 192 | **0.189** | 0.233 | 0.190 | **0.232** | 0.190 | 0.236 | 0.204 | 0.247 | 0.196 | 0.245 | 0.210 | 0.250 | 0.221 | 0.254 | 0.194 | 0.241 |
| | 336 | 0.242 | **0.273** | 0.245 | 0.276 | 0.242 | 0.276 | 0.261 | 0.290 | **0.238** | 0.277 | 0.265 | 0.290 | 0.278 | 0.296 | 0.245 | 0.282 |
| | 720 | 0.321 | 0.327 | 0.319 | **0.326** | 0.324 | 0.332 | 0.340 | 0.341 | 0.314 | 0.334 | 0.342 | 0.340 | 0.358 | 0.347 | **0.314** | 0.334 |
| Count | 1st/2nd | 4/6 | 9/8 | 6/8 | 14/8 | 2/7 | 0/8 | 0/0 | 1/1 | 13/4 | 6/2 | 0/0 | 0/0 | 0/0 | 0/0 | 6/8 | 1/8 |

Table 1: Results on the first group of datasets. The best results are in bold, and the second-best results are underlined.

| Data | T | FDL | | FTDL_parallel (Ours) | | FTDL | | iTransformer (2023) | | PatchTST (2023) | | Autoformer (2021) | | FEDformer (2022) | | SCINet (2022) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| PEMS03 | 12 | 0.058 | 0.156 | 0.058 | 0.155 | **0.057** | **0.154** | 0.071 | 0.174 | 0.099 | 0.216 | 0.272 | 0.385 | 0.126 | 0.251 | 0.066 | 0.172 |
| | 24 | 0.073 | 0.173 | 0.072 | 0.173 | **0.071** | **0.170** | 0.093 | 0.201 | 0.142 | 0.259 | 0.334 | 0.440 | 0.149 | 0.275 | 0.085 | 0.198 |
| | 48 | 0.098 | 0.195 | 0.095 | 0.194 | **0.091** | **0.190** | 0.125 | 0.236 | 0.211 | 0.319 | 1.032 | 0.782 | 0.227 | 0.348 | 0.127 | 0.238 |
| | 96 | 0.121 | 0.216 | 0.121 | 0.216 | **0.114** | **0.209** | 0.164 | 0.275 | 0.269 | 0.370 | 1.031 | 0.796 | 0.348 | 0.434 | 0.178 | 0.278 |
| PEMS04 | 12 | 0.070 | 0.169 | 0.070 | 0.167 | **0.069** | **0.166** | 0.078 | 0.183 | 0.105 | 0.224 | 0.424 | 0.491 | 0.138 | 0.262 | 0.073 | 0.177 |
| | 24 | 0.083 | 0.182 | 0.083 | 0.182 | **0.080** | **0.179** | 0.095 | 0.205 | 0.153 | 0.275 | 0.459 | 0.509 | 0.177 | 0.293 | 0.084 | 0.193 |
| | 48 | 0.099 | 0.198 | 0.101 | 0.200 | **0.095** | **0.194** | 0.120 | 0.233 | 0.229 | 0.339 | 0.646 | 0.610 | 0.270 | 0.368 | 0.099 | 0.211 |
| | 96 | 0.116 | 0.214 | 0.117 | 0.215 | **0.107** | **0.205** | 0.150 | 0.262 | 0.291 | 0.389 | 0.912 | 0.748 | 0.341 | 0.427 | 0.114 | 0.227 |
| PEMS07 | 12 | **0.051** | 0.143 | 0.053 | 0.144 | **0.051** | **0.141** | 0.067 | 0.165 | 0.095 | 0.207 | 0.199 | 0.336 | 0.109 | 0.225 | 0.068 | 0.171 |
| | 24 | 0.062 | 0.155 | 0.061 | 0.151 | **0.059** | **0.150** | 0.088 | 0.190 | 0.150 | 0.262 | 0.323 | 0.420 | 0.125 | 0.244 | 0.119 | 0.225 |
| | 48 | 0.075 | 0.167 | 0.077 | 0.168 | **0.069** | **0.163** | 0.110 | 0.215 | 0.253 | 0.340 | 0.390 | 0.470 | 0.165 | 0.288 | 0.149 | 0.237 |
| | 96 | 0.093 | 0.184 | 0.092 | 0.183 | **0.080** | **0.173** | 0.139 | 0.245 | 0.346 | 0.404 | 0.554 | 0.578 | 0.262 | 0.376 | 0.141 | 0.234 |
| PEMS08 | 12 | 0.070 | 0.168 | **0.067** | **0.159** | 0.069 | 0.161 | 0.079 | 0.182 | 0.168 | 0.232 | 0.436 | 0.485 | 0.173 | 0.273 | 0.087 | 0.184 |
| | 24 | **0.084** | 0.177 | 0.089 | 0.181 | 0.093 | **0.174** | 0.115 | 0.219 | 0.224 | 0.281 | 0.467 | 0.502 | 0.210 | 0.301 | 0.122 | 0.221 |
| | 48 | 0.131 | 0.208 | **0.128** | 0.202 | 0.132 | **0.186** | 0.186 | 0.235 | 0.321 | 0.354 | 0.966 | 0.733 | 0.320 | 0.394 | 0.189 | 0.270 |
| | 96 | 0.185 | 0.219 | **0.183** | 0.222 | 0.191 | **0.203** | 0.221 | 0.267 | 0.408 | 0.417 | 1.385 | 0.915 | 0.442 | 0.465 | 0.236 | 0.300 |
| Count | 1st/2nd | 2/8 | 0/9 | 3/10 | 1/9 | 12/1 | 15/1 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/1 | 0/0 |

Table 2: Results on the second group of datasets. The best results are in bold, and the second-best results are underlined.




(a) FTDL

(b) PatchTST

(c) TimeXer

Figure 4: Example of instance-level prediction comparison.

dimensions, enabling faster training, while PatchTST and TimeXer use patch-based attention, which increases computational cost. ModernTCN shows the worst efficiency, likely due to its expanded internal channels in intermediate layers—especially problematic for multi-channel datasets like Traffic and PEMS07. Notably, FTDL consistently achieves the lowest training time and parameter count across all scenarios, making it the most efficient model overall.

| Scenarios | 720-prediction on Traffic | | 96-prediction on PEMS07 | |
|---|---|---|---|---|
| | Time(s) | Params | Time(s) | Params |
| FTDL | 67.07 | 1,791,836 | 58.20 | 1,151,654 |
| PatchTST | 262.79 | 6,011,600 | 346.49 | 2,177,120 |
| iTransformer | 67.59 | 3,572,432 | 59.09 | 3,252,320 |
| TimeXer | 493.78 | 7,285,456 | 704.32 | 5,059,168 |
| ModernTCN | 2455.26 | 827,181,172 | 3658.13 | 861,063,556 |

Table 3: Efficiency comparison across models.

**Model Ablation Study** We conduct additional ablation experiments on 96-step prediction tasks using two representative datasets: ETTh1 and Traffic. In addition to the three previously introduced implementations, FDL, FTDL_parallel, and FTDL, we include two other variants: FTDL_sequential, which connects FDL and TDL in sequence, and TDL, which applies temporal-domain learning only.

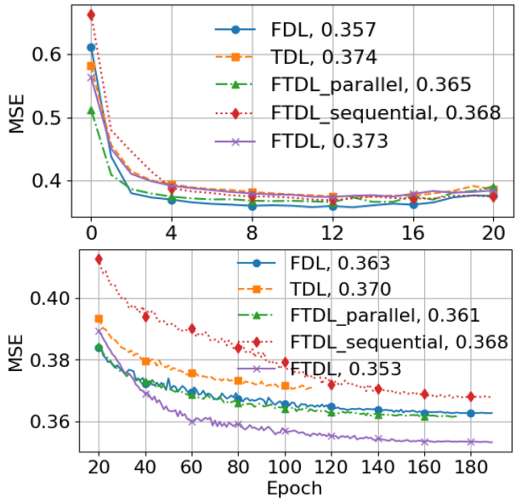
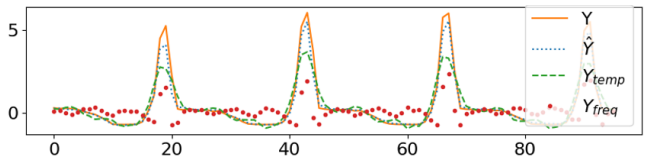



Figure 5: MSE scores throughout training. top: 96-prediction on ETTh1, bottom: 96-prediction on Traffic.

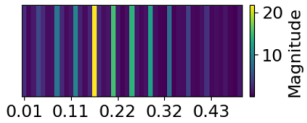
Figure 5 illustrates the MSE scores for each variation throughout training, the best scores are also indicated in the legend. FDL consistently outperforms TDL across all stages, supporting our intuitive idea that alternative representations—often overlooked in favor of purely temporal views—can reveal valuable structural patterns.

For the ETTh1 dataset, FDL achieves the best MSE of 0.357, outperforming TDL and all other variants. This suggests the stronger representational capacity of FDL compared to TDL. It may also be influenced by the characteristics of the data itself, which contains only 7 variables—making it more susceptible to overfitting when using complex models like FTDL. Training converges early around epoch 20, with steadily declining MSE. In contrast, for the Traffic dataset, which contains 862 variables, FTDL_parallel yields a slight performance improvement (MSE 0.361), while FTDL_sequential results in a minor drop (0.368) compared to FDL (0.363). Notably, FTDL significantly enhances performance, achieving the lowest MSE of 0.353. The architectural design of FTDL enables TDL to access both raw input and processed features, improving representational diversity and contributing to more stable learning.

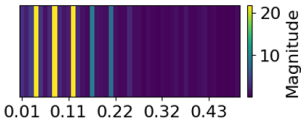
**Contributions of FDL and TDL** We identify the individual contributions of FDL and TDL by analyzing the output of FTDL_parallel, which adopts a symmetric architecture. Figure 6 shows an instance-level result from the Traffic dataset. In Row 1, the prediction $Y = Y_{\text{temp}} + Y_{\text{freq}}$ corresponds to the output of FTDL_parallel, where $Y_{\text{temp}}$ and $Y_{\text{freq}}$ are produced by TDL and FDL, respectively. Row 2 displays the heatmaps of $Y_{\text{freq}}$ and $Y_{\text{temp}}$. We observe that high-frequency components are mainly predicted by FDL, while low-frequency ones are captured by TDL. This behavior is even more evident with a synthetic toy dataset. Additional results from the toy dataset and other real-world datasets are provided in Appendix D.3.




(a) Prediction result and its components, $Y_{\text{temp}}$ and $Y_{\text{freq}}$




(b) $Y_{\text{freq}}$ heatmap

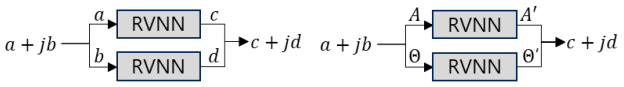



(c) $Y_{\text{temp}}$ heatmap

Figure 6: FDL vs. TDL contributions (Traffic, 96-step).

Our experimental results demonstrate that FDL and TDL cooperate effectively in modeling mixed-frequency time series. This synergy helps mitigate the spectral bias, a phenomenon where multilayer perceptrons (MLPs) tend to prioritize learning low-frequency patterns over high-frequency ones. The combination of FDL and TDL achieves better frequency specialization and improved overall performance.

**Effectiveness of CVNN for FDL** To evaluate the effectiveness of CVNN, we also implemented FDL using two alternatives: Dual Real-Valued Neural Network (Dual RVNN), which applies separate real-valued networks to the real and imaginary parts and combines their outputs; and Amplitude-Phase Transformation (A&P RVNN), which transforms the input into amplitude and phase components, $A = \text{abs}(a + jb)$ and $\Theta = \arctan(b/a)$, respectively. These components are independently processed by RVNN to produce $A'$ and $\Theta'$, which are then recombined into a complex-valued output: $c = A' \cdot \cos(\Theta')$, $d = A' \cdot \sin(\Theta')$. Figure 7 illustrates the alternative implementations.




(a) Dual RVNN (b) A&P RVNN

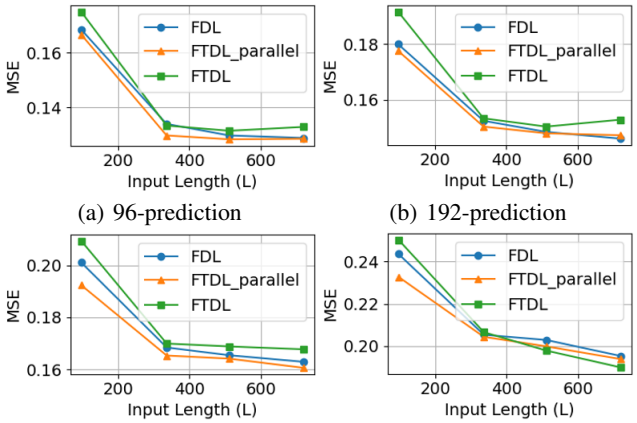
Figure 7: Alternative implementations of FDL.

The performance comparisons for 96-step prediction across three datasets are summarized in Table 4. The CVNN-based implementation consistently outperforms the two alternatives. Its superior performance can be attributed to its ability to retain complex-valued parameters, enabling direct modeling of both real and imaginary components. This facilitates richer feature representation and leads to improved predictive accuracy. Experimental results on a synthetic toy dataset, illustrating the effectiveness of CVNN and the individual roles of FDL and TDL, are provided in Appendix D.3, and the reasons for the high performance of FDL are summarized in Appendix D.7.

**Impact of Input Length** As discussed in previous works, transformer-based models often fail to benefit from a longer input length (Zeng et al. 2023; Nie et al. 2023). We conduct experiments using the Electricity dataset, which includes a moderate number of variates, to investigate the impact of input length by varying $L \in \{96, 336, 512, 720\}$ across several prediction tasks. Figure 8 shows MSE values change with the input length.

| Method | CVNN | | Dual RVNN | | A&P RVNN | |
|---|---|---|---|---|---|---|
| Data | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTm1 | 0.287 | 0.332 | 0.290 | 0.333 | 0.326 | 0.369 |
| Weather | 0.147 | 0.186 | 0.148 | 0.189 | 0.160 | 0.208 |
| Traffic | 0.377 | 0.268 | 0.382 | 0.270 | 0.420 | 0.300 |

Table 4: Performance of three implementations of FDL.




(a) 96-prediction (b) 192-prediction

(c) 336-prediction (d) 720-prediction

Figure 8: Impact of input length on MSE (Electricity).

We observe that increasing the input length generally improves performance across all three implementations. However, extending it further to 720 results in performance drops in some cases involving relatively short-term prediction with the more complex FTDL. This suggests that model complexity amplifies sensitivity to input length in shorter prediction tasks. Shorter prediction horizons tend to be more negatively affected by longer inputs, possibly due to statistical mismatches introduced by excessively long input windows. Thus, the input length should be chosen based on both the prediction horizon and the characteristics of the data.

## 5 Conclusion

In this paper, we proposed FTDL—a methodology that captures prediction-relevant features from both frequency and temporal domains. We presented three straightforward implementations using basic components. Comprehensive evaluations on synthetic and real-world datasets confirm that FTDL achieves strong and consistent performance, highlighting its potential as a practical and generalizable solution for time series forecasting. We hope this work contributes to ongoing discussions around model design in time series forecasting.

## References


Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Basri, R.; Galun, M.; Geifman, A.; Jacobs, D.; Kasten, Y.; and Kritchman, S. 2020. Frequency bias in neural networks for input of non-uniform density. In *International conference on machine learning*, 685–694. PMLR.

Bassey, J.; Qian, L.; and Li, X. 2021. A survey of complex-valued neural networks. *arXiv preprint arXiv:2101.12249*.

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

Chen, S.-A.; Li, C.-L.; Yoder, N.; Arik, S. O.; and Pfister, T. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*.

Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2306.09364*.

Eldele, E.; Ragab, M.; Chen, Z.; Wu, M.; and Li, X. 2024. Tslanet: Rethinking transformers for time series representation learning. *arXiv preprint arXiv:2404.08472*.

Fei, J.; Yi, K.; Fan, W.; Zhang, Q.; and Niu, Z. 2025. Amplifier: Bringing Attention to Neglected Low-Energy Components in Time Series Forecasting. *arXiv preprint arXiv:2501.17216*.

Gu, S.; and Ding, L. 2018. A complex-valued vgg network based deep learing algorithm for image recognition. In *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*, 340–343. IEEE.

Hammad, M. 2024. Comprehensive Survey of Complex-Valued Neural Networks: Insights into Backpropagation and Activation Functions. *arXiv preprint arXiv:2407.19258*.

Hugging Face. 2023. Yes, Transformers are Effective for Time Series Forecasting. https://huggingface.co/blog/autoformer. Accessed: 2023-09-12.

Hyndman, R. J.; and Athanasopoulos, G. 2018. *Forecasting: principles and practice*. OTexts.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lara-Ben'itez, P.; Carranza-Garcia, M.; and Riquelme, J. C. 2021. An experimental review on deep learning architectures for time series forecasting. *International journal of neural systems*, 31(03): 2130001.

Lee, H.; and Ko, S. 2024. TESTAM: A time-enhanced spatio-temporal attention model with mixture of experts. *arXiv preprint arXiv:2403.02600*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; and Xu, Q. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35: 5816–5828.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*.

Luo, D.; and Wang, X. 2024. Moderntcn: A modern pure convolution structure for general time series analysis. In *The twelfth international conference on learning representations*, 1–43.

Luo, T.; Ma, Z.; Xu, Z.-Q. J.; and Zhang, Y. 2019. Theory of the frequency principle for general deep neural networks. *arXiv preprint arXiv:1906.09235*.

Ni, Z.; Yu, H.; Liu, S.; Li, J.; and Lin, W. 2023. BasisFormer: Attention-based Time Series Forecasting with Learnable and Interpretable Basis. *arXiv preprint arXiv:2310.20496*.

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*.

Sun, Q.; Li, X.; Li, L.; Liu, X.; Liu, F.; and Jiao, L. 2019. Semi-supervised complex-valued GAN for polarimetric SAR image classification. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, 3245–3248. IEEE.

Tan, X.; Li, M.; Zhang, P.; Wu, Y.; and Song, W. 2019. Complex-valued 3-D convolutional neural network for PolSAR image classification. *IEEE Geoscience and Remote Sensing Letters*, 17(6): 1022–1026.

Tancik, M.; et al. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS*.

Trabelsi, C.; Bilaniuk, O.; Zhang, Y.; Serdyuk, D.; Subramanian, S.; Santos, J. F.; Mehri, S.; Rostamzadeh, N.; Bengio, Y.; and Pal, C. J. 2017. Deep complex networks. *arXiv preprint arXiv:1705.09792*.

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Uteuliyeva, M.; Zhumekenov, A.; Takhanov, R.; Assylbekov, Z.; Castro, A. J.; and Kabdolov, O. 2020. Fourier neural networks: A comparative study. *Intelligent Data Analysis*, 24(5): 1107–1120.

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

Wang, Y.; Wu, H.; Dong, J.; Qin, G.; Zhang, H.; Liu, Y.; Qiu, Y.; Wang, J.; and Long, M. 2025. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in Neural Information Processing Systems*, 37: 469–498.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*.

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.

Yi, K.; Fei, J.; Zhang, Q.; He, H.; Hao, S.; Lian, D.; and Fan, W. 2024. Filternet: Harnessing frequency filters for time series forecasting. *Advances in Neural Information Processing Systems*, 37: 55115–55140.

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. 35(12): 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. 27268–27286.