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

Yaling Tao, Kentaro Takagi, Takashi Watanabe, Kouta Nakata

Corporate Laboratory, Toshiba
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki-shi, Kanagawa 212-8582, Japan
{yaling.tao.h68, kentaro.takagi.k93, takashi.watanabe.r21, kouta.nakata.d99}@mail.toshiba

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.

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.

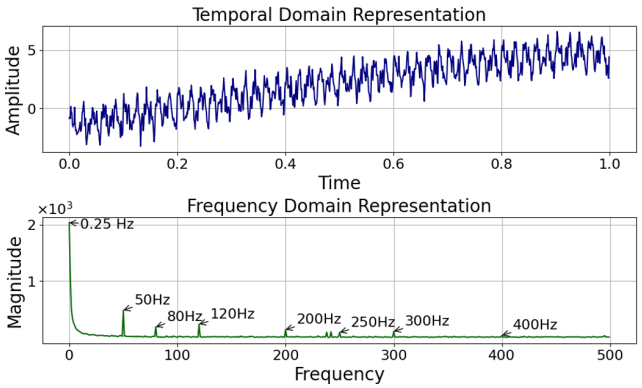


Figure 1: A synthetic time series illustrated in both the temporal and frequency domains, showing distinct patterns in each representation.

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.

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 θ 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.

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.

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

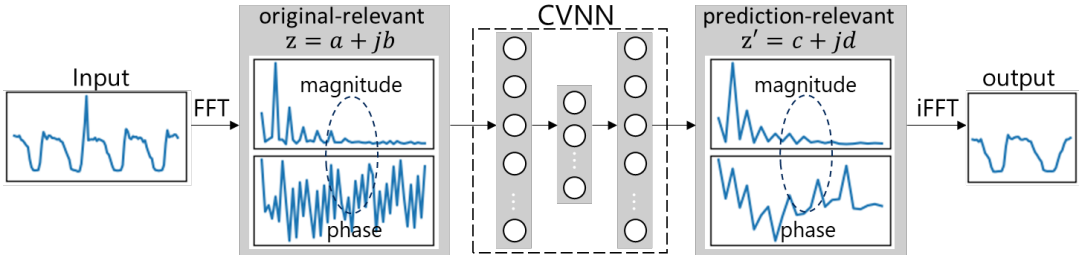


Figure 2: Learning prediction-relevant features from frequency domain.

design to preserve complex-valued features and maintain structural symmetry. The encoder consists of a complex-valued MLP, denoted as CMLP_1, applied after the FFT, followed by a fully connected linear layer 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 CFC_2, followed by another complex-valued MLP, denoted CMLP_2, and concludes with an inverse FFT (iFFT) to transform back into the temporal domain. We note that the last layer of CMLP_1 (and 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 CFC_1 and CFC_2. This allows the model to preserve low-level frequency features across the transformation pipeline. The dimensions of the hidden features between layers CFC_1 and 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:

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

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, MLP_1 and MLP_2 are real-valued multilayer perceptrons, and FC_1, FC_2 are fully connected layers. The output $\mathbf{Y}_{\text{temp}}$ represents the prediction in the temporal domain.

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.

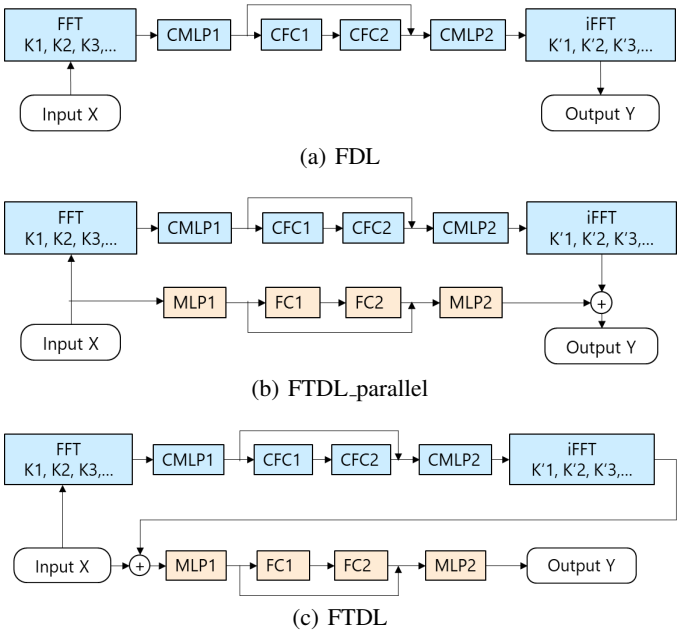


Figure 3: Three implementations of our model.

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.

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 .

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

Experiments

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 , and additional experiments on a third group of datasets are reported in Appendix .

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

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.

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 .

Model Analysis

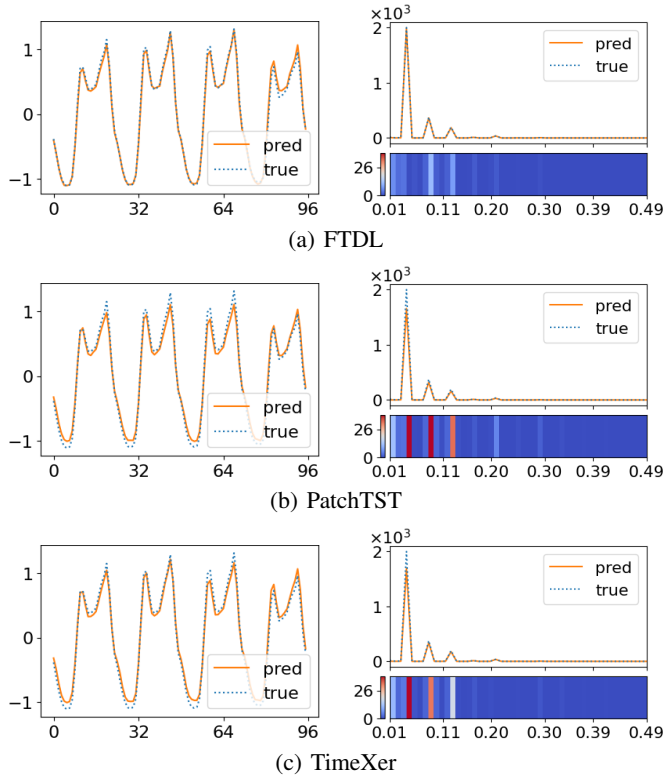
Efficiency We compare the efficiency under two resource consumption scenarios: a 720-step prediction on the Traffic dataset and a 96-step prediction on the PEMS07 dataset. Table 3 summarizes the results in terms of training time per epoch (in seconds), unless otherwise specified, and total parameter 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.

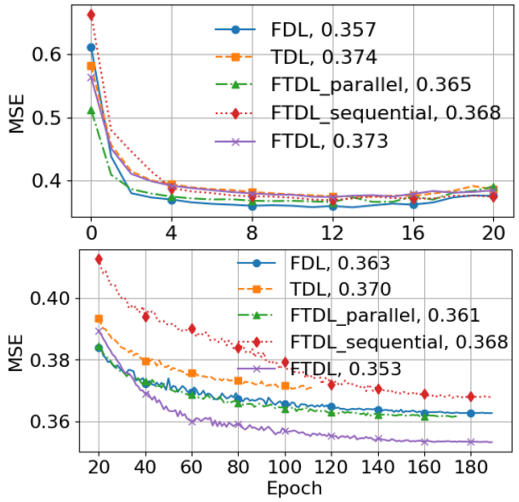
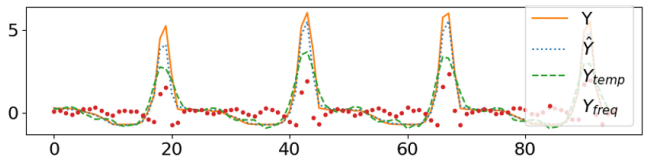


Figure 5: MSE scores throughout training. top: 96-prediction on ETTh1, bottom: 96-prediction on Traffic.

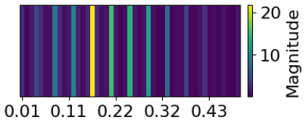
Figure 5 illustrates the MSE scores for each variation throughout training, the best scores are also indicated in the legend. FDL consistently outperforms TDL across all stages, supporting our intuitive idea that alternative representations—often overlooked in favor of purely temporal views—can reveal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

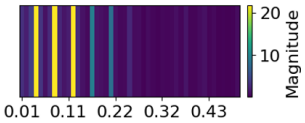
Contributions of FDL and TDL We identify the individual contributions of FDL and TDL by analyzing the output of FTDL_parallel, which adopts a symmetric architecture. Figure 6 shows an instance-level result from the Traffic dataset. In Row 1, the prediction $Y = Y_{\text{temp}} + Y_{\text{freq}}$ corresponds to the output of FTDL_parallel, where Y_{temp} and Y_{freq} are produced by TDL and FDL, respectively. Row 2 displays the heatmaps of Y_{freq} and Y_{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 .



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



(b) Y_{freq} heatmap

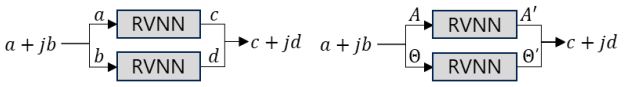


(c) Y_{temp} heatmap

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 Θ', 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

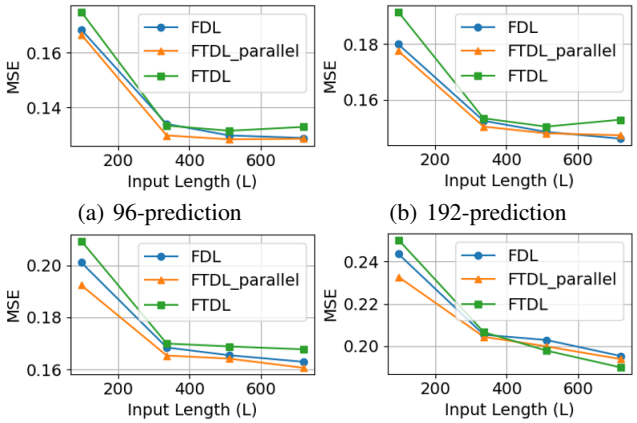
Figure 7: Alternative implementations of FDL.

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 , and the reasons for the high performance of FDL are summarized in Appendix .

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.

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.

Appendices

Related Work

Efforts on Temporal Modeling

Classical methods have traditionally assumed that time series follow predefined patterns or distributions, such as Markov Models, Bayesian Structural Time-Series Models, and ARIMA models (Lara-Ben'itez, Carranza-Garcia, and Riquelme 2021; Box et al. 2015). However, real-world time series often originate from complex, nonlinear, and non-stationary processes. To address these challenges, machine learning approaches—particularly those based on neural networks—have been proposed. These include Long Short-Term Memory (LSTM) models, Temporal Convolutional Networks (TCNs) (Bai, Kolter, and Koltun 2018), TimesNet (Wu et al. 2022), SCINet (Liu et al. 2022), ModernTCN (Luo and Wang 2024), and Transformer-based models (Vaswani et al. 2017). Many of these methods borrow techniques from other domains, treating time series data as sequential language data or one-dimensional images. By focusing on temporal modeling, they have significantly advanced the field.

SCINet (Liu et al. 2022) introduces a recursive downsampling–interaction architecture that captures multi-resolution temporal dynamics, resulting in strong forecasting performance. ModernTCN (Luo and Wang 2024) explores how to better utilize one-dimensional convolution in time series analysis. It introduces the ModernTCN block, which consists of one depthwise convolution (DWConv) layer and two pointwise convolution (PWConv) layers. The DWConv is designed to learn temporal dependencies of each univariate time series independently, while the two PWConvs are used to generate new feature representations and then integrate information across the temporal dimension.

Transformer-Based Forecasters

Transformers have recently contributed to the rapid advancement of multivariate time series forecasting. Several transformer-based models that align with open datasets have been validated in recent years (Liu et al. 2021; Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022). These methods typically embed multiple variables of the same timestamp into indistinguishable channels and apply attention mechanisms to these temporal tokens to capture temporal dependencies.

However, the trend of transformers losing to linear models in both performance and efficiency has become a hot topic (Zeng et al. 2023). This has led to increased focus on multivariate correlations and patch-based designs in follow-up models. For instance, Crossformer (Zhang and Yan 2022) partitions each series into segments, which are then embedded into 2D feature vectors. A two-stage attention mechanism is applied to capture both cross-time and cross-dimensional dependencies. PatchTST (Nie et al. 2023) divides time series into subseries-level patches, which serve as input tokens to the transformer. In contrast to channel correlation, this approach advocates channel independence, where each input token only contains information from a single univariate time series. Similarly, iTransformer (Liu et al. 2023) takes an inverted view of time series, where the entire time series of each variable is independently embedded into a token, which is then used for attention to capture multivariate correlations.

A recent work, TimeXer (Wang et al. 2025), emphasizes the importance of capturing both temporal and variate-wise dependencies within time series. It performs attention mechanisms at both patch and variate levels. Likewise, TESTAM (Lee and Ko 2024) introduces a time-enhanced spatio-temporal attention model with a mixture-of-experts architecture that separately models recurring and non-recurring patterns, combining temporal modeling with static and dynamic spatial dependencies for improved forecasting performance.

FFT Utilization for Time Series

The Fast Fourier Transform (FFT) is a commonly employed technique in time series forecasting, enabling the transformation of data from the temporal domain to the frequency domain. By decomposing time series into its frequency components, each having magnitude and phase, FFT facilitates the detection of periodic patterns and trends that may not be discernible in the original data. Recent studies have explored the application of FFT in forecasting tasks, including transformer-based models like FEDformer (Zhou et al. 2022) and Autoformer (Wu et al. 2021), as well as temporal modeling approaches like TIMESNET (Wu et al. 2022).

FEDformer (Zhou et al. 2022) enhances Transformer-based forecasting by incorporating frequency-domain attention. It applies the FFT to convert time series into the frequency domain and selects a fixed number of Fourier components to construct a sparse attention mechanism.

Autoformer (Wu et al. 2021) introduces a decomposition forecasting architecture that separates time series into trend and seasonal components. To capture periodic dependencies, it applies the Fast Fourier Transform (FFT) to the input and computes auto-correlation in the frequency domain.

TimesNet (Wu et al. 2022) transforms 1D time series into 2D tensors to capture both intra-period and inter-period variations. It applies FFT to identify dominant frequency components based on their amplitude. These selected components are then used to construct a 2D representation of the time series, where each row corresponds to a specific period or frequency band.

TSLANet (Eldele et al. 2024) is a lightweight time series model that rethinks the use of Transformers by leveraging frequency-domain filtering. It applies FFT to convert time series into the frequency domain and uses an Adaptive Spectral Block (ASB) to dynamically filter out high-frequency noise. This helps the model focus on meaningful spectral components while maintaining computational efficiency

Amplifier (Fei et al. 2025) is based on the idea that low-energy components—those with small amplitudes in the frequency domain—can carry important patterns that typical models often miss. It applies FFT to transform time series data into the frequency domain, then selectively boosts these low-energy components. This helps preserve and highlight subtle but meaningful variations during modeling.

FilterNet (Yi et al. 2024) introduces a frequency-domain approach using learnable filters to extract informative temporal patterns. Two filter types are proposed: (i) *PaiFilter*, which makes a universal frequency kernel learnable for signal filtering and temporal modeling, and (ii) *TexFilter*, which selects filtered frequencies based on compatibility with input signals for dependency learning. We adopt the stronger variant, TexFilter, as one of our baselines.

All of these approaches treat FFT primarily as a preprocessing step, where the frequency-domain representations obtained via FFT are used for feature engineering, such as extracting amplitude or other statistical features. After this, the data is typically returned to the time domain using inverse FFT (iFFT), and learning is conducted mostly in the time domain. The frequency-domain representation serves only as an intermediate aid—not a direct learning target.

In contrast, FTDL takes a fundamentally different approach. Rather than relying on handcrafted features or reverting to the time domain, FTDL directly learns from multiple frequency-domain representations. It uses FFT to obtain complex spectra—including both amplitude and phase—and processes the full spectrum using a CVNN. This enables the model to leverage low-energy components, which are often ignored in traditional methods but may carry important phase information. By learning directly in the frequency domain, FTDL captures a more complete and nuanced picture of the signal, helping the model identify which components are most relevant for accurate forecasting.

Frequency Principle (or Spectral Bias)

The Frequency Principle (F-Principle) refers to the phenomenon where deep neural networks tend to learn low-frequency components of a target function before fitting high-frequency details (Luo et al. 2019). This behavior has been observed across various architectures and tasks, and it plays a critical role in the generalization and optimization dynamics of neural networks.

Luo et al. (Luo et al. 2019) provided a theoretical foundation for the F-Principle in general deep neural networks, explaining how overparameterized models tend to fit target functions with smooth approximations first, gradually incorporating high-frequency components. This behavior helps mitigate overfitting and contributes to stable training.

Tancik et al. (Tancik et al. 2020) introduced Fourier Features to help networks learn high-frequency functions in low-dimensional domains. Basri et al. (Basri et al. 2020) showed that frequency bias persists even under non-uniform input distributions, suggesting it is intrinsic to the optimization process and can act as implicit regularization.

Despite these advances, fully understanding or effectively leveraging the F-Principle remains a subject of ongoing investigation. Our experimental results with FTDL—showing that frequency-domain learning tends to capture high-frequency components, while temporal-domain learning focuses on low-frequency patterns—offer preliminary indications that it may be relevant to this line of research.

Complex-Valued Neural Network

Complex-valued neural networks (CVNN) process information using complex-valued parameters and variables, allowing them to handle both real and imaginary components effectively. In many application domains, complex numbers arise naturally, and the correlation between the real and imaginary parts is meaningful. By working within the complex domain, CVNN can better capture the relationships between the real and imaginary components and more accurately model phase information (Bassey, Qian, and Li 2021; Hammad 2024). This is particularly relevant in time-series forecasting where data often exhibit periodicity and frequency-domain characteristics. However, research on applying complex-valued neural networks to time-series forecasting is relatively scarce, and it is less common compared to their application in image processing (Gu and Ding 2018; Sun et al. 2019; Tan et al. 2019).

Experimental Details

Data Description

The statistics of the three groups of datasets are summarized in Table 5.

The seven datasets in the first group G1 are used in most recent research involving long-term multivariate time series forecasting. **ETT** (Electricity Transformer Temperature) are collected from two different electric transformers labeled 1 and 2. Each transformer contains data at two different resolutions: 15 minutes (denoted as m) and 1 hour (denoted as h). **Traffic** records road occupancy rates from various sensors on San Francisco freeways. **Electricity** consists of hourly electricity consumption data from 321 customers. **Weather** includes 21 meteorological indicators from Germany, such as humidity and air temperature. These datasets have been extensively used for benchmarking and are publicly available at (Wu et al. 2021).

The second group G2 contains the public traffic network data from Caltrans Performance Measurement System (PeMS). We use the same four subsets, **PEMS03**, **PEMS04**, **PEMS07** and **PEMS08**, adopted in iTransformer(Liu et al. 2023) and SCINet (Liu et al. 2022). These datasets are respectively constructed from four districts in California.

The third group G3 contains other three traffic datasets from three different public traffic networks. **METR-LA** and **PEMS-BAY** contain four-month speed data recorded by 207 sensors on Los Angeles highways and 325 sensors on Bay Area, respectively (Li et al. 2017). **EXPY-TKY** consists of three-month speed data collected from 1843 links in Tokyo. We process and partition each dataset as in TESTAM (Lee and Ko 2024).

Dataset Creation

Unlike image or text datasets, raw time series data requires preprocessing before it can be used for deep learning. For multivariate long-term forecasting, each dataset was split into training, validation, and test sets based on fixed proportions of the total sequence length. To standardize the data, we computed the mean and standard deviation from the training set and applied these statistics to scale the entire

Group	Dataset	Variables	Time Steps	Interval	Start Time	Partition
G1	ETTh1	7	17,420	1 hour	2016-07-01	7/1/2
	ETTm1	7	69,680	15 min	2016-07-01	7/1/2
	ETTh2	7	17,420	1 hour	2016-07-01	7/1/2
	ETTm2	7	69,680	15 min	2016-07-01	7/1/2
	Traffic	862	17,544	1 hour	2016-07-01	7/1/2
	Electricity	321	26,304	1 hour	2016-07-01	7/1/2
	Weather	21	52,696	10 min	2020-01-01	7/1/2
G2	PEMS03	358	26,209	5 min	2012-05-01	6/2/2
	PEMS04	307	16,992	5 min	2017-07-01	6/2/2
	PEMS07	883	28,224	5 min	2017-05-01	6/2/2
	PEMS08	170	17,856	5 min	2012-03-01	6/2/2
G3	METR-LA	207	34,272	5 min	2012-03-01	7/1/2
	PEMS-BAY	325	52,116	5 min	2017-01-01	7/1/2
	EXPY-TKY	1843	13,248	10 min	2021-10-01	Train/Val: 2021-10,11 Test: 2021-12

Table 5: Statistics of the three groups of datasets. G1: datasets for long-term forecasting with $T \in \{96, 192, 336, 720\}$. G2: the PeMS datasets for forecasting with $T \in \{12, 24, 48, 96\}$. G3: traffic datasets for 15-min, 30-min, 60-min forecasting.

dataset, including the validation and test sets. After preprocessing, data samples with specified input and output lengths were generated from each set in the form of $(X, \hat{Y})$. Due to the nature of time-series splits and the specified input lengths, the samples used for training and validation may vary slightly. However, the test set remains consistent across all baselines to ensure fair evaluation. Figure 9 shows an example of how data samples are created for the ETTh1 dataset.

Implementation Details

We conduct experiments on three implementations: FDL, FTDL_parallel and FTDL, as illustrated in Figure 3. Each variant shares the same core architecture but differs in combination of FDL and TDL. The core component CMLP consists of a single complex-valued linear layer followed by a complex-valued ReLU activation. These blocks are stacked within the model to process frequency-domain representations. Similarly, MLP consists of a single linear layer followed by a ReLU activation within the model to process temporal-domain representations.

Figure 11 shows an example of the FTDL_parallel network architecture for the 96-step prediction task on the Electricity dataset. This example illustrates the entire model learning process. In this configuration, the input length is set to $L = 512$, the batch size to 128, the hidden feature dimension to $d_{ff} = 128$, and the seed value to 2024. For this experiment, the input has the format $(B, L, D) = (128, 512, 321)$, where B denotes the batch size, L the input length, and D the number of time-series variables. As shown in Figure 11, the model consists of:

- **Normalization layers** (*RevIN*), described in Section ,
- **FDL** and **TDL**, represented as *FreqEmbedding_fft_time* and *TimeEmbedding* in the figure.

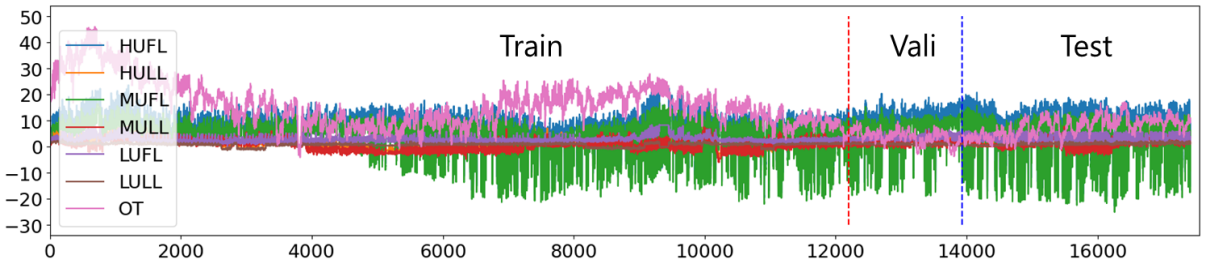
As described for the model in Figure 3, the FDL is composed of CMLP layers (CMLP1 and CMLP2) and complex-valued fully connected layers (CFC1 and CFC2), while the TDL, in a symmetric manner, consists of MLP layers (MLP1 and MLP2) and fully connected layers (FC1 and FC2). After normalization, in the FDL, inputs of size $(128, 512, 321)$ are transformed by applying FFT along the time axis, resulting in complex-valued representations of the same size $(128, 512, 321)$. These representations are then reshaped to $(128, 321, 512)$, and all subsequent operations are performed along the time axis. The dimensions are embedded to 512 by CMLP1, reduced to 128 by CFC1, expanded again to 512 by CFC2, and finally projected to an output length of 96 by CMLP2. In the TDL, the processing steps are similar to those in the FDL, except that FFT is not applied, so the data remain real-valued. Finally, the outputs of the FDL and TDL, both of size $(128, 321, 96)$, are summed to produce the final prediction results and then reshaped to $(128, 96, 321)$ as the model output.

The overall model has approximately 887,874 trainable parameters, with an estimated computational cost of 163.53M multiply-add operations and a total memory footprint of about 1.52 GB during training. We set the learning rate to $lr = 0.0006$ and trained the model for 200 epochs, applying early stopping if the loss did not decrease for 20 consecutive epochs. The learning process is illustrated in Figure 10. Training converged after 195 epochs in approximately 2.34 hours (about 43 seconds per epoch), achieving a test performance of MSE = 0.1275 and MAE = 0.2173 at epoch 170.

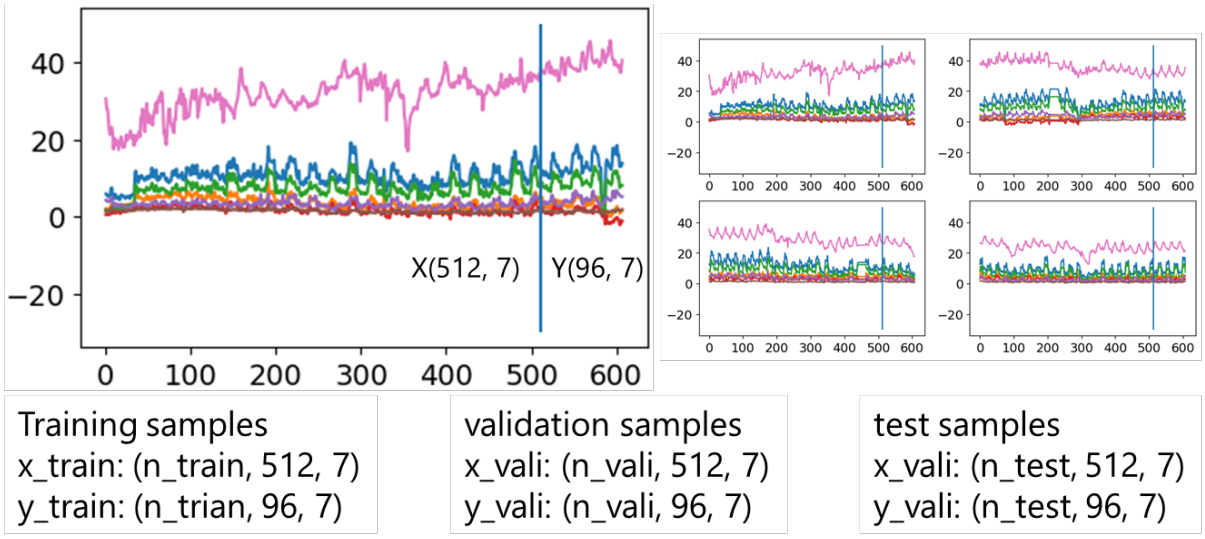
Loss Functions and Evaluation Metrics

Loss Function Selection

We empirically compared Mean Absolute Error (MAE), Mean Squared Error (MSE), and Huber loss, and found that MAE performs slightly better on data with frequent peaks, while MSE is more suited for seasonal patterns. Huber loss offered a balanced alternative between the two. Since the



(a) Data splitting



(b) Data samples

Figure 9: An example of how data samples are created for the ETTh1 dataset. The number of samples used for training and validation may vary slightly depending on the input length settings. However, the test set remains consistent across all baselines to ensure fair evaluation.

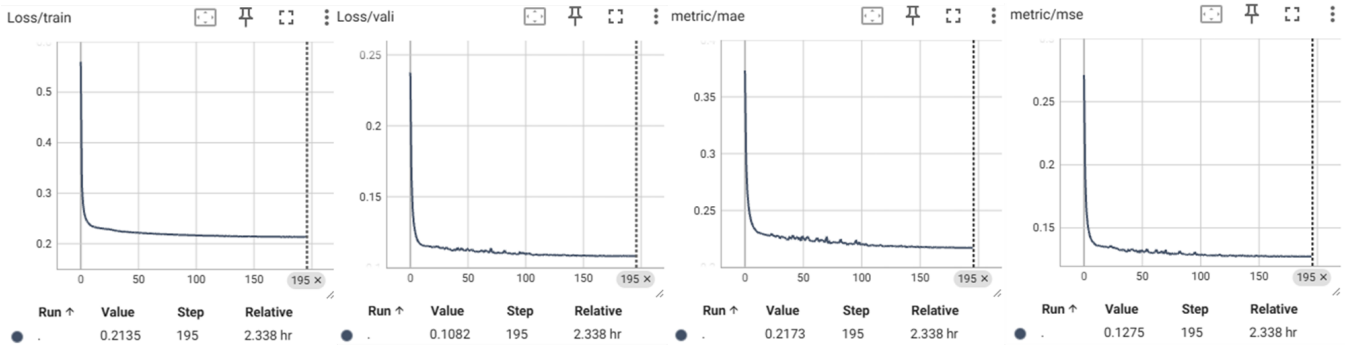
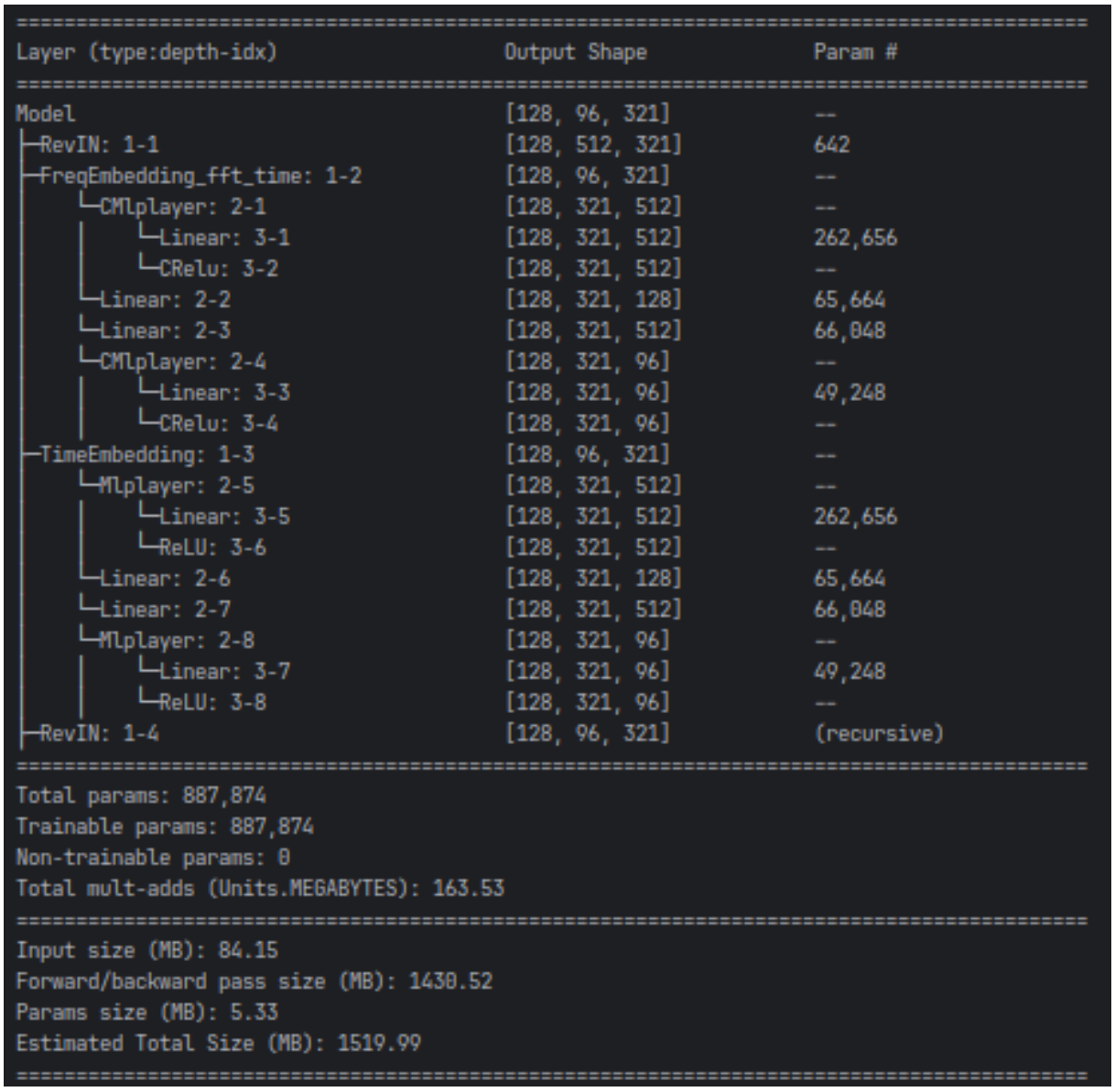


Figure 10: Learning curves of the FTDL_parallel architecture for the 96-step prediction task on the Electricity dataset, showing training and validation loss, as well as test metrics (MAE and MSE) over epochs.



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Model                                    [128, 96, 321]            --
├─RevIN: 1-1                             [128, 512, 321]           642
├─FreqEmbedding_fft_time: 1-2            [128, 96, 321]            --
│    └─CMlplayer: 2-1                    [128, 321, 512]           --
│    │    └─Linear: 3-1                  [128, 321, 512]           262,656
│    │    └─CRelu: 3-2                   [128, 321, 512]           --
│    └─Linear: 2-2                       [128, 321, 128]           65,664
│    └─Linear: 2-3                       [128, 321, 512]           66,048
│    └─CMlplayer: 2-4                    [128, 321, 96]            --
│    │    └─Linear: 3-3                  [128, 321, 96]            49,248
│    │    └─CRelu: 3-4                   [128, 321, 96]            --
├─TimeEmbedding: 1-3                     [128, 96, 321]            --
│    └─Mlplayer: 2-5                     [128, 321, 512]           --
│    │    └─Linear: 3-5                  [128, 321, 512]           262,656
│    │    └─ReLU: 3-6                    [128, 321, 512]           --
│    └─Linear: 2-6                       [128, 321, 128]           65,664
│    └─Linear: 2-7                       [128, 321, 512]           66,048
│    └─Mlplayer: 2-8                     [128, 321, 96]            --
│    │    └─Linear: 3-7                  [128, 321, 96]            49,248
│    │    └─ReLU: 3-8                    [128, 321, 96]            --
├─RevIN: 1-4                             [128, 96, 321]            (recursive)
==========================================================================================
Total params: 887,874
Trainable params: 887,874
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 163.53
==========================================================================================
Input size (MB): 84.15
Forward/backward pass size (MB): 1430.52
Params size (MB): 5.33
Estimated Total Size (MB): 1519.99
==========================================================================================
```

Figure 11: An example of the FTDL_parallel architecture for the 96-step prediction task on the Electricity dataset. *RevIN*: normalization layer; *FreqEmbedding_fft_time*: FDL component, where FFT transformation along the time axis is applied first, followed by complex-valued CMLP layers and linear layers; *TimeEmbedding*: TDL component, structured symmetrically to the FDL without FFT.

overall performance differences were minor, we selected MAE for its simplicity and consistent results across our datasets. Three loss functions are summarized as follows.

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

$$\mathrm{MSE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{NTD} \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{d=1}^{D} \left(Y_{n,t,d} - \hat{Y}_{n,t,d} \right)^2$$

$$\mathrm{Huber}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{NTD} \sum_{n,t,d} h(Y_{n,t,d} - \hat{Y}_{n,t,d}),$$

$$\text{where } h(r) = \begin{cases} \frac{1}{2} r^2 & \text{if } |r| \leq \delta \\ \delta \left(|r| - \frac{1}{2} \delta \right) & \text{otherwise} \end{cases}$$

Scenarios	Loss	Metrics	
		MSE	MAE
FDL on ETTh1 for 96-prediction	MSE	0.3665	0.3985
	MAE	0.3578	0.3879
	Huber	0.3602	0.3918
FTDL on Traffic for 96-prediction	MSE	0.3550	0.2589
	MAE	0.3573	0.2491
	Huber	0.3568	0.2561

Table 6: Performances with three loss functions: MSE, MAE, and Huber

Table 6 presents two scenarios comparing performances on representative datasets: ETTh1 and Traffic. In our experiments, the threshold parameter δ for the Huber loss is set to 1. From the results, we observe that MAE yields the best performance on ETTh1, which contains frequent peaks, while MSE performs best on Traffic, which exhibits strong seasonal patterns. Huber loss consistently provides intermediate results, making it a robust fallback option.

Forecasting Evaluation Metrics

Following previous works, we use MAE and MSE as evaluation metrics for the first and second groups. These are computed on normalized data. For the third group, we adopt the same metrics along with TESTAM (Lee and Ko 2024), MAE, RMSE, and MAPE, all calculated on the original scale after inverse transformation. The formulations for RMSE and MAPE metrics are defined as follows:

$$\mathrm{RMSE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \sqrt{\frac{1}{NTD} \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{d=1}^{D} \left(Y_{n,t,d} - \hat{Y}_{n,t,d} \right)^2}$$

$$\mathrm{MAPE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{100}{NTD} \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{d=1}^{D} \left| \frac{Y_{n,t,d} - \hat{Y}_{n,t,d}}{Y_{n,t,d}} \right|$$

Extra experimental results

Results on the Third Group of Datasets

Table 7 shows forecasting results on the third group datasets. In contrast to the results observed on the previous two groups, the performance of the models on these datasets varies significantly, suggesting that each method may be specialized for certain data characteristics. For example, the baseline TESTAM performs very well on the METR-LA data for all predictions, while our methods, along with the baseline PatchTST, show poor performance on it. For the PEMS-BAY data, FTDL outperforms the others, achieving all the best predictions, and FTDL_parallel obtains all the second-best predictions. For the EXPY-TKY data, FTDL achieves the majority of the best results, and FDL, FTDL_parallel, and PatchTST also show good performance in some cases. The baseline TESTAM, which performs well on all predictions for the METR-LA data, lags behind on all predictions for both PEMS-BAY and EXPY-TKY datasets.

Additional Instance-Level Prediction Results

Besides the results shown in Section , we present additional instance-level prediction comparisons. Figure 13 shows another example on the Traffic dataset for 96-step prediction. Figures 14 and 15 illustrate two more instances on the ETTh1 dataset, also for 96-step prediction.

Figures 16 and 18 compare predicted values with ground truth for several instances from the first group (ETTh1, Traffic, Electricity, Weather) and the second group (PEMS03, PEMS04, PEMS07, PEMS08) datasets, respectively. For the first group, 96-step prediction results of the model with the best MSE score from Table 1 are visualized; for the second group, 48-step prediction results of the best-performing model, FTDL, are visualized. These results indicate that the models generally capture overall trends, though finer fluctuations and sharp transitions remain challenging to predict.

Contribution of FDL and TDL Components

We conducted extraction experiments to verify the individual contributions of the FDL and TDL components within FTDL_parallel across multiple datasets. Figure 17 presents an instance-level result from a toy dataset designed to simulate time series containing both low- and high-frequency components. These results are consistent with the behavior observed in the Traffic dataset (Figure 6 in Section), reaffirming that FDL is responsible for capturing high-frequency components, while TDL focuses on low-frequency structures.

Figures 19, 20, 21, 22, 23, 24, and 25 present additional instance-level results from real-world datasets: Traffic, Electricity, PEMS03, and PEMS08. In the Traffic and Electricity datasets, we observe behavior consistent with Figure 6, where the data contain both low- and high-frequency components. In these cases, FDL effectively captures the high-frequency signals, while TDL focuses on the low-frequency trends. In contrast, certain instances from PEMS03 (Figure 23) and PEMS08 (Figure 25) exhibit different heatmap patterns: Y_{temp} is prominently activated, whereas Y_{freq} shows little to no response. This is expected, as the underlying

Data	T	FDL			FTDL_parallel			FTDL			PatchTST			TESTAM		
	(min)	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR -LA	5	3.15	6.83	7.62	3.17	6.87	7.66	3.12	6.88	7.59	3.11	6.89	7.54	**2.54**	**4.93**	**6.42**
	30	3.67	8.14	9.29	3.78	8.22	9.42	3.67	8.03	9.33	3.76	8.29	9.58	**2.96**	**6.04**	**7.92**
	60	4.42	9.47	11.64	4.37	9.61	11.27	4.37	9.51	11.28	4.46	10.12	11.46	**3.36**	**7.09**	**9.67**
PEMS -BAY	5	1.18	2.42	2.38	1.16	2.39	2.34	**1.12**	**2.32**	**2.29**	1.17	2.39	2.45	1.29	2.77	2.61
	30	1.42	3.16	3.12	1.39	3.12	3.00	**1.35**	**3.00**	**2.92**	1.45	3.16	3.24	1.59	3.65	3.56
	60	1.73	3.98	3.96	1.70	3.92	3.93	**1.61**	**3.71**	**3.76**	1.73	3.94	3.99	1.85	4.33	4.31
EXPY -TKY	5	5.35	8.66	22.03	5.28	8.62	**21.48**	**5.27**	**8.56**	22.07	5.30	8.62	22.38	5.84	9.23	25.36
	30	5.65	9.30	23.94	**5.60**	9.24	23.97	5.62	**9.20**	24.13	5.65	9.30	**23.75**	6.42	10.24	28.90
	60	6.00	9.87	26.29	5.98	9.80	**25.76**	5.99	**9.70**	26.69	**5.95**	9.82	26.25	6.75	10.24	31.01
Count 1st/2nd		0/5			3/16			13/5			2/5			9/0		

Table 7: Results on the third group of datasets. The best results are in bold, and the second-best results are underlined.

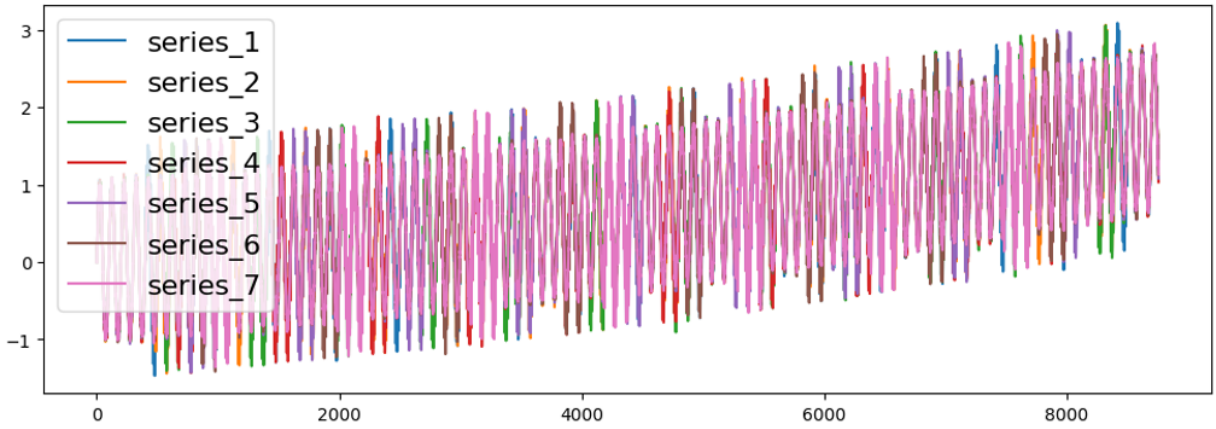


Figure 12: Synthetic datasets designed to simulate time series.

Scenarios	L	FDL			FTDL_parallel			FTDL		
		MSE	MAE	Time(s)	MSE	MAE	Time(s)	MSE	MAE	Time(s)
96-prediction	96	0.1684	0.2544	6.40	0.1666	0.2476	7.05	0.1749	0.2618	6.49
	336	0.1340	0.2283	14.31	0.1298	0.2176	15.22	0.1334	0.2270	16.40
	512	0.1298	0.2228	20.04	0.1284	0.2167	22.74	0.1315	0.2253	23.10
	720	0.1289	0.2208	29.26	0.1286	0.2181	29.60	0.1329	0.2235	32.53
192-prediction	96	0.1800	0.2653	8.51	0.1776	0.2584	9.95	0.1917	0.2746	8.62
	336	0.1524	0.2442	16.32	0.1503	0.2381	17.28	0.1533	0.2452	18.08
	512	0.1484	0.2402	22.12	0.1479	0.2368	24.08	0.1503	0.2422	25.02
	720	0.1460	0.2373	34.34	0.1472	0.2365	33.13	0.1528	0.2481	34.43
336-prediction	96	0.2010	0.2855	14.10	0.1923	0.2735	15.37	0.2093	0.2912	13.76
	336	0.1684	0.2595	22.98	0.1653	0.2520	23.37	0.1699	0.2613	23.47
	512	0.1654	0.2536	29.13	0.1641	0.2541	29.38	0.1688	0.2586	30.19
	720	0.1629	0.2523	36.37	0.1606	0.2516	39.40	0.1677	0.2589	40.65
720-prediction	96	0.2435	0.3190	24.65	0.2327	0.3064	26.07	0.2502	0.3242	25.01
	336	0.2054	0.2908	33.09	0.2044	0.2858	34.00	0.2066	0.2930	34.94
	512	0.2029	0.2895	39.32	0.1999	0.2830	39.62	0.1979	0.2862	40.90
	720	0.1954	0.2823	45.91	0.1939	0.2822	49.90	0.1900	0.2778	48.25

Table 8: Impact of input length: examples from predictions on the Electricity dataset

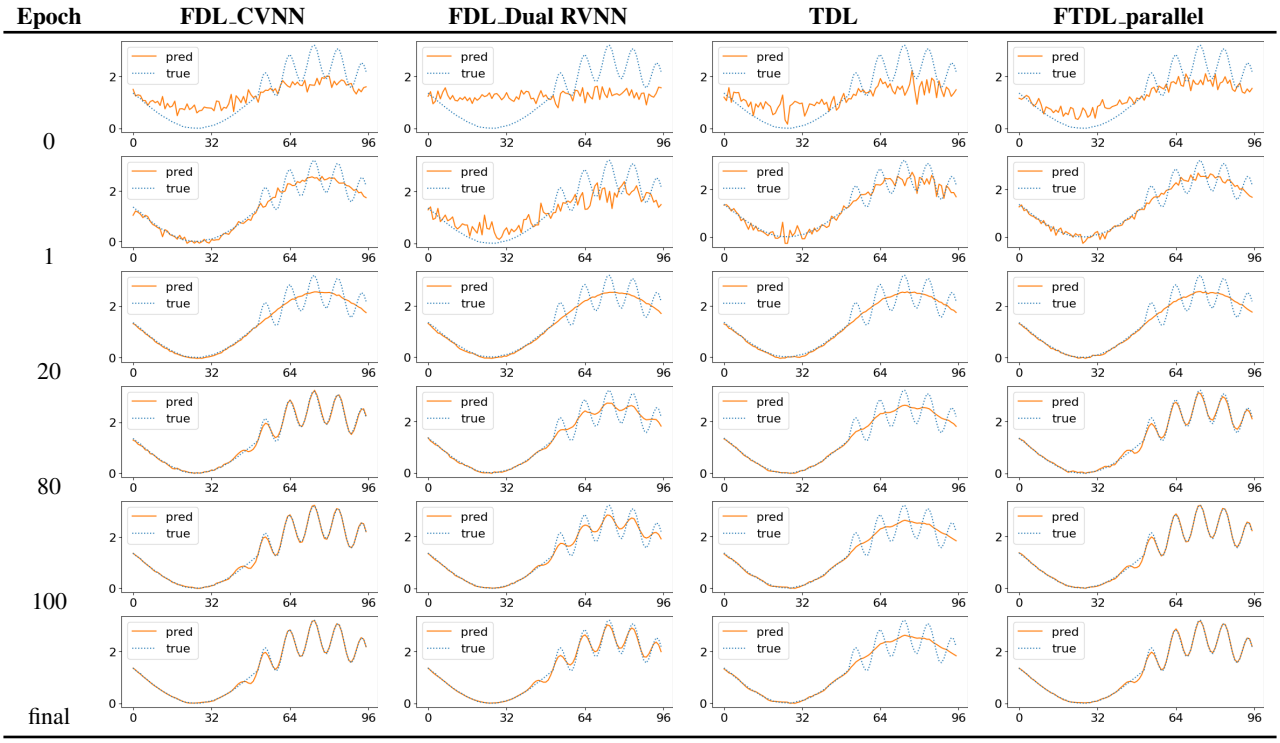


Table 9: Comparison of learning progress across models on Toy dataset

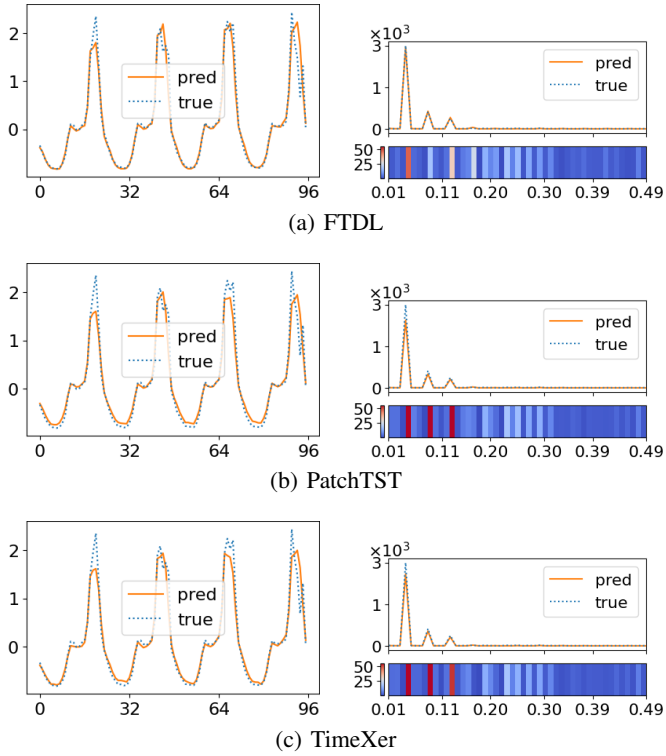


Figure 13: Another example of instance-level prediction comparison (Traffic, 96-step).

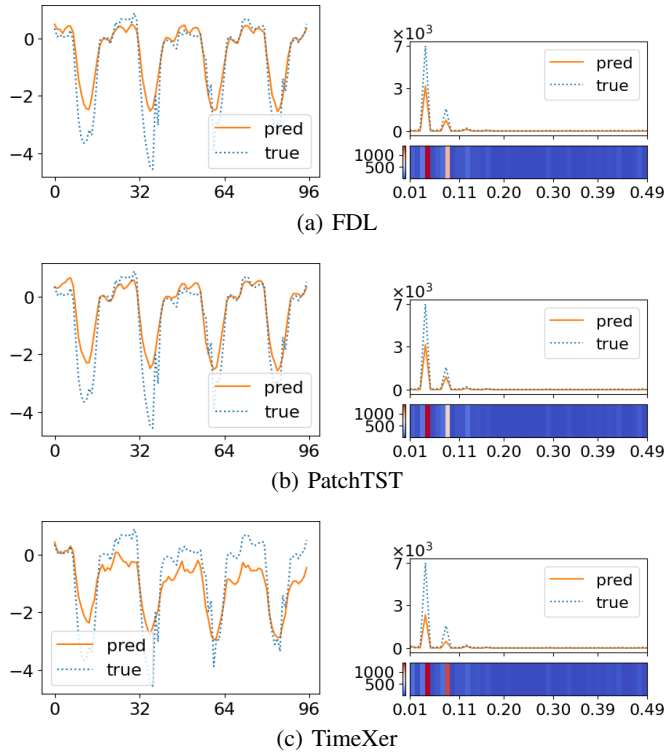
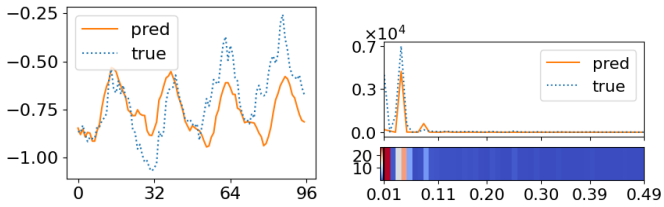
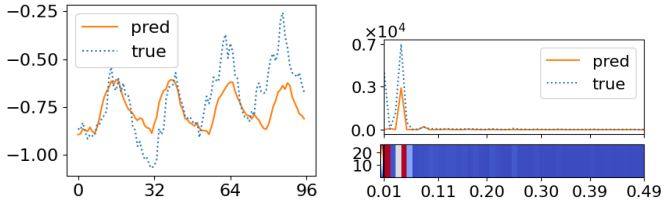


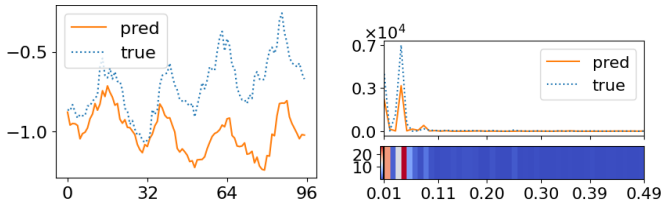
Figure 14: An example of instance-level prediction comparison (ETTh1, 96-step).



(a) FDL



(b) PatchTST



(c) TimeXer

Figure 15: Another example of instance-level prediction comparison (ETTh1, 96-step).

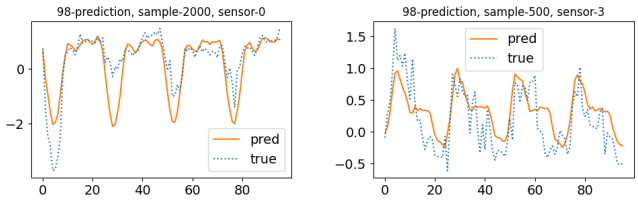
waveforms in these cases lack clear periodicity and instead exhibit dominant trend-like behavior.

Details of Impact of Input Length

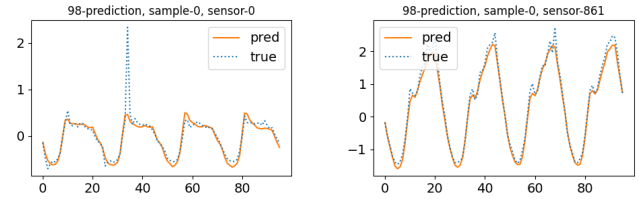
We investigate the performance of our methods with respect to this issue through comprehensive experiments. We select the Electricity dataset, which contains a moderate number of multivariate features, as the target data. The experiments are conducted by setting the input length $L \in \{96, 336, 512, 720\}$ for several long-term prediction horizons $T \in \{96, 192, 336, 720\}$. We record performance metrics, including MAE and MSE, as well as the time in seconds taken per epoch. Results are presented in Table 8.

Our results show that a longer look-back window, ranging from 96 to 512, consistently improves forecasting performance across all three implementations of our method. However, when the look-back window is further increased to 720, performance deteriorates in several cases, as indicated underlined in Table 8. Among these cases, out of a total of 4 instances, the relatively complex implementation FTDL accounts for 3, while FTDL_parallel accounts for 1. This suggests that more complex models are more prone to this issue.

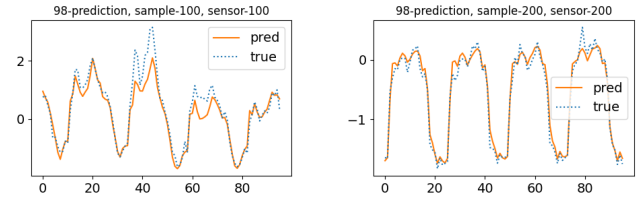
Looking at these cases, we observe that as the input length increases further, shorter prediction horizons tend to deteriorate more easily. Given that the interval of the Electricity data is 1 hour, a 96-hour prediction task with a look-back window of 720 means using data from the past month to pre-



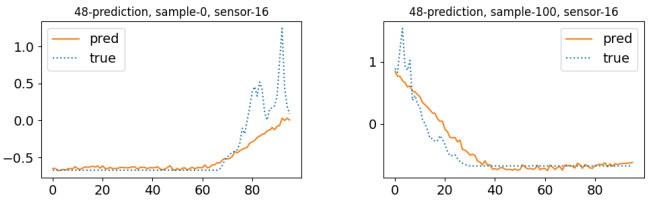
(a) ETTh1 Instances



(b) Traffic Instances

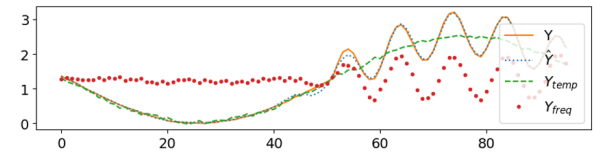


(c) Electricity Instances

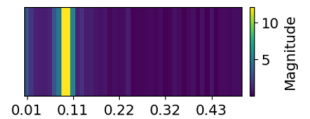


(d) Weather Instances

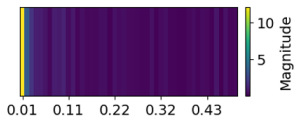
Figure 16: Examples of 96-step-ahead predictions on ETTh1, Traffic, Electricity, and Weather datasets.



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

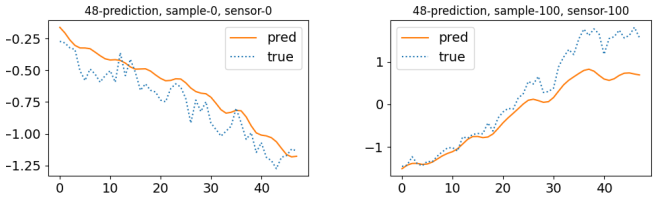


(b) Y_{freq} heatmap

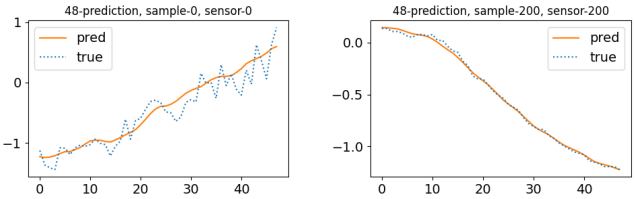


(c) Y_{temp} heatmap

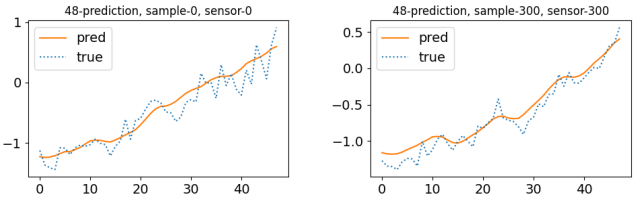
Figure 17: FDL vs. TDL contributions (An example from the Toy dataset, 96-step prediction).



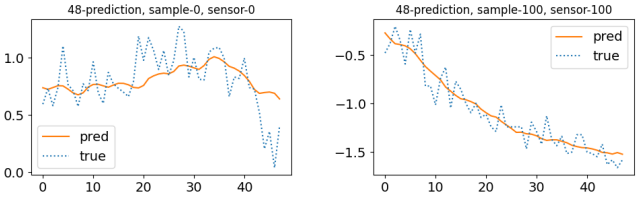
(a) PEMS03 Instances



(b) PEMS04 Instances

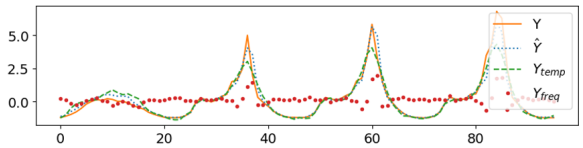


(c) PEMS07 Instances

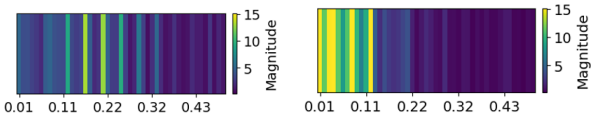


(d) PEMS08 Instances

Figure 18: Examples of 48-step-ahead predictions on PEMS03, PEMS04, PEMS07, and PEMS08 datasets.

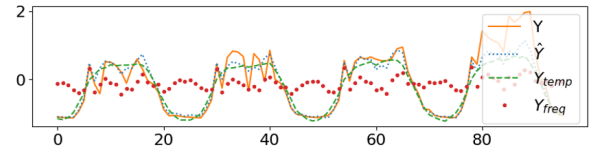


(a) Prediction result and its components, Y_{temp} and Y_{freq}

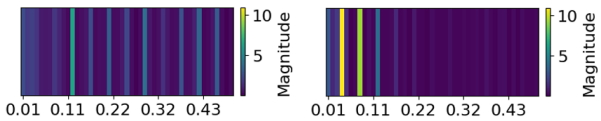


(b) Y_{freq} heatmap (c) Y_{temp} heatmap

Figure 19: FDL vs. TDL contributions (Another example from the Traffic dataset, 96-step prediction).

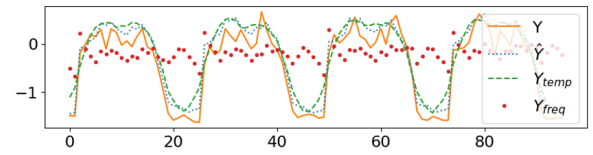


(a) Prediction result and its components, Y_{temp} and Y_{freq}

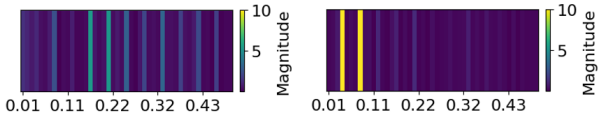


(b) Y_{freq} heatmap (c) Y_{temp} heatmap

Figure 20: FDL vs. TDL contributions (An example from the Electricity dataset, 96-step prediction).

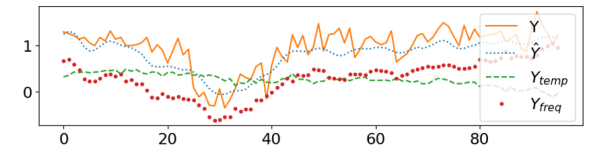


(a) Prediction result and its components, Y_{temp} and Y_{freq}

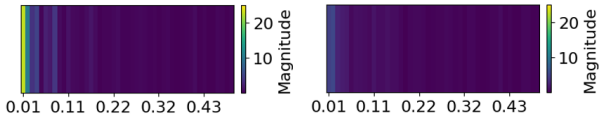


(b) Y_{freq} heatmap (c) Y_{temp} heatmap

Figure 21: FDL vs. TDL contributions (Another example from the Electricity dataset, 96-step prediction).

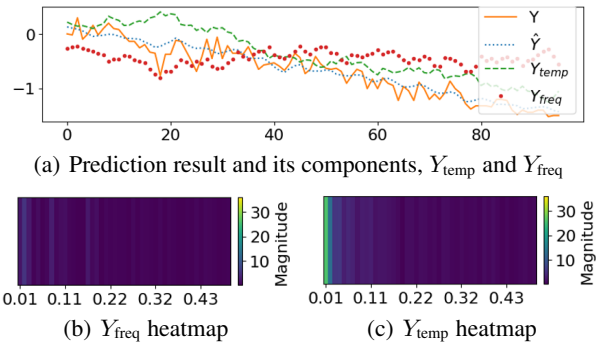


(a) Prediction result and its components, Y_{temp} and Y_{freq}



(b) Y_{freq} heatmap (c) Y_{temp} heatmap

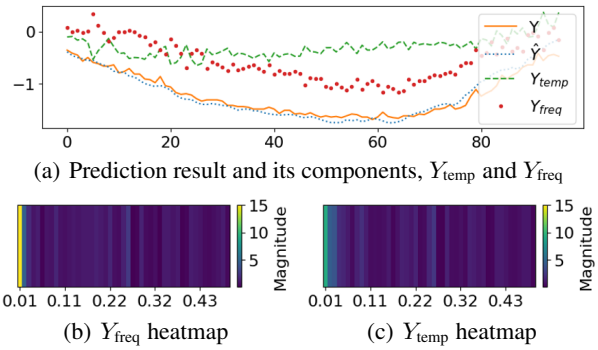
Figure 22: FDL vs. TDL contributions (An example from the PEMS03 dataset, 96-step prediction).



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

(b) Y_{freq} heatmap (c) Y_{temp} heatmap

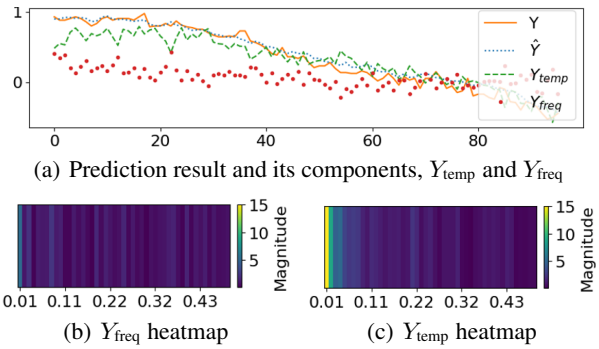
Figure 23: FDL vs. TDL contributions (Another example from the PEMS03 dataset, 96-step prediction).



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

(b) Y_{freq} heatmap (c) Y_{temp} heatmap

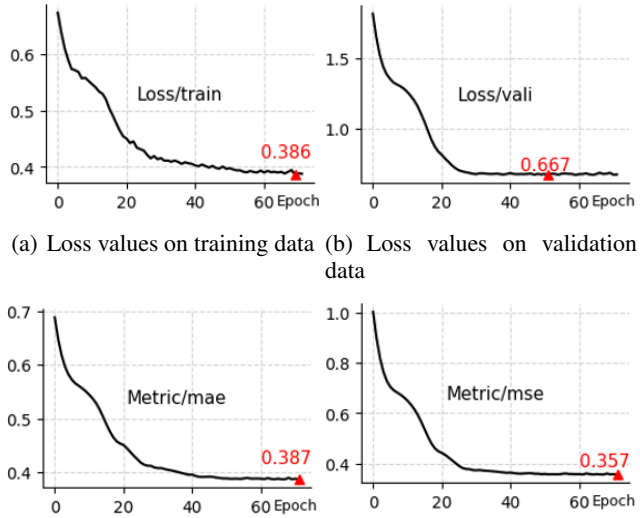
Figure 24: FDL vs. TDL contributions (An example from the PEMS08 dataset, 96-step prediction).



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

(b) Y_{freq} heatmap (c) Y_{temp} heatmap

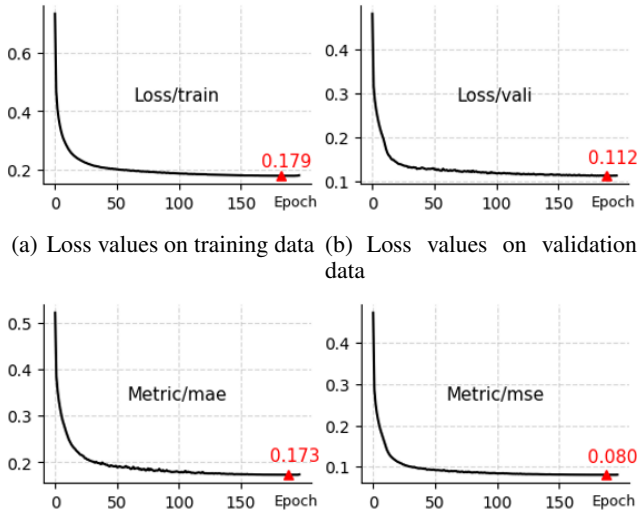
Figure 25: FDL vs. TDL contributions (Another example from the PEMS08 dataset, 96-step prediction).



(a) Loss values on training data (b) Loss values on validation data

(c) MAE scores on test data (d) MSE scores on test data

Figure 26: Loss and metrics change throughout the whole learning process of FDL on ETTh1 data.



(a) Loss values on training data (b) Loss values on validation data

(c) MAE scores on test data (d) MSE scores on test data

Figure 27: Loss and metrics change throughout the whole learning process of FTDL on PEMS07 data.

dict the next 4 hours. Such long inputs can alter the statistical properties of the target output. Therefore, an appropriate input length should be set based on both the prediction horizon and the characteristics of the data.

Our experiments indicate that our methods align with the general principle: a longer look-back window, within a certain range, increases the receptive field, potentially improving forecasting performance. Turning to the issue of time efficiency, we observe that as the input size increases, the execution time gradually increases. This represents a trade-off between prediction performance and time consumption. However, in all cases, the execution time per epoch remains under 1 minute, which is acceptable.

Convergence

We monitor the learning process of our methods across all datasets. Figures 26 and 27 illustrate two examples of training dynamics for 96-step predictions on two representative datasets: ETTh1 and PEMS07, which contain the fewest and most variables, respectively. Specifically, we show the learning process of FDL on ETTh1 and FTDL on PEMS07. We observe that the training and validation losses decreases steadily. Correspondingly, the MAE and MSE metrics on the test set also exhibit consistent reduction throughout the training process. The simplicity of the model architecture helps mitigate overfitting, a common concern with transformer-based models.

Effectiveness of CVNN for Time Series Forecasting

To further validate the capability of Complex-Valued Neural Networks (CVNNs), we conducted a series of intuitive experiments using synthetic datasets designed to simulate time series with mixed low- and high-frequency components. Each series was generated by combining sinusoidal signals with controlled frequency transitions and additive Gaussian noise.

The dataset consists of seven synthetic time series, each representing hourly data over one year (8,760 samples), similar in structure to the ETT benchmark. Each series includes low-frequency sinusoidal patterns, with periodic injections of high-frequency components at fixed intervals. A slight Gaussian noise was added to 50% of the time steps to simulate real-world variability. Additionally, a nonlinear trend was applied to each series to introduce non-stationarity. Figure 12 shows the generated synthetic data.

We evaluated four models: FDL_CVNN, FDL_Dual RVNN, TDL, and FTDL_parallel, where FDL_CVNN refers to the previously introduced complex-valued variant of FDL. Each model was trained on the synthetic toy dataset, and predictions were visualized across multiple epochs to assess learning dynamics.

As shown in Table 9, FDL_CVNN and FTDL_parallel exhibit faster convergence across both low- and high-frequency components, with progressively refined frequency representations during training. Their final predictions closely match the ground truth. In contrast, FDL_Dual RVNN captures periodic patterns but adapts more slowly to high-frequency signals compared to FDL_CVNN and FTDL_parallel. Nonetheless, it outperforms TDL, which struggles to model rapid oscillations.

These experiments confirm that FDL_CVNN consistently outperforms both FDL_Dual RVNN and TDL in learning complex frequency structures. Its ability to capture both low- and high-frequency patterns with precision highlights FDL_CVNN as a strong candidate for time series modeling tasks involving mixed-frequency data.

Why Does FDL Work Well?

We summarize the reasons for the high performance of FDL as follows:

- **Frequency Domain Representation**. Time series data $x \in \mathbb{R}^{D \times L}$ is transformed into the frequency domain using the Discrete Fourier Transform:

$$\tilde{x}_k^d = \sum_{t=1}^{L} x_t^d \cdot e^{-j\frac{2\pi}{L}kt}$$

  This transformation decomposes the signal into its constituent frequencies, making periodicity and trends explicit. Many real-world time series have strong low-frequency and periodic components, which are easier to model and predict in the frequency domain.
- **Complex-Valued Neural Networks (CVNN)**. FDL uses CVNNs to process complex-valued frequency data:

$$\tilde{y} = \text{CVNN}(\tilde{x})$$

  CVNNs learn relationships between amplitude and phase, which are crucial for reconstructing realistic time series. This is mathematically richer than real-valued networks, which cannot directly model phase information.
- **Conjugate Symmetry**. For real-valued time series, the FFT output satisfies conjugate symmetry:

$$\tilde{x}_k = \tilde{x}_{L-k}^*$$

  This property ensures that the inverse FFT will reconstruct a real-valued signal. FDL's implementation respects this symmetry, so predictions in the frequency domain remain physically meaningful.