PRISM: A HIERARCHICAL MULTISCALE APPROACH FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Forecasting is critical in areas such as finance, biology, and healthcare. Despite the progress in the field, making accurate forecasts remains challenging because real-world time series contain both global trends, local fine-grained structure, and features on multiple scales in between. Here, we present a new forecasting method, PRISM (Partitioned Representation for Iterative Sequence Modeling), that addresses this challenge through a learnable tree-based partitioning of the signal. At the root of the tree, a global representation captures coarse trends in the signal, while recursive splits reveal increasingly localized views of the signal. At each level of the tree, data are projected onto a time-frequency basis (e.g., wavelets or exponential moving averages) to extract scale-specific features, which are then aggregated across the hierarchy. This design allows the model to jointly capture global structure and local dynamics, enabling both reconstruction and forecasting. Experiments across benchmark datasets show that our method outperforms state-of-the-art methods for forecasting and also requires less runtime and memory. Overall, these results demonstrate that our hierarchical approach provides a lightweight and flexible framework for forecasting multivariate time series. The code [will be] available in [anonymized].

1 INTRODUCTION

The ability to anticipate the future is one of the most valuable tools in science and society. From natural systems to human behavior, prediction allows us to plan, adapt, and intervene before events unfold (1; 3; 32). Time series forecasting provides a formal way to make such predictions, but it remains challenging because signals evolve across multiple scales (1). Global trends interact with seasonal rhythms and local fluctuations, creating a hierarchical structure in which information at one scale sets context for the next. We argue that effective forecast models must capture this hierarchy explicitly, aligning long-term dynamics with fine-scale variability in a coherent representation.

Recent work has moved toward this goal, but current approaches fall short of learning unified hierarchical time-frequency representations. Some models incorporate frequency cues to capture periodicity and compact spectral structure (24; 31), others emphasize temporal hierarchy through coarse-to-fine decompositions (2; 25), and still others operate mainly in the frequency domain (26; 27) without modeling the temporal structure. While these directions highlight the value of hierarchical design, they construct it in only one domain at a time or flatten frequency into shallow features without linking it to temporal context. What remains missing is a coherent and reconstructable hierarchy that organizes both time and frequency, and captures long-term structure, short-term variability, and the interactions between them in a unified representation.

In this work, we introduce PRISM (*Partitioned Representations for Iterative Sequence Modeling*), a new model that fills this gap by jointly learning a hierarchical decomposition in both time and frequency. PRISM builds a binary tree over time, recursively partitioning the input into overlapping segments that preserve local context. At each node, a time-frequency decomposition (via Haar wavelets (15) or alternative transforms like exponential moving average filters) produces multiple bands, which are then adaptively weighted by a lightweight router that reweighs bands. The outputs of child nodes are stitched together through linear mixing, yielding stable reconstructions and multiscale representations that are both interpretable and effective for forecasting. This design provides a

unified tree over time and frequency, with an explicit reconstruction pathway that grounds forecasts in coherent signal structure.

Across benchmark datasets, PRISM achieves accuracy competitive with strong state-of-the-art baselines and surpasses them in many forecasting settings. The model also provides interpretability through its frequency-weighting mechanism, revealing which bands contribute most to predictions, and robustness through its reconstructible design, which stabilizes learning and scales naturally to longer contexts. Together, these results demonstrate that hierarchical multiscale representations, when organized into a unified time–frequency tree, provide a principled pathway toward more accurate and efficient forecasting.

Our contributions are as follows:

- **A unified time-frequency hierarchy**: PRISM factorizes time series data using a hierarchical binary tree splitting in the time domain and a hierarchical frequency encoder at every step of the time hierarchy, providing a new type of hierarchical time-frequency features.
- **A novel feature importance scoring approach**: For each hierarchical time-frequency feature of time series, PRISM assigns learnable importance scores to highlight the features that improve the forecasting accuracy augmented with the auxiliary reconstruction loss.
- **Comprehensive evaluation**: We show that our design choices improve the accuracy of our model. PRISM outperforms current strong baselines on the highest number of standard datasets and forecast horizons, establishing new state-of-the-art across forecasting tasks.

2 Related Work

We group prior work by how they use time and frequency information. We first review models that process both domains at once. We then cover models that build a hierarchy only in time. Next we discuss methods that work mainly in the frequency domain. We conclude this section by positioning our approach among these prior works.

Joint time and frequency modeling methods. Several methods combine information from the time and frequency domains at the same time. TimesNet (24) first discovers dominant periods with an FFT search. It reshapes a 1D series into 2D period–phase tensors and applies 2D CNN blocks to capture patterns inside each period and across periods. This gives a compact time stack guided by frequency peaks. FEDformer (31) adds a seasonal–trend decomposition to a Transformer. It replaces standard attention with frequency-enhanced blocks that operate on a sparse set of Fourier or wavelet modes. This reduces cost and focuses the model on informative bands. ETSformer (23) builds on exponential smoothing and adds frequency attention. It organizes the network into level, growth, and seasonality modules, so each module has a clear role. TFDNet (11) forms a time–frequency matrix with the short-time Fourier transform (STFT), then uses multi-scale encoders and separate time and frequency blocks for trend and seasonal parts. This design lets the model align slow and fast dynamics across domains. These models mix domains and gain efficiency from compact spectral views. However, they do not build a learnable hierarchy in both time and frequency, and they do not define a reconstructable pathway that stitches local pieces back together. Another work is CoST (22), which uses time-domain and frequency-domain contrastive losses to learn disentangled trend and seasonal representations for downstream forecasting

Time domain hierarchical decomposition methods. A second line of work builds a multiscale structure only along time. Early statistical models such as ARIMA (1) and SARIMA (1) decompose a series into trend, seasonality, and noise with fixed forms and linear dynamics.

Later deep learning methods keep the idea of decomposition but learn it from data. For instance: N-HiTS (2) uses hierarchical interpolation and multi-rate sampling. It assembles forecasts at several resolutions, so coarse blocks capture slow movements and fine blocks refine details. DLinear (28) is another notable work which uses seasonal-trend decomposition integrated with linear layers, maintaining a simple structure with impressive performance. TimeMixer (20) is an MLP forecaster that performs decomposable multi-scale mixing on the past. It also ensembles several simple predictors for the future to improve stability. SCINet (8) applies a recursive split with downsampling, convolutions, and cross-branch interaction. The split-and-interact pattern captures multi-resolution patterns while keeping computation low. N-BEATS (14) is a basis-expansion stack with backcast and

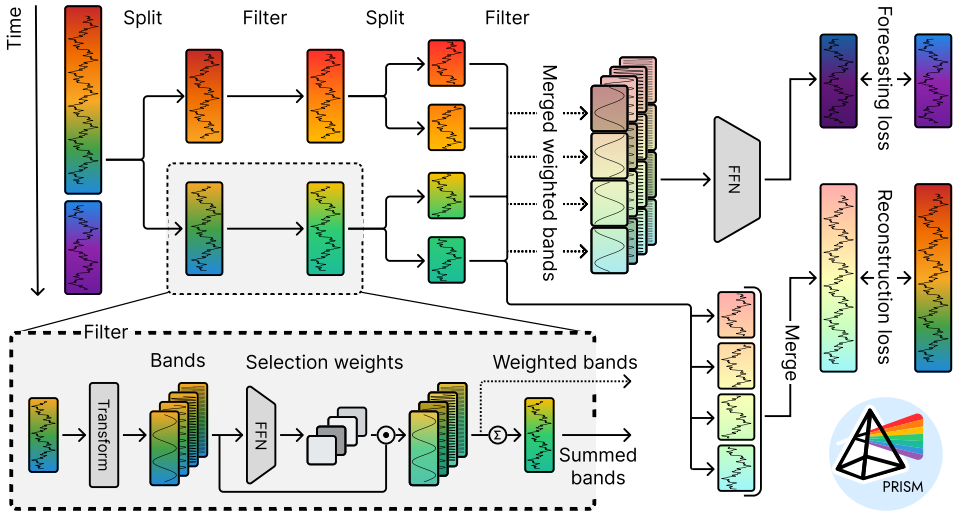

Figure 1: **The PRISM model overview.** The timeseries is partitioned into chunks recrusively to produce smaller temporal segments. At each level of this splitting procedure, we apply learnable filter banks that use time-frequency representations and learnable weights to extract features from the segment. The weights at each level of the feature hierarchy are distinct and not shared, allowing the model to extract different sets of coefficients that are meaningful for forecasting and reconstruction. The learning is driven by two objectives, a forecasting loss (top, right) on predictions of the future and a reconstruction loss on the past (bottom, right).

forecast heads. The interpretable variant uses polynomial and harmonic bases to separate trend and seasonality. Residual stacking gives a deep hierarchy over time. Autoformer ([25]) places a progressive trend–seasonal decomposition inside the network. An auto-correlation module models long-range dependencies in a cost-effective way. MICN ([19]) uses multiple convolutional branches with different kernels. A local branch with downsampling convolution captures short-range cues, while a global branch with isometric convolution covers long context. Pyraformer ([9]) introduces pyramidal attention. An inter-scale tree summarizes features at several resolutions with low complexity, which helps on long sequences. These designs learn a clear hierarchy over time. They do not pair it with a matching frequency hierarchy, and they do not route information across bands at each time scale.

Multiscale frequency modeling or selection methods. A third line of work focuses on the frequency side. FreTS ([26]) transforms a series to the frequency domain and applies redesigned MLPs to the real and imaginary parts of these coefficients to give a compact global spectral representation. FiLM ([30]) uses Legendre projections to form a compact long memory for the history and Fourier projections to reduce noise before using a light expert mixer to combine predictions from multi-scale views of the history. D-PAD ([27]) splits a series into frequency ranges with a multi-component decomposition. It then applies a decomposition–reconstruction–decomposition pipeline to disentangle information at the same frequency and fuses the components for forecasting. These methods emphasize spectral selection and disentangling. They do not build a time hierarchy and they do not define a joint tree over both domains.

Positioning of our method. Our method, PRISM, learns a hierarchy in both domains at the same time. It builds a binary tree over time with overlap to keep local context and smooth boundaries. At each node it applies a same-length, reconstructable band partition. A lightweight router selects bands per node and forwards a band-weighted sum to the left and right children. Child outputs are stitched with a linear cross-fade, so reconstruction is simple and stable. This yields a coherent time–frequency hierarchy that supports both reconstruction and forecasting, and it fills the gap left by prior work that is hierarchical in only one domain or that mixes domains without a shared hierarchy.

3 METHODS

Here we describe PRISM, our model for the forecasting of the time series. We train our model on a combination of two losses, reconstruction of the past and prediction of the future. We provide an overview of our method in Figure 1.

3.1 PROBLEM SETUP

We define the task of time series forecasting as follows (12): Let $\mathcal{X} = \{x_t\}_{t=1}^{T}$ represent a univariate or multivariate time series in input space $\mathbb{R}^D$, where $t \in \{1, 2, \ldots, T\}$ represents time. We consider a context window $\mathcal{X}_{\text{context}} = \{x_t\}_{t=1}^{T_{\text{context}}}$ of length T_{context} and a prediction window $\mathcal{X}_{\text{forecast}} = \{x_t\}_{t=T-T_{\text{forecast}}}^{T}$ of length T_{forecast} such that $T_{\text{context}} + T_{\text{forecast}} = T$. The objective is then to build a model that given the context window $\mathcal{X}_{\text{context}}$, predicts the time series in the prediction window $\mathcal{X}_{\text{forecast}}$. To prepare the data for our model, we sample segments $x \in \mathbb{R}^{B \times T \times C}$ from raw time series provided in datasets. Here B denotes batch size, C denotes the number of the recorded channels in a multivariate time series, and T denotes the length (duration) of the time series.

3.2 THE PRISM MODEL

In our PRISM model, the data is cycled through (i) the time decomposition step; (ii) the frequency decomposition step, (iii) the importance weighting of frequency bands, and (iv) the reconstruction step, thus generating hierarchical representations of input time series both in time and frequency domains. The temporal hierarchy is obtained through iterated bisection of the time series; thus, this hierarchy is not formed in a single step but requires the entire iteration. The frequency hierarchy, in contrast, is obtained separately for every level of the time hierarchy. It provides signal decomposition and cleaning, progressively refining the input for the next time-decomposition steps. We provide the details below.

In the **time decomposition step** i_{temp}, we split the time series of the length (duration) $T(i_{\text{temp}})$ (dimensions: $T(i_{\text{temp}}) \times C$; here and below we omit the batch dimension B) along the time axis into two equally sized segments with the overlap o (resulting durations: $T(i_{\text{temp}}+1) = (T(i_{\text{temp}})+o)/2$). On the first iteration of this step, the duration $T(i_{\text{temp}} = 0)$ equals to the length of the model's context window T_{context}; on the second iteration it equals to $(T_{\text{context}} + o)/2$ and so on. This sequential process produces a hierarchy of overlapping segments of the time series, with $M = 2^{i_{\text{temp}}}$ segments on the iteration i_{temp}, in the form of the binary tree.

Each time decomposition step is followed by the **frequency decomposition step** where we use the Haar discrete wavelet transformation (15) (DWT; K=6) to split each segment of the time series (whose size $T(i_{\text{temp}}) \times C$ varies by the iteration i_{temp}) into the hierarchy of K signals corresponding to different frequency bands of the original signal (overall dimension: $T(i_{\text{temp}}) \times C \times K$). The hierarchical frequency filters here are easily interchangeable. In ablations, we replace Haar DWT with the fast Fourier transformation (16; 18) (FFT; K=4, rectangular masks), binomial pyramids (5) (K=4, k_0=3, k_{grow}=2), differences of Gaussians (7) (DoG; K=6, σ_0=1.0, ratio=1.6), exponential moving averages (4) (EMA; K=4, τ_0=8.0, grow=3.0), and the MCD block from the D-PAD model (27).

To calibrate the impact of frequency bands in time segments of the signal, we follow each frequency decomposition step with the **importance weighting** step. To this end, we compute the summary statistics for each frequency band of every time segment of the signal ($\mu, \sigma, a_{\max}, \|\Delta\|_1, \|\Delta^2\|_1, a_{\max}/(\sigma+\varepsilon)$) and pass it as an input to a two-layer MLP. Here μ is the mean value, σ is the standard deviation, and $a_{\max}$ is the maximum amplitude of the signal; $\|\cdot\|_1$ is the l_1-norm; Δ and Δ^2 are the first and the second derivatives of the signal. Using the summary statistics of the signal as opposed to the signal itself as input reduces the computational complexity of the model and allows for the shared use of the same MLP with signals of different length (e.g. weight sharing across the scales of time hierarchy). The MLP produces a single importance score $s_{c,k}$ per input (here the channel $c \in C$ and the frequency band $k \in K$). The importance scores $s_{c,k}$ are then converted into the importance weights $w_{c,k}$ for the frequency bands through the softmax operation with temperature τ.

These importance weight are used in the subsequent **reconstruction step** to combine the K signals (of size $T(i_{\text{temp}}+1) \times C$) from all the frequency bands into the joint signal (of size $T(i_{\text{temp}}+1) \times C$). This way, the joint signal retains the information that is most important for the downstream forecasting task, while irrelevant information is suppressed. All the steps described above are then iterated, resulting in the processing of progressively smaller segment of time series, enabling multiscale analysis of its data. On the last iteration, prior to combining individual bands into the joint signal, they are concatenated along the time dimension with linear cross-fade over the overlap window o to form the sequence of the original size $T_{\text{context}} \times C$.

The concatenated multi-band signal (of size $B \times T_{\text{context}} \times C \times K$) is then fed as input to $M \times K$ two-layer MLPs (for M temporal segments times K frequency segments) that outputs the forecast for the time series over the forecast window ($\chi^{\text{model}}_{\text{forecast}}$ of size $B \times T_{\text{forecast}} \times C$). The other version of the multi-band signal, summed over the band dimension ($\chi^{\text{model}}_{\text{context}}$ of size $B \times T_{\text{context}} \times C$), is meant to reconstruct the time series signal over the context window. Both outputs are used in the loss function as described below. We train the model end-to-end with an MSE loss function that couples the forecasting loss with the auxiliary reconstruction loss:

$$\mathcal{L} = \text{MSE}(\chi^{\text{model}}_{\text{forecast}}, \chi_{\text{forecast}}) + \text{MSE}(\chi^{\text{model}}_{\text{context}}, \chi_{\text{context}}). \quad (1)$$

We train our model in batches of $B = 512$ using the Adam optimizer with learning rate 1e-4. Early stopping monitors the validation MSE with the patience of 15 steps and the improvement margin of $\delta = 2 \times 10^{-4}$. We evaluate the model on the held-out test split to report final MSE and MAE.

4 Results

4.1 Experiment setup

Datasets. To comprehensively examine the performance of our model under diverse settings, we have tested it on a standard set of time-series datasets (24). The *ETT* datasets (29) (*ETTh1, ETTh2, ETTm1, ETTm2*) record electricity transformer temperature data collected either hourly or every 15 minutes from 2016 to 2018, each including seven variables such as oil temperature and load. The *Electricity* dataset (6) consists of hourly electricity consumption records for 321 customers between 2012 and 2014, while the *Traffic* dataset contains hourly road occupancy rates measured by 862 sensors across the San Francisco Bay Area. The *Exchange* dataset (6) consists of daily exchange rates of eight currencies against the US dollar, collected from 1990 to 2016. Additionally, the *Weather* dataset (28) includes 21 meteorological indicators collected every 10 minutes from a weather station in 2020. For the consistency with prior literature, we followed the established protocols by splitting the *ETT* datasets into training, validation, and test sets using a 6:2:2 ratio, and applied a 7:1:2 split for the remaining datasets (21). This panel of diverse benchmarks has enabled the assessment of our model's forecasting capabilities across varying temporal resolutions and signal characteristics (24).

Evaluation. To contextualize our results, we compared the accuracy of the forecasts of our model to the accuracies of prominent other models available in literature. To evaluate the models under varied settings, we trained them to make predictions over varied time horizons including 96, 192, 336, and 720 timesteps into the future. As the models in prior work were often trained and evaluated only on subsets of these benchmarks and time horizons, we have reevaluated these models to provide the comprehensive results in all the considered settings.

4.2 Performance on forecasting benchmarks

`PRISM` sets overall SOTA performance on a panel of standard datasets. In this section, we have compared the performance of our model to that of several highest-performing methods including D-PAD (27) (current state-of-the-art method for timeseries forecasting), DLinear (28), N-HiTS (2), iTransformer (10), PatchTST (13), and XPatch (17) (Table 1). We used the total of 8 datasets, considering 4 forecast horizons for each of them ($4 \times 8 = 32$ evaluations total), as outlined above. We found that, out of 32 considered setting, our model, `PRISM`, has shown the best MSE in 17 settings and the best MAE in 18 settings, consistently outperforming all the models within the comparison. Our model was followed by D-PAD, a model that has shown the best MSE in 9 settings and the best MSE in 10 settings. Notably, D-PAD is a model that utilizes a hierarchical temporal encoder but does not use a hierarchical frequency encoder. Thus, overall, our results speak to the utility of jointly using hierarchical time and frequency decompositions in the forecasting of time series. We further investigate this proposal in ablation sections below.

4.3 Ablations

To verify that each design choice is necessary and that the overall configuration is optimized for time-series forecasting, we conduct targeted ablations that remove or alter each of the model's component in turns. Our `PRISM` model consists of a hierarchical tree encoder for the time domain,

Table 1: **Forecasting performance across varying future horizons.** The MSE and MAE are reported for 8 datasets over 4 seeds. We compare the performance of our method PRISM with D-PAD (27), DLinear (28), N-HiTS (2), and transformer-based baselines (iTransformer (10), PatchTST (13), and XPatch (17)). The Context window is 336 for all datasets and the horizons are 96, 192, 336, 720 samples.

Dataset	Ctx	H	PRISM		D-PAD		DLinear		N-HiTS		iTransformer		PatchTST		XPatch	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	336	96	**0.355**	**0.374**	0.357	0.376	0.375	0.399	0.475	0.498	0.386	0.405	0.414	0.419	0.376	0.386
	336	192	**0.390**	0.405	0.394	**0.402**	0.405	0.416	0.492	0.519	0.441	0.436	0.413	0.429	0.417	0.407
	336	336	0.386	**0.412**	**0.374**	**0.412**	0.479	0.443	0.550	0.564	0.487	0.458	0.422	0.440	0.449	0.425
	336	720	0.421	0.445	**0.419**	**0.442**	0.472	0.490	0.598	0.641	0.503	0.491	0.447	0.468	0.470	0.456
ETTh2	336	96	0.267	0.322	0.272	0.327	0.289	0.353	0.328	0.364	0.297	0.349	0.274	0.337	**0.233**	**0.300**
	336	192	0.311	0.359	0.331	0.368	0.383	0.418	0.372	0.408	0.380	0.400	0.341	0.382	**0.291**	**0.338**
	336	336	**0.318**	**0.364**	0.321	0.370	0.448	0.465	0.397	0.421	0.428	0.432	0.329	0.384	0.344	0.377
	336	720	0.390	**0.421**	0.399	0.426	0.605	0.551	0.461	0.497	0.427	0.445	**0.379**	0.422	0.407	0.427
ETTm1	336	96	0.288	**0.321**	**0.285**	0.328	0.299	0.343	0.370	0.468	0.334	0.368	0.293	0.346	0.311	0.346
	336	192	0.325	0.359	**0.323**	**0.349**	0.335	0.365	0.436	0.488	0.377	0.391	0.333	0.370	0.348	0.368
	336	336	0.358	0.379	**0.351**	**0.372**	0.369	0.386	0.483	0.510	0.426	0.420	0.369	0.392	0.388	0.391
	336	720	**0.410**	0.419	0.412	**0.405**	0.425	0.421	0.489	0.537	0.491	0.459	0.416	0.420	0.461	0.430
ETTm2	336	96	**0.158**	**0.234**	0.162	0.247	0.167	0.263	0.184	0.262	0.180	0.264	0.166	0.256	0.164	0.248
	336	192	**0.214**	**0.267**	0.218	0.283	0.224	0.303	0.260	0.293	0.250	0.309	0.223	0.296	0.230	0.291
	336	336	**0.264**	0.327	0.267	**0.321**	0.284	0.342	0.313	0.359	0.311	0.348	0.274	0.329	0.292	0.331
	336	720	0.360	0.383	**0.353**	**0.372**	0.397	0.421	0.411	0.421	0.412	0.407	0.362	0.385	0.381	0.383
Traffic	336	96	0.362	0.237	**0.359**	**0.236**	0.410	0.282	0.407	0.290	0.395	0.268	0.360	0.249	0.481	0.280
	336	192	**0.376**	**0.244**	0.377	0.245	0.423	0.287	0.423	0.302	0.417	0.276	0.379	0.256	0.484	0.275
	336	336	0.401	0.257	**0.391**	**0.253**	0.436	0.296	0.446	0.321	0.433	0.283	0.392	0.264	0.500	0.279
	336	720	0.433	**0.271**	**0.413**	0.282	0.446	0.305	0.528	0.369	0.467	0.302	0.432	0.286	0.534	0.293
Electricity	336	96	**0.122**	**0.219**	0.128	0.220	0.140	0.237	0.151	0.254	0.148	0.240	0.129	0.222	0.159	0.244
	336	192	**0.145**	**0.235**	0.148	0.236	0.153	0.247	0.170	0.273	0.149	0.231	0.147	0.240	0.160	0.248
	336	336	**0.159**	**0.243**	0.163	0.253	0.160	0.259	0.200	0.291	0.178	0.253	0.163	0.259	0.182	0.267
	336	720	**0.194**	**0.281**	0.201	0.286	0.203	0.301	0.244	0.356	0.225	0.317	0.197	0.290	0.216	0.298
Exchange	336	96	**0.081**	**0.198**	0.098	0.219	0.082	0.202	0.092	0.208	0.086	0.206	0.095	0.217	0.082	0.199
	336	192	0.176	0.297	0.204	0.321	**0.157**	**0.293**	0.208	0.300	0.177	0.299	0.201	0.322	0.177	0.298
	336	336	0.314	0.396	0.429	0.464	**0.305**	**0.414**	0.371	0.509	0.331	0.417	0.372	0.447	0.349	0.425
	336	720	0.712	0.606	0.721	0.607	**0.643**	**0.601**	0.888	1.447	0.847	0.691	0.873	0.699	0.891	0.711
Weather	336	96	**0.140**	**0.177**	0.143	0.181	0.176	0.237	0.160	0.197	0.174	0.214	0.149	0.198	0.146	0.184
	336	192	**0.187**	**0.227**	0.189	0.229	0.220	0.282	0.207	0.265	0.221	0.254	0.194	0.241	0.190	0.228
	336	336	**0.234**	**0.266**	0.239	0.268	0.265	0.319	0.273	0.301	0.278	0.296	0.245	0.282	0.236	0.273
	336	720	**0.302**	**0.311**	0.304	0.313	0.323	0.362	0.363	0.352	0.358	0.347	0.314	0.334	0.309	0.321
Best count (MSE)			**17**	—	**9**	—	3	—	—	—	—	—	1	—	2	—
Best count (MAE)			—	**18**	—	**10**	—	3	—	—	—	—	—	0	—	2

(hierarchical) feature decomposition for the frequency domain, learnable importance scoring for frequency features, and the joint loss that encompasses forecasting and reconstruction accuracy. We consider all these components below.

Hierarchical trees enable highest forecast accuracy among time-series encoders. Our method, PRISM, uses a hierarchical tree encoder to generate multiscale representations of input time series. To ensure the utility of our encoder, we have considered an alternative where we have replaced it with an MLP encoder (Table 2, "Encoder"). This alternation has degraded the performance of the model by the average of 32.94%. To test whether the hierarchical processing in the time domain positively affects the time-series forecasting capabilities of the model, we considered an ablation where only one layer of the tree was evaluated. We found that this ablation has degraded the performance of the model by the average of 8.83%, suggesting that limited temporal partitioning underexposes the model to important timescales. These results suggest the utility of hierarchical temporal encoding of data in time-series forecasting. We provide the detailed per-dataset results in Supplementary Table 3.

Wavelets enable the highest forecast accuracy among signal feature banks. Our model, PRISM, decomposes time series into hierarchical frequency components. While we chose to use wavelets for such a decomposition, PRISM is a general-purpose method that seamlessly admits different feature types. To evaluate whether the wavelets offer the best choice of basis functions for time-series forecasting, we have considered several classical and novel alternatives from literature (Table 2, "Frequency filter"). As a prominent new alternative, we have applied the MCD block of the D-PAD algorithm. Among the classical choices, we considered the fast Fourier transformation (FFT), the difference of Gaussians (DoG) and the binomial pyramid. We have also tested the exponential moving average (EMA) as a simple baseline. We found that the use of the FFT has led to the substantial loss of the average time-series forecasting performance, amounting to 14.32%. The other classical choices, binomial pyramid, EMA, and DoG have led to a smaller yet consistent decrease of the

Table 2: **Ablations and architecture search.** The **average** MSE and MAE are reported for 8 datasets over 4 seeds. We compare the performance of our method PRISM with different choices of encoders, frequency filters, importance scoring, loss functions, and other architecture choices. The context window is 336 for all datasets and the horizons are 96, 192, 336, 720 samples. We provide the expanded (non-averaged) data in Appendix.

Ablation	Alternative	Mean MSE	Mean MAE	ΔMSE vs PRISM (%)	ΔMAE vs PRISM (%)
–	PRISM	**0.307**	**0.345**	–	–
Encoder	Tree-level 1	0.334	0.372	-8.83%	-7.92%
	MLP	0.408	0.410	-32.94%	-18.78%
Frequency filter	EMA (exponential moving average)	0.330	0.368	-7.45%	-6.56%
	MCD (D-PAD (27) filter block)	0.323	0.364	-5.23%	-5.62%
	DoG (difference of Gaussians)	0.331	0.369	-7.79%	-6.92%
	Binomial pyramid	0.328	0.367	-6.73%	-6.41%
	FFT (fast Fourier transformation)	0.351	0.378	-14.32%	-9.43%
	Learnable filters	0.333	0.370	-8.56%	-7.14%
Importance scoring	All shared weights (across the entire tree)	0.329	0.368	-7.01%	-6.73%
	No shared weights (across the entire tree)	0.321	0.364	-4.66%	-5.46%
	No MLPs (all importance scores = 1)	0.332	0.373	-8.21%	-8.25%
	Learnable threshold (importance scores = 0 or 1)	0.338	0.373	-10.01%	-8.22%
Loss	No reconstruction loss	0.333	0.369	-8.46%	-6.87%
Architecture	No residual connections	0.329	0.367	-7.23%	-6.35%
	All residual connections (in every layer)	0.337	0.373	-9.80%	-8.04%

performance amounting to 6.73%, 7.45% and 7.79% respectively. The smallest (yet still negative) gap was observed, in favor of PRISM, with the MCD method, amounting to 5.23%. We have also tried a fully learnable filter, that has led to the decrease of -8.56% Together, these results speak to the utility of the wavelets in time-series forecasting. We provide the detailed per-dataset results in Supplementary Table 4.

Learnable importance scores per tree hierarchy level are the best way to combine wavelets. Our method, PRISM, computes the importance scores for individual frequency components of the time series before adding them back together. To compute the importance scores in PRISM, we use a trainable MLP module. To establish the importance of the importance scoring, we performed the ablation where there is no MLP at all (Table 2, "Importance scoring"). We found that this ablation has reduced the performance of our model on our panel of datasets by the average of 8.21%. To try an alternative, we performed the second ablation where we replaced the MLP with a learnable threshold – similarly to wavelet literature. We found that the resulting performance was sill lagging by 10.01% compared to PRISM.

For the MLP, our model PRISM uses shared weights within a tree layer but different MLP weights across the tree layers. To verify whether per-layer is the best way of the weight sharing, we trained two additional models where the MLP weights were either shared among all the tree layers or, alternatively, were not shared at all (were individual within each branch and each tree layer). Both alternations have led to the decrease of the forecasting performance on our datasets amounting to 7.01% and 4.66% respectively. We provide the detailed per-dataset results in Supplementary Table 5.

Auxiliary reconstruction loss and residual connections improve time-series forecasts. Here we have examined the impacts of our remaining design choices on the accuracy of time series forecasts. First, we have evaluated the importance of our auxiliary reconstruction loss, that has accompanied the forecast accuracy loss in PRISM (Table 2, "Loss"). We found that the ablation of the auxiliary reconstruction loss has decreased the overall forecast performance on our panel of datasets by the average of 8.46%. We then have investigated the role of the residual connections for time-series forecasting (Table 2, "Architecture"). To alter our choice of having a single set of long-range residual connections, bypassing the entire model, we considered the variants of the model architecture with residual connections in each layer and, alternatively, with no residual connections at all. We found that these alternation have led to the average decreases in the model's performance amounting to 9.80% and 7.23% respectively. Together, these results speak toward the importance of the auxiliary reconstruction loss and long-range residual connections in time-series forecasting models. We provide the detailed per-dataset results in Supplementary Table 3.

Comparison of training times Besides the forecasting accuracy, another important property of time-series models is the training time. To evaluate how our model compares to other prominent models in terms of the runtime, we evaluated the training speed of several models on the ETTh1 dataset using the forecast window of 96 timepoints, one of the most popular evaluation settings among the datasets considered in this work. We followed the standard approach and recorded the times that were necessary for the training of multiple (ten) epochs to negate the effects of the training overhead. For our model, PRISM, the training of 10 epochs took 65 seconds. We found that, this way, our model has shown the two-fold improvement over its closest counterpart, the D-PAD, whose training took 141 seconds for 10 epochs. The next counterpart of our model, DLinear, was somewhat faster to train, taking 42 seconds per 10 epochs, however, this speedup came at the cost of the decreased average performance in the forecasting task (-2.95% MSE). Continuing this trend, iTransformer took 32 second per 10 epochs at the performance drop of 4.70% MSE. The fastest results were shown by PatchTST, N-HiTS, and xPatch with 12, 10, and 7 seconds per 10 epochs respectively, at the performance decrease of 1.70, 6.87, and 3.45% respectively. Overall, these results suggest that our model offers a reasonable time-accuracy tradeoff while PatchTST also fares well on this metric.

4.4 How is the model building time-frequency representations?

We next sought to investigate the types of time-frequency representations that are formed at different nodes of the tree, and how they are mixed together to build forecasts. In Figure 2, we show the decomposition formed across three frequency bands (low, medium, high) in two time segments (left, right) – making the total of $2 \times 3 = 6$ representations. Although the time trees in our work could be deeper and sets of frequency features could be wider, we chose this level of decomposition for interpretability and illustration purposes. From top to bottom, in Figure 2 we show these six individual representations (top); the cumulative reconstruction of the input signal in the context window obtained as a weighted sum of these representations multiplied by their importance scores w (middle; the cumulative sum goes from left representations to right representations), and the cumulative forecast for the input signal over the forecast window, obtained with the MLP head that bases on these individual representations (bottom; the cumulative sum goes from left to right).

We found that our model, PRISM, faithfully captures high-, mid-, and low-frequency components of the input signal, while also honoring the local variabilities of the signal if different temporal segments. When added, these components progressively contribute to the dynamics of time series in both reconstruction and forecasting tasks. Of note, even though the wavelets are able to perfectly capture the dynamics of time series in the context window, the actual reconstructed signal that we observe differs from the ground truth. This is because, in our model, the reconstruction objective is balanced with the forecasting objective: We hypothesize that the model, through its end-to-end training, learns to ignore the features that do not contribute to the time series forecasting (e.g. noise

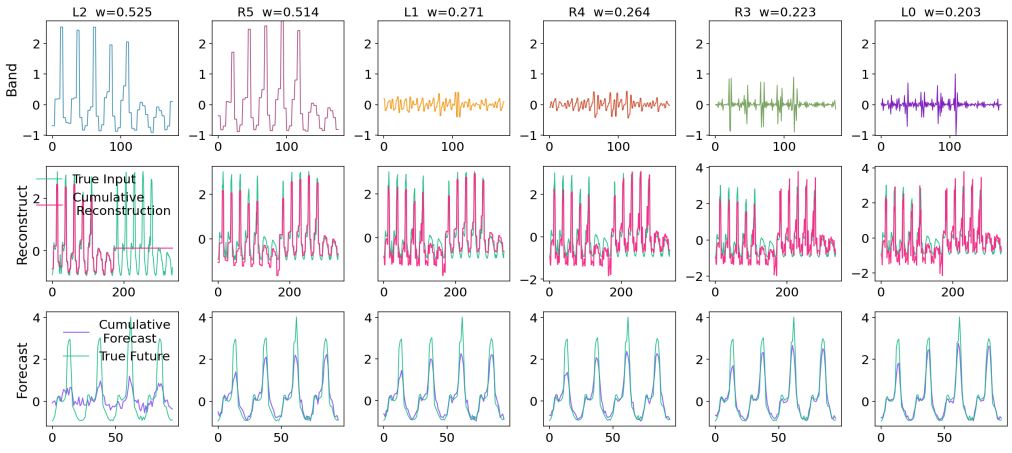

Figure 2: **Multiscale features learned by the model and their impact on time series prediction tasks.** (Top) multiscale signal components from different time- and frequency levels (L and R stands for left and right tree nodes in time decomposition; w is the associated importance weight), (middle) cumulative forecast of the time series based on these components, (bottom) cumulative reconstruction of the time series from these components.

of different frequencies) by setting low importance weights to corresponding frequency components. Overall, our model successfully forecasts time series by prioritizing relevant frequency components in defined temporal domains and downplaying different sources of noise.

Importance weights To evaluate whether our model, PRISM, has captured the relevant components of the input time series data, we evaluated the importance weights learned by the model across the hierarchy of frequencies in our datasets. To make the evaluation tractable, we have combined the importance weights from different branches of time hierarchy. While in the wavelet decomposition, to reconstruct the signal, all the components should be combined with the same weights, we found that the importance scores learned by our model are non-uniform and differ by the dataset (Figure 3). Overall, our model captures the important frequencies in the data while suppressing irrelevant noise.

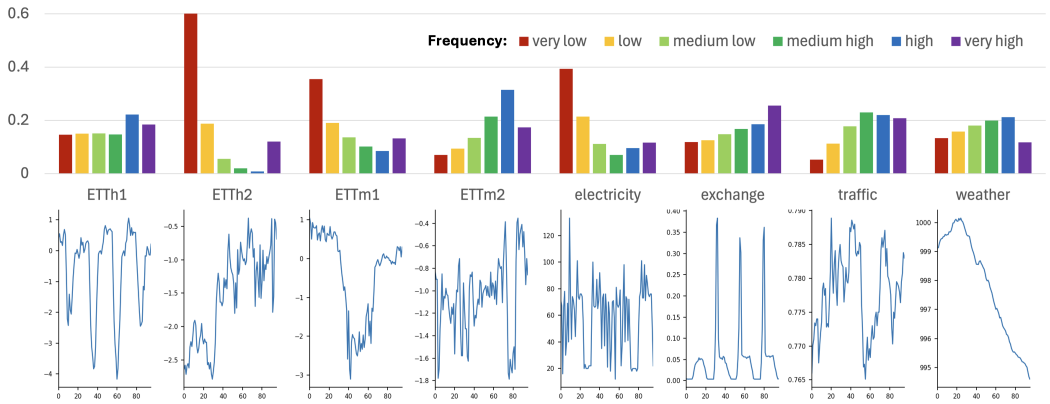

Figure 3: **Importance weights learned by the model in different datasets.** (Top) the importance weights of various frequency components in the multiscale time series representation for subsequent reconstruction and forecasting (averaged across time hierarchy). (Bottom) examples of time series data in corresponding datasets.

5 Discussion

In this work we introduced PRISM , a time-series forecasting model for that builds a unified hierarchy across time and frequency. By partitioning time series into overlapping segments, decomposing each segment into frequency bands, and routing information through importance weights, PRISM learns representations that enable both reconstructions and forecasts for time series. Our experiments show that this design offers competitive or state-of-the-art accuracy on benchmark datasets while providing interpretability through frequency weighting and stability through joint reconstruction.

Our results reinforce the view that hierarchical design is a powerful principle for time series modeling. Prior methods of time series forecasting have exploited hierarchy in either time or frequency, but PRISM demonstrates that organizing both domains together provides tangible benefits. In particular, the ability to align global structure with local fluctuations through a shared tree enables forecasts that remain robust across horizons.

Beyond accuracy, PRISM offers several practical advantages. The frequency weighting mechanism highlights which bands contribute to predictions, giving a direct window into the model's reasoning. At the same time, the recursive decomposition keeps computation lightweight, scaling gracefully with longer input windows without the quadratic costs typical of attention-based models.

While PRISM is effective, there are natural extensions that have the potential of further improving our method. First, our current frequency decomposition relies on fixed transformations (e.g., Haar wavelets or FFT). Learning adaptive frequency decomposition bases could further improve the method's flexibility. Second, the binary partitioning strategy imposes a fixed tree structure, whereas data-driven partitioning may capture irregular dynamics more faithfully.

The ability to forecast reliably across scales is central to many scientific and societal challenges. By showing how hierarchical representations can unify time and frequency, PRISM provides a foundation for models that are both accurate and interpretable. We hope this work stimulates further research into hierarchical design and its role in building general-purpose forecasting models.

REFERENCES

[1] G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970. Google Books ID: 5BVfnXaq03oC.

[2] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 6989–6997, 2023.

[3] A. Ebadi and E. Sahafizadeh. Comparing time-series analysis approaches utilized in research papers to forecast covid-19 cases in africa: A literature review. *arXiv preprint arXiv:2310.03606*, 2023.

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

[5] R. A. Haddad, A. N. Akansu, et al. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Signal Processing*, 39(3):723–727, 1991.

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

[7] T. Lindeberg and D. Fagerström. Scale-space with casual time direction. In *European conference on computer vision*, pages 229–240. Springer, 1996.

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

[9] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022.

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

[11] Y. Luo, S. Zhang, Z. Lyu, and Y. Hu. Tfdnet: Time–frequency enhanced decomposed network for long-term time series forecasting. *Pattern Recognition*, 162:111412, 2025.

[12] N. Nguyen and B. Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9117–9125, 2021.

[13] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[14] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[15] D. B. Percival and A. T. Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge university press, 2000.

[16] D. B. Percival and A. T. Walden. *Spectral analysis for univariate time series*, volume 51. Cambridge University Press, 2020.

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

[18] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.

[19] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[21] Y. Wang, H. Wu, J. Dong, Y. Liu, M. Long, and J. Wang. Deep time series models: A comprehensive survey and benchmark. 2024.

[22] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*, 2022.

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

[24] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

[25] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[26] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, and Z. Niu. Frequency-domain mlps are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36:76656–76679, 2023.

[27] X. Yuan and L. Chen. D-pad: Deep-shallow multi-frequency patterns disentangling for time series forecasting. *arXiv preprint arXiv:2403.17814*, 2024.

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

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

[30] T. Zhou, Z. Ma, Q. Wen, L. Sun, T. Yao, W. Yin, R. Jin, et al. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in neural information processing systems*, 35:12677–12690, 2022.

[31] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[32] J. Zou, Q. Zhao, Y. Jiao, H. Cao, Y. Liu, Q. Yan, E. Abbasnejad, L. Liu, and J. Q. Shi. Stock market prediction via deep learning techniques: A survey. *arXiv preprint arXiv:2212.12717*, 2022.