Filter Banks: from Low-Rank Representations to Deep Models for Efficient Time Series Forecasting

Ashutosh Vaishnav
Department of Computer Science
Aalto University, Finland
ashutosh.vaishnav@aalto.fi

Mohsen Amidzadeh
Department of Computer Science
Aalto University, Finland
mohsen.amidzade@aalto.fi

Teemu Kämäräinen
Department of Computer Science
University of Helsinki, Finland
teemu.kamarainen@helsinki.fi

Matti Siekkinen
Department of Computer Science
Aalto University, Finland
matti.siekkinen@aalto.fi

Mario Di Francesco
Department of Computer Science
Aalto University, Finland
mario.di.francesco@aalto.fi

Abstract

Time series forecasting is an actively researched problem with diverse applications. Recent work has shown that simple linear models can compete with complex deep learning architectures in terms of forecasting accuracy. This has led to the design of several small models, yet the reasons behind their effectiveness remain insufficiently understood. We address this gap through a theoretical framework grounded in reduced-rank regression and kernel analysis. We show both analytically and empirically that the intrinsic complexity of many TSF tasks is lower than commonly assumed and that the performance of linear models is linked to their alignment with low-rank data structures. Building on these insights, we propose a new design for low-rank neural networks that incur lower computational cost than linear models. Specifically, we introduce a simple *filter bank* architecture that provides a principled way to control model complexity through tunable and interpretable hyperparameters that directly correspond to the rank of the forecasting model. The proposed architecture also serves as a versatile building block for constructing deeper networks. Our evaluation shows that filter bank architectures achieve accuracy competitive with state-of-the-art models on long-term TSF benchmarks, at a substantially lower computational cost than recently proposed solutions.

1 Introduction

Time series forecasting (TSF) is important in diverse scenarios including energy systems, healthcare, finance, and resource management [Brink et al., 2016, Nielsen, 2019]. Accordingly, TSF has been studied through methods ranging from classical statistical models [Anderson, 2011] to deep architectures such as recurrent neural networks [Jin et al., 2020] and transformers [Zhou et al., 2021, Nie et al., 2023]. These modern models often achieve strong accuracy, but typically with very large parameter counts [Zhou et al., 2022].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

At the same time, recent work has shown that much simpler TSF models can be highly competitive. Linear models can outperform state-of-the-art transformers [Zeng et al., 2023], and later designs achieve competitive performance with architectures cheaper than a single linear layer [Xu et al., 2023]. Moreover, Toner and Darlow [2024] showed that these methods are essentially equivalent to unconstrained linear regression. Taken together, these results challenge the view that better TSF necessarily requires greater model complexity and motivate compact forecasters [Lin et al., 2024, Ekambaram et al., 2024, Huang et al., 2025, Ma et al., 2026], especially for resource-constrained deployments [Lin et al., 2020, Ekambaram et al., 2024].

Yet, the empirical success of such simple models remains poorly understood. Recent work has addressed this gap through low-rank regularization for linear TSF [Wang et al., 2026], while kernel-based interpretations of closed-form linear models have recently been explored in the context of sequence modeling [Yugay et al., 2026]. We complement these through reduced-rank regression and kernel analysis, arguing that many TSF tasks have intrinsically low complexity and that linear models succeed when aligned with low-rank structure in the data.

Guided by these insights, we introduce a *filter bank* architecture whose tunable hyperparameters map directly to model rank and which also serves as a building block for deeper networks. Our experiments show that filter bank architectures achieve accuracy competitive with state-of-the-art models on long-term TSF benchmarks, at a substantially lower computational cost than recently proposed solutions.

This work makes the following contributions.

- It uses reduced-rank regression to show that popular long-term TSF datasets overfit even single-layer linear models, and that constraining rank greatly reduces parameters with minimal accuracy loss while aligning with kernel regression (Section 3).
- It introduces a filter bank architecture that explicitly constrains model rank through tunable hyperparameters, and extends this design to rank-adaptive deep networks (Section 4).
- It shows empirically that filter bank models achieve competitive accuracy with few parameters and substantially lower computational cost on workstation- and embedded-class hardware (Section 5).

2 Related Work

A large body of work addresses TSF, with recent progress driven by deep sequence models [Benidis et al., 2022]. We focus on work most closely related to accurate yet efficient forecasters.

Transformer-based methods such as Autoformer [Wu et al., 2021], Informer [Zhou et al., 2021], FEDformer [Zhou et al., 2022], and PatchTST [Nie et al., 2023] combine attention with decomposition or frequency-based representations. They report strong benchmark results, but their complexity can be substantial, and Zeng et al. [2023] showed that much simpler baselines are competitive. Motivated by efficiency, several alternative neural architectures have been proposed. Mixing-style models such as TimeMixer [Wang et al., 2024] and Tiny Time Mixers [Ekambaram et al., 2024], state-space and Mamba-style designs such as TimePro [Ma et al., 2025] and S-Mamba [Wang et al., 2025], and Kolmogorov–Arnold approaches such as TimeKAN [Huang et al., 2025] all target cheaper sequence modeling.

Low-complexity linear and frequency-domain approaches such as DLinear [Zeng et al., 2023], FITS [Xu et al., 2023], FilterNet [Yi et al., 2024], SparseTSF [Lin et al., 2024], and MixLinear [Ma et al., 2026] are particularly relevant because they show that compact models can be highly competitive in long-term forecasting. In contrast with them, our method provides a more rigorous alternative: we explicitly impose low-rank structures through learned filters, rather than a fixed basis, to control capacity with interpretable and fully learnable architectures; our work is grounded on a kernel-based analysis framework, beyond purely empirical compression.

Several recent theory-first works also help contextualize why simple linear sequence models perform well. Wang et al. [2026] analyze linear TSF through characteristic roots and low-rank regularization, while Yugay et al. [2026] show that closed-form linear models admit a Gaussian-process interpretation in the context of anomaly detection. Our work is complementary: we derive a kernel-based filter-bank decomposition for forecasting and use it to motivate a learnable architecture.

Low-rank optimization is central to matrix completion / approximation and regression [Candès and Recht, 2009, Tropp and Webber, 2023, Meyer et al., 2011, Yin and Müller, 2025]. It has been

employed to compress convolutional neural networks [Tai et al., 2016, Sui et al., 2023, He et al., 2024], enable parameter-efficient adaptation in LoRA [Hu et al., 2022], and for spectral filtering in the frequency domain [Ma et al., 2026]. We build on the same principle, but leverage it to design forecasting models that are learned end-to-end from time series data.

3 Theoretical Foundations

This section develops our theoretical foundations through two complementary frameworks: reduced-rank regression and kernel regression. The former formalizes the search for a concise low-rank model that minimizes a squared loss; whereas the latter yields an equivalent representation in terms of horizon-invariant filters and horizon-varying coefficients, under the assumption that the predictor belongs to a reproducing kernel Hilbert space (RKHS). In this representation, the effective rank is governed by the interplay between the kernel eigenspectrum and regularization; Appendix A provides further details. Together, these perspectives motivate the architectures introduced later.

Before introducing these frameworks, we formalize the TSF problem. Let N denote the number of training samples. For each time index $i \in \{1, ..., N\}$, the historical observations are represented by $\mathbf{x}_i = [x_{i,1}, \ldots, x_{i,L}]^\intercal \in \mathbb{R}^L$, corresponding to a *look-back window* of L time steps, and the objective is to predict the next H values $\mathbf{y}_i = [x_{i,L+1}, \ldots, x_{i,L+H}]^\intercal \in \mathbb{R}^H$, namely the *prediction horizon*. For simplicity, we treat $\mathbf{x}_i$ as a single-channel sequence, since channel-independent models have been shown to outperform multivariate ones in this setting [Nie et al., 2023, Zeng et al., 2023]. The goal is to predict $\mathbf{y}_i$ from $\mathbf{x}_i$ as accurately as possible under an empirical loss.

We consider TSF strategies $h(\cdot) \in \mathscr{H} : \mathbb{R}^L \to \mathbb{R}^H$, where $\mathscr{H}$ denotes a function class of interest. The best model within $\mathscr{H}$ solves:

$$\mathrm{P}_1 : \qquad \min_{h(\cdot)} R[h] := \frac{1}{N} \sum_{i=1}^{N} \|h(\mathbf{x}_i) - \mathbf{y}_i\|^2, \qquad \text{s.t. } h(\cdot) \in \mathscr{H},$$

where $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ are samples drawn from the underlying distribution $\mathrm{P}_{\mathbf{x},\mathbf{y}}(\cdot, \cdot)$, and $R[\cdot]$ is the empirical loss. We refer to it as the Mean Square Error (MSE) throughout. The optimal solution to problem P_1 can be obtained with Ordinary Least Squares (OLS) [Toner and Darlow, 2024].

3.1 Reduced Rank Regression

Consider a standard linear TSF model:

$$\hat{\mathbf{y}}_i = \mathbf{W}^\intercal \mathbf{x}_i, \quad \text{for } i \in \{1, \ldots, N\}, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{L \times H}$ is the weight matrix, $\mathbf{x}_i \in \mathbb{R}^L$ the input time series at time i, and $\hat{\mathbf{y}}_i \in \mathbb{R}^H$ the respective predicted output. A principled way to construct low-rank yet accurate models is given by *reduced-rank regression* (RRR), wherein the weight matrix $\mathbf{W}$ is constrained to have rank $r < \min\{L, H\}$ [Mukherjee, 2013]. A rank-r matrix $\mathbf{W}$ can be equivalently expressed based on the following low-rank decomposition:

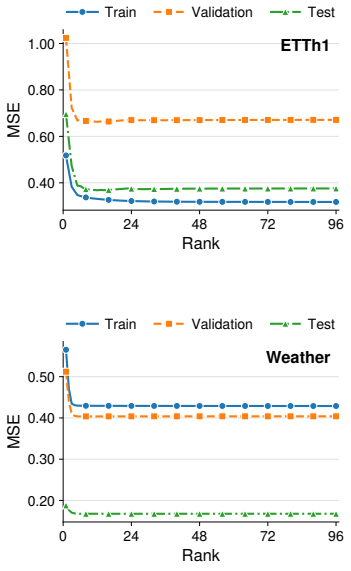
$$\mathbf{W} = \mathbf{L}_r \mathbf{R}_r^\intercal, \tag{2}$$

for two matrices $\mathbf{L}_r \in \mathbb{R}^{L \times r}$ and $\mathbf{R}_r \in \mathbb{R}^{H \times r}$. Such a decomposition reduces the number of parameters from LH (for a full-rank matrix) to $r(L + H)$, thereby achieving a significant reduction in the model parameters, especially when $r \ll \min\{L, H\}$ holds. For instance, a typical long-term TSF setup consists of $L = H = 720$; a full-rank model requires $720^2 = 518,400$ parameters, while a rank-10 model only needs $10 \cdot (720 + 720) = 14,400$ parameters, resulting in a 97% reduction.

This low-rank constraint naturally leads to the following *rank-constrained regression* problem:

$$\mathrm{P}_{\mathrm{RRR}} : \qquad \min_{\mathbf{W}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{W}^\intercal \mathbf{x}_i - \mathbf{y}_i\|^2, \qquad \text{s.t. } \mathrm{rank}(\mathbf{W}) \leq r, \tag{3}$$

where the rank constraint r directly controls model complexity through the parameter count. Although the rank constraint in $\mathrm{P}_{\mathrm{RRR}}$ makes the problem non-convex, its globally optimal solution still admits a closed-form expression [Mukherjee, 2013].



(a)

Dataset	Horizon / Full rank	Optimal rank r^*	Compr. ratio	Validation MSE	Test MSE with RRR	Test MSE with OLS	Impr. (%)
ETTh1	96	12	7.06×	0.663	0.368	0.376	2.09
	192	14	10.83×	0.891	0.402	0.413	2.69
	336	7	32.73×	1.092	0.442	0.448	1.25
	720	9	40.00×	1.209	0.491	0.492	0.15
ETTh2	96	22	3.85×	0.226	0.308	0.309	0.49
	192	6	25.26×	0.314	0.423	0.423	-0.06
	336	17	13.48×	0.417	0.537	0.539	0.32
	720	3	120.00×	0.694	0.844	0.900	6.18
ETTm1	96	18	4.71×	0.384	0.307	0.306	-0.31
	192	15	10.11×	0.494	0.337	0.335	-0.33
	336	14	16.36×	0.625	0.363	0.364	0.33
	720	20	18.00×	0.865	0.412	0.413	0.34
ETTm2	96	23	3.68×	0.117	0.166	0.166	0.06
	192	24	6.32×	0.159	0.228	0.228	0.08
	336	6	38.18×	0.213	0.297	0.295	-0.55
	720	18	20.00×	0.296	0.414	0.415	0.07
Weather	96	6	14.12×	0.404	0.168	0.168	0.03
	192	6	25.26×	0.462	0.211	0.210	-0.12
	336	7	32.73×	0.517	0.255	0.255	0.00
	720	7	51.43×	0.586	0.314	0.314	0.00
Average			**24.70**×				**0.64**

(b)

Figure 1: (a) Train, validation, and test loss vs. rank for ETTh1 (top) and Weather (bottom) at $H = 96$. Validation and test losses reach their optimum at low rank, then remain flat (Weather) or slightly worsen (ETTh1) at higher rank, indicating overfitting. (b) RRR vs. OLS across datasets and horizons. RRR yields large compression with little impact on test accuracy.

Proposition 1 [Izenman [1975], Mukherjee [2013]]**.** *Consider the reduced-rank regression* P_{RRR} *with a given* r*. Its solution* $\mathbf{W}_{\mathrm{R}}$ *complies with the following expression*

$$\mathbf{W}_{\mathrm{R}} = \mathbf{W}_0 \mathbf{V}_r \mathbf{V}_r^{\intercal}, \tag{4}$$

where $\mathbf{W}_0 = (\mathbf{X}^{\intercal}\mathbf{X})^{-1}\mathbf{X}^{\intercal}\mathbf{Y}$ *is the OLS solution,* $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^{\intercal} \in \mathbb{R}^{N \times L}$ *and* $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^{\intercal} \in \mathbb{R}^{N \times H}$ *denote the input and output data matrices (respectively), and* $\mathbf{V}_r$ *has the first* r *right singular vectors from the singular value decomposition of* $\mathbf{X}\mathbf{W}_0 = \mathbf{U}\mathbf{D}\mathbf{V}^{\intercal}$.

3.1.1 The RRR Framework for TSF

The RRR framework allows us to characterize low-rank solutions to TSF problems. For this analysis, we consider the Electricity Transformer Temperature (ETT) and Weather datasets [Zhou et al., 2021], which are among the most widely used benchmarks in long-term TSF research. The ETT dataset comprises ETTh1, ETTh2, ETTm1, and ETTm2, namely temperature values sampled every hour (h) and every 15 minutes (m) at two physical transformers (1 and 2). In all cases, we assume a look back window of $L = 720$ samples. We use the standard chronological train/validation/test splits, standardize each channel with training statistics, fit a single model shared across channels without instance normalization, and evaluate all ranks from 1 to the full rank $\min\{L, H\} = H$. The optimal rank r^* is selected on the validation set as the rank after which the validation MSE does not improve by more than 10^{-4} for three consecutive ranks.

Fig. 1a shows that as rank r increases for $H = 96$, the training loss decreases monotonically while the validation and test losses quickly reach their optimum at low rank and then remain flat or slightly worsen, indicating overfitting at high rank. Fig. 1b further shows that RRR yields substantial compression, from 3.7× to 120×, while maintaining performance comparable to or slightly better than the full-rank OLS solution: RRR matches or improves on OLS in 15 of the 20 cases, and the relative improvement in test MSE ranges from -0.55% to $+6.18\%$. Compression generally increases with the horizon and is consistently high on Weather, where ranks $r^* \leq 7$ match the full-rank accuracy at all horizons. The largest accuracy gains occur on ETTh1 for $H \leq 336$ and on ETTh2 at $H = 720$. These results suggest that low-rank approximations capture essential time-varying patterns efficiently, which is consistent with recent analysis of linear TSF [Wang et al., 2026]. The benefit of low-rank representations is further validated in Section 5.2.1.

3.2 Kernel Regression

We now turn to kernel analysis to obtain an alternative characterization of the forecasting model based on risk minimization [Shawe-Taylor and Cristianini, 2004]:

$$\hat{\mathbf{y}}_i = [\hat{x}_{i,L+1}, \ldots, \hat{x}_{i,L+H}]^\intercal = \underbrace{\mathbf{K_{yx}}(\mathbf{K_{xx}} + \lambda \mathbf{I})^{-1}}_{\mathbf{H}} \mathbf{x}_i, \quad \text{for } i \in \{1, \ldots, N\}, \tag{5}$$

where λ is a regularization parameter, and $\mathbf{K_{yx}} \in \mathbb{R}^{H \times L}$ and $\mathbf{K_{xx}} \in \mathbb{R}^{L \times L}$ are Gram matrices. Their entries are given by $[\mathbf{K_{yx}}]_{nm} = \mathscr{K}(y_{i,n}, x_{i,m})$ and $[\mathbf{K_{xx}}]_{nm} = \mathscr{K}(x_{i,n}, x_{i,m})$, where $\mathscr{K}(x, x') : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ denotes a positive-definite kernel function. The following proposition characterizes the complexity of this model.

Proposition 2. *Consider the forecasting model in Eq.* (5) *associated with a positive-definite kernel function $\mathscr{K}(x, x')$ with a bounded RKHS norm. Assume that the samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ are treated as i.i.d. draws from $P_{\boldsymbol{x},\boldsymbol{y}}(\cdot, \cdot)$ and that the corresponding eigenfunctions are bounded by a common constant. The n-th prediction sample at time i of the empirical forecasting model $\hat{y}_{i,n} = \hat{x}_{i,L+n}$ complies with a concentration bound as*

$$\Pr\left(\left|\hat{y}_{i,n} - \sum_{l=1}^{L}\sum_{k=1}^{K} \gamma_{k,n}\, f_{k,l}\, x_{i,L-l+1}\right| \geq s\right) \leq 2\exp\left(-\frac{L^3\, s^2}{B_0^6\, g(\mathbf{x}_i, K)}\right), \tag{6}$$

with $n \in \{1, \ldots, H\}$, where $\left\{\mathbf{f}_k = [f_{k,1}, \ldots, f_{k,L}]\right\}_{k=1}^K$ are horizon-invariant filters, $\{\gamma_{k,n} \in \mathbb{R}\}_{n=1}^H$ horizon-varying coefficients of the k-th filter, s the confidence parameter, K an integer value determining the number of filters, B_0 the maximum upper-bound of the eigenvalues of $\boldsymbol{K_{xx}}$, and $g(\boldsymbol{x}_i, K) := K^2 \sum_{l=1}^{L} x_{i,l}$.

Proof Sketch (see Appendix A for a full proof). Applying Mercer's theorem yields $\mathscr{K}(x, x') = \sum_{k=1}^{K} \mu_k\, \phi_k(x)\, \phi_k(x')$, where $\{\mu_k\}_{k=1}^K$ and $\{\phi_k(\cdot)\}_{k=1}^K$ are the eigenvalues and eigenfunctions of the integral operator associated with the kernel function $\mathscr{K}(x, x')$. $\mathbf{K}_{xx}$ and its regularized inverse $(\mathbf{K}_{xx} + \lambda I)^{-1}$ share eigenvectors $\frac{1}{\sqrt{L}}\phi_k$ with eigenvalues $1/(L\,\mu_k + \lambda)$. Then, for a confidence interval s and with probability at least $1 - 2\exp\left(-\frac{L^3\, s^2}{B_0^6\, g(\mathbf{x}_i, K)}\right)$, we obtain

$$\hat{y}_{i,n} = \sum_{l=1}^{L}\left[\mathbf{K}_{yx}(\mathbf{K}_{xx} + \lambda I)^{-1}\right]_{n,l} x_{i,l} = \sum_{k=1}^{K} \frac{\mu_k / L}{\mu_k + \lambda / L}\phi_k(y_n) \sum_{l=1}^{L} \phi_k(x_l)\, x_{i,l} = \sum_{k=1}^{K}\sum_{l=1}^{L} \gamma_{k,n} f_{k,l}\, x_{i,L-l+1},$$

where $[\mathbf{K}_{yx}]_{n,l} = \mathscr{K}(y_n, x_l)$ is the cross-kernel matrix between output and input samples. This representer-type construction serves to analyze the structure of the prediction operator and is not a prediction procedure: the future values enter only through the horizon-dependent coefficients $\gamma_{k,n}$, which, like the filters, are learned from training data in the model below, so that at inference time the prediction uses only the look-back window $\mathbf{x}_i$. □

Remark 3. Proposition 2 yields an approximation whose concentration bound vanishes exponentially as L increases, thereby ensuring high accuracy for sufficiently large L. For large L, we have:

$$\hat{y}_{i,n} = \sum_{l=1}^{L}\sum_{k=1}^{n_f} \gamma_{k,n}\, f_{k,l}\, x_{i,L-l+1}, \quad \text{for } n \in \{1, ..., H\}, \tag{7}$$

where we replace K with n_f to emphasize its role as the number of filters in this kernel-based model. This also highlights that n_f specifies the RKHS norm constraint of the model: decreasing n_f tightens the RKHS norm (see Appendix B).

We now formulate the optimization problem for the kernel-based forecasting model in Eq. (7):

$$\mathrm{P}_{\mathrm{ker}}: \qquad \min_{\boldsymbol{\Gamma},\, \mathbf{F}} \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{\Gamma}^\intercal \mathbf{F}\, \mathbf{x}_i - \mathbf{y}_i\|^2, \tag{8}$$

where $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_{n_f}]^\intercal$ with $[\boldsymbol{\gamma}_j \in \mathbb{R}^H]_i = \gamma_{j,i}$, is the coefficient matrix, and $\mathbf{F} = [\mathbf{f}_1^\intercal, \ldots, \mathbf{f}_{n_f}^\intercal]^\intercal \in \mathbb{R}^{n_f \times L}$ is the filter matrix. Note that the forecasting model of $\mathrm{P}_{\mathrm{ker}}$, $\hat{\mathbf{y}}_i = \boldsymbol{\Gamma}^\intercal\, \mathbf{F}\, \mathbf{x}_i$, requires optimizing $n_f(L + H)$ parameters. According to Proposition 2, n_f specifies the capacity of the function class $\mathscr{H}$ and thus plays the same role as parameter r in problem $\mathrm{P}_{\mathrm{RRR}}$. Specifically, the rank of $\boldsymbol{\Gamma}^\intercal\mathbf{F}$ is at most n_f, with equality holding as stated in the following corollary.

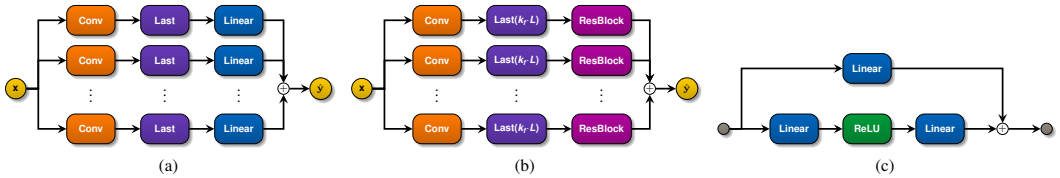


Figure 2: (a) Low-rank filter bank architecture with n_f branches (FB-LR). (b) Deep filter bank architecture (FB-deep) retaining a fraction k_f of the samples. (c) Residual block used in FB-deep.

Corollary 4. *The kernel regression problem P_{ker} is equivalent to the rank-constrained regression problem P_{RRR} for the case $n_f = r$, which can be simply recovered by replacing $\mathbf{\Gamma}^\mathsf{T}\mathbf{F}$ with $\mathbf{W}$.*

3.3 Summary and Implications on Model Design

The analysis above shows that constraining model complexity can improve generalization, and that TSF problems often admit much lower intrinsic complexity than standard linear models assume. Kernel regression further reveals a decomposition into horizon-invariant filters and horizon-varying coefficients [Eq. (7)], naturally leading to rank-constrained models (Corollary 4).

However, both RRR and traditional kernel methods have practical limitations: RRR is limited to linear transformations, whereas kernel methods require manual selection of functions and hyperparameters that depend on prior knowledge of the dataset.

Neural networks overcome these limitations while preserving the structural insights above: they can learn horizon-invariant filters $\{\mathbf{f}_i\}_{i=1}^{n_f}$ and horizon-varying coefficients $\{\boldsymbol{\gamma}_j\}_{j=1}^{n_f}$ through gradient-based optimization, without manual kernel selection. We design such architectures next.

4 Filter Bank Architectures

We now design neural network architectures that explicitly leverage these insights to learn representations corresponding to the expressions derived in the previous section. We seek architectures that realize the filter-based decomposition in Eq. (7), control capacity through parameters analogous to r in RRR and n_f in kernel methods, and remain learnable through standard gradient-based optimization without explicitly setting filters or kernels. This design is also aligned with the minimum description length (MDL) principle, according to which simpler models often generalize better [Grünwald, 2007, Krizhevsky et al., 2012, Han et al., 2015].

4.1 Low-Rank Filter Bank

Following Section 3.2, the kernel forecasting model in Eq. (7) can be realized by the *low-rank filter bank* (FB-LR) architecture shown in Fig. 2a. The rows of $\mathbf{F} = [\mathbf{f}_1^\mathsf{T}, \dots, \mathbf{f}_{n_f}^\mathsf{T}]^\mathsf{T}$ define the convolution filters, while the rows of $\mathbf{\Gamma}$ define the linear weights in each branch. The final prediction is the sum of the branch outputs:

$$\hat{\mathbf{y}}_i = \sum_{k=1}^{n_f} \boldsymbol{\gamma}_k \, [\mathbf{f}_k * \mathbf{x}_i]_L, \quad \text{for } i \in \{1, \dots, N\}, \tag{9}$$

where $*$ denotes the discrete convolution operator, yielding $[\mathbf{f}_k * \mathbf{x}_i]_l = \sum_{m=1}^{L} f_{k,K-m+1} \, x_{i,l+m-K}$. This architecture does not depend on a specific kernel function and can be learned directly while retaining the structural insights provided by kernel-regression analysis.

The filter bank implements the model through convolution followed by retaining only the last convolution sample, even though it could also be written as a matrix-vector product [see Eq. (7)]. This reduces parameters by treating the convolution kernel size as a hyperparameter, and naturally extends to retaining a short vector of recent convolution samples. The deep architecture introduced in the next section uses this extension to process each branch output with nonlinear residual blocks.

4.2 A Deep Filter Bank

Building on the preceding observation and prior work [Wu et al., 2021, Das et al., 2023], we design a deep variant of the filter-bank model that preserves its low-rank structure. To this end, we retain $s := k_f L$ convolution samples in each branch and process them with a residual block, as shown in Fig. 2c, where k_f is the *keep fraction* and satisfies $k_f L \in \{1, ..., L\}$. The resulting architecture, shown in Fig. 2b, is called *deep filter bank* (FB-deep). Formally, branch k retains the vector of the last s convolution samples, $\mathbf{z}_{i,k} := \big[[\mathbf{f}_k * \mathbf{x}_i]_{L-s+1}, \ldots, [\mathbf{f}_k * \mathbf{x}_i]_L\big]^\top \in \mathbb{R}^s$, whereas FB-LR retains only $[\mathbf{f}_k * \mathbf{x}_i]_L$. This vector is processed by the residual block $\mathscr{R}_k : \mathbb{R}^s \to \mathbb{R}^H$ of Fig. 2c, given by

$$\mathscr{R}_k(\mathbf{z}) := \mathbf{A}_{k,2}\,\mathrm{ReLU}\big(\mathbf{A}_{k,1}\mathbf{z} + \mathbf{b}_{k,1}\big) + \mathbf{A}_{k,\mathrm{skip}}\mathbf{z} + \mathbf{b}_{k,2}, \tag{10}$$

where, for a hidden dimension d_k, the learned parameters are $\mathbf{A}_{k,1} \in \mathbb{R}^{d_k \times s}$, $\mathbf{b}_{k,1} \in \mathbb{R}^{d_k}$, $\mathbf{A}_{k,2} \in \mathbb{R}^{H \times d_k}$, $\mathbf{A}_{k,\mathrm{skip}} \in \mathbb{R}^{H \times s}$, and $\mathbf{b}_{k,2} \in \mathbb{R}^H$. The FB-deep prediction is then obtained by:

$$\hat{\mathbf{y}}_i = \sum_{k=1}^{n_f} \mathscr{R}_k(\mathbf{z}_{i,k}), \quad \text{for } i \in \{1, \ldots, N\}. \tag{11}$$

The following remark characterizes the rank of the corresponding linearized model.

Remark 5 (Rank of FB-deep)**.** Although rank analysis does not directly apply to nonlinear models, we can assess the rank of FB-deep (Fig. 2b) after removing the ReLU activation in Eq. (10). The prediction of the resulting model at time i follows:

$$\hat{\mathbf{y}}_i = \sum_{k=1}^{n_f} \sum_{m=1}^{s} \boldsymbol{\alpha}_{k,m} \big[\mathbf{f}_k * \mathbf{x}_i\big]_{L-m+1}, \quad \text{for } i \in \{1, \ldots, N\}, \tag{12}$$

where $\boldsymbol{\alpha}_{k,m} \in \mathbb{R}^H$ collects the horizon-varying coefficients associated with the m-th convolution sample for the k-th branch. Comparing Eq. (12) with Eq. (9), we conclude that this model induces a linear form $\mathbf{W}$ as a combination of s filter-bank models with different coefficients, and therefore its rank is upper bounded by

$$\mathrm{rank}(\mathbf{W}) \leq \min\{n_f k_f L,\ H,\ L\}.$$

Note that FB-LR is recovered when $k_f = 1/L$, for which $s = 1$ and the rank is at most n_f. Increasing k_f gradually relaxes this rank constraint in this model. Appendix E shows that the same bound holds for the local linearizations of the nonlinear FB-deep and measures their effective rank.

5 Evaluation

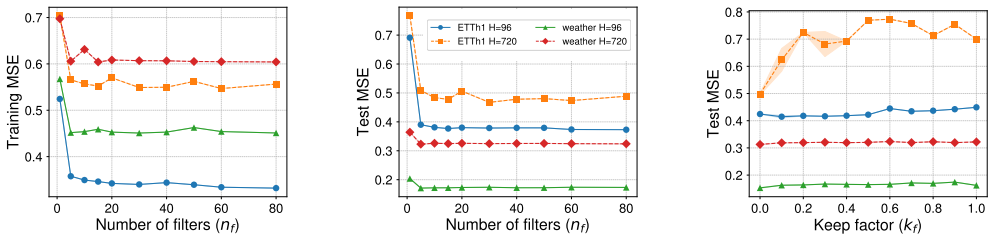
We now evaluate the proposed architectures on the long-term TSF benchmark [Zhou et al., 2021], which comprises diverse real-world datasets, including those already considered in Section 3.1.1. Appendix C provides further details on the datasets and setup, while Appendix D reports supplementary empirical results. The code is available at `https://github.com/ashutoshvai1/FilterBanks`.

5.1 Sensitivity Analysis and Ablation Study

We begin with a sensitivity analysis of the key hyperparameters of the proposed architectures: the number of filters n_f for FB-LR and the keep fraction k_f for FB-deep (Figure 3). We fix the convolution kernel length to $K = 300$ and use a learning rate of 0.01; for the study on k_f, we further set

Table 1: Test MSE for filter-bank variants. Best results in **bold**, second-best on a shaded background.

	FB-LR				FB-deep (k_f=1/L)				FB-deep			
Dataset	96	192	336	720	96	192	336	720	96	192	336	720
Weather	0.167	0.209	0.254	0.318	**0.145**	**0.185**	**0.235**	**0.317**	0.148	0.192	0.241	0.319
ETTh1	**0.371**	**0.405**	**0.441**	**0.473**	0.415	0.447	0.492	0.545	0.400	0.454	0.441	0.478
ETTh2	**0.271**	**0.346**	0.391	**0.422**	0.306	0.365	0.391	0.452	0.291	0.376	**0.382**	0.451
ETTm1	0.315	0.338	**0.371**	**0.414**	0.300	0.356	0.390	0.440	**0.292**	**0.338**	0.373	0.440
ETTm2	**0.161**	**0.216**	**0.270**	**0.360**	0.174	0.236	0.287	0.369	0.172	0.231	0.286	0.367
Traffic	0.387	0.417	0.425	0.463	0.476	0.527	0.539	0.554	**0.360**	**0.381**	**0.395**	**0.430**
Electricity	0.134	0.153	0.169	0.209	0.165	0.183	0.199	0.242	**0.128**	**0.148**	**0.159**	**0.196**



(a) (b) (c)

Figure 3: (a) Train and (b) test MSE versus n_f for FB-LR. (c) Test MSE vs. keep fraction k_f for FB-deep.

$n_f = 20$. Each configuration is trained with five different random seeds, and the plots report mean values with standard deviations as shaded regions where non-negligible.

Fig. 3a shows that the training loss consistently decreases as n_f increases, reflecting greater model expressiveness. In contrast, Fig. 3b shows that the test loss drops rapidly for small n_f, then saturates or slightly worsens as n_f grows. This pattern matches the overfitting behavior in Fig. 1a, where the loss is expressed as a function of rank. Saturation occurs earlier for Weather (around $n_f \approx 5$) than for ETTh1, which is consistent with Fig. 1b, where Weather has small optimal ranks ($r^* \leq 7$ at all horizons). Finally, Fig. 3c indicates that smaller values of k_f generally yield better generalization, particularly for ETTh1, whereas Weather is relatively insensitive to this hyperparameter in the considered configuration.

We continue with an ablation study comparing three architectural configurations: FB-LR, FB-deep with the sampling fraction fixed at its minimum value (i.e., $k_f = 1/L$), and FB-deep with tunable k_f. The convolution and last blocks of FB-deep with $k_f = 1/L$ exactly match those of FB-LR. Hyperparameters were tuned on validation data via the Tree-structured Parzen Estimator algorithm [Bergstra et al., 2011]. Table 1 reports the test MSE for these variants. The results highlight the benefit of treating k_f as a tunable hyperparameter, as FB-deep achieves the best or second-best accuracy in 26 of the 28 dataset–horizon pairs. The table also reveals a dataset-dependent pattern: larger datasets (Traffic and Electricity) benefit from FB-deep with tunable k_f, whereas the smaller ETT datasets generally favor the simpler FB-LR. Weather is an interesting exception, where fixing k_f to its minimum value outperforms tuning it. This is consistent with our earlier finding that Weather has small optimal RRR ranks at all horizons – increasing the keep fraction raises the rank of the learned representation and degrades generalization. Overall, the selected k_f and n_f values are primarily dataset-dependent capacity controls, with larger datasets generally favoring larger values.

5.2 Comparison with State of the Art

We now compare the accuracy and efficiency of FB-LR and FB-deep against representative methods from different categories in the state of the art: SparseTSF [Lin et al., 2024], a low-complexity model with a lightweight processing pipeline; DLinear [Zeng et al., 2023], a linear baseline; TimeKAN [Huang et al., 2025], based on Kolmogorov–Arnold networks; TimeMixer [Wang et al., 2024], a channel-mixing architecture with multi-layer perceptrons; iTransformer [Liu et al., 2024] and PatchTST [Nie et al., 2023], both transformer-based models; and TimePro [Ma et al., 2025], a state-space model. Appendix D.1 broadens this comparison with additional baselines, and Appendix D.6 evaluates the models on two financial datasets.

5.2.1 Prediction Accuracy

Table 2 reports the MSE for the considered datasets and different prediction horizons. Our models use a look-back window of $L = 720$, as does SparseTSF [Lin et al., 2024], whereas the other baselines use the look-back windows of their reported results (see Appendix C). In addition, Appendix D.2 repeats the comparison with the same look-back window ($L = 720$) for all models, while Appendix D.5 reports the variation of our models across random seeds. First, comparing the low-rank FB-LR with the full-rank DLinear baseline, we observe that the former performs better in 78.6% of the cases (22 of 28 dataset–horizon pairs). This indicates that enforcing a low-rank struc-

Table 2: Test MSE for long-term TSF on benchmark datasets. Best results in **bold**, second-best are shaded.

Dataset	Horizon	FB-deep (Ours)	FB-LR (Ours)	SparseTSF (ICML'24)	DLinear (AAAI'23)	TimeKAN (ICLR'25)	TimeMixer (ICLR'24)	iTransformer (ICLR'24)	PatchTST (ICLR'23)	TimePro (ICML'25)
Weather	96	**0.148**	0.167	0.172	0.176	0.162	0.163	0.174	0.149	0.166
	192	**0.192**	0.209	0.215	0.218	0.207	0.211	0.221	0.194	0.216
	336	**0.241**	0.254	0.260	0.262	0.263	0.263	0.278	0.245	0.273
	720	0.319	0.318	0.318	0.323	0.338	0.344	0.358	**0.314**	0.351
Electricity	96	**0.128**	0.134	0.138	0.140	0.174	0.153	0.148	0.129	0.139
	192	**0.148**	0.153	0.151	0.153	0.182	0.166	0.162	0.149	0.156
	336	**0.159**	0.169	0.166	0.169	0.197	0.185	0.178	0.166	0.172
	720	**0.196**	0.209	0.205	0.204	0.236	0.224	0.225	0.210	0.209
Traffic	96	**0.360**	0.387	0.389	0.413	–	0.462	0.395	0.366	–
	192	**0.381**	0.417	0.398	0.423	–	0.473	0.417	0.388	–
	336	**0.395**	0.425	0.411	0.437	–	0.498	0.433	0.398	–
	720	**0.430**	0.463	0.448	0.466	–	0.506	0.467	0.457	–
ETTm1	96	**0.292**	0.315	0.314	0.299	0.322	0.317	0.334	0.293	0.326
	192	0.338	0.338	0.343	0.335	0.357	0.367	0.377	**0.333**	0.367
	336	0.373	0.371	**0.369**	**0.369**	0.382	0.391	0.426	**0.369**	0.402
	720	0.440	**0.414**	0.418	0.425	0.445	0.454	0.491	0.416	0.469
ETTm2	96	0.172	**0.161**	0.165	0.167	0.174	0.175	0.180	0.166	0.178
	192	0.231	**0.216**	0.218	0.221	0.239	0.240	0.250	0.223	0.242
	336	0.286	**0.270**	0.272	0.274	0.301	0.303	0.311	0.274	0.303
	720	0.367	0.360	**0.350**	0.368	0.395	0.392	0.412	0.362	0.400
ETTh1	96	0.400	0.371	**0.362**	0.384	0.367	0.385	0.386	0.385	0.375
	192	0.454	0.405	**0.403**	0.443	0.414	0.443	0.441	0.413	0.427
	336	0.441	0.441	**0.434**	0.446	0.445	0.512	0.487	0.440	0.472
	720	0.478	0.473	**0.426**	0.504	0.444	0.497	0.503	0.456	0.476
ETTh2	96	0.291	**0.271**	0.294	0.282	0.290	0.289	0.297	0.274	0.293
	192	0.376	0.346	0.339	0.350	0.375	0.378	0.380	**0.338**	0.367
	336	0.382	0.391	**0.359**	0.414	0.423	0.432	0.428	0.367	0.419
	720	0.451	0.422	**0.383**	0.588	0.443	0.464	0.427	0.391	0.427

ture improves performance across datasets, consistent with recent results on low-rank regularization as a useful inductive bias [Wang et al., 2026].

The accuracy of the different architectures also provides insight into which model classes are best suited to particular datasets. Weather, Electricity, and Traffic are the three largest datasets in the benchmark and generally favor deep architectures such as PatchTST over linear models. In particular, FB-deep achieves the lowest MSE at all prediction horizons on these datasets except Weather at horizon 720, demonstrating its effectiveness on large-scale forecasting tasks. Traffic shows the clearest benefit from the low-rank bias among the larger benchmarks, whereas Electricity benefits more from the relaxed-rank deep variant. Conversely, low-complexity models generally perform better on the ETT datasets. SparseTSF performs particularly well on ETTh1 and ETTh2, while FB-LR excels on ETTm2. For ETTm1, PatchTST obtains the lowest MSE in two of the four prediction horizons, whereas FB-LR achieves the best result for the 720-sample horizon. The largest gap against our models occurs on ETTh1 at horizon 720, where SparseTSF's periodicity-oriented inductive bias appears especially favorable.

Overall, FB-LR and FB-deep achieve accuracy competitive with the state of the art at a substantially lower computational cost (Section 5.2.2), while being much simpler than transformers, mixers, KANs, and state-space models. With the same look-back window ($L = 720$) for all models (Appendix D.2), some rankings change, but FB-deep remains among the two best models at all horizons on Electricity and Traffic, whereas on the ETT datasets FB-LR and SparseTSF are each among the two best models in 9 of the 16 settings; FB-LR also outperforms DLinear in 19 of the 28 dataset–horizon pairs.

Table 3: Cost of TSF models on ETTm1 with $L = 720$ and the configurations of Table 2. Times are median $\pm$ standard deviation over 10,000 forward passes; memory is the peak allocated GPU memory.

Model	Parameters	MACs	Memory (MiB)	RTX 3080 time (ms)	Jetson time (ms)
FB-LR (Ours)	35.3k	241.9k	8.4	0.36 ± 0.03	1.60 ± 0.10
SparseTSF	32.4k	932.4k	8.3	0.22 ± 0.03	0.94 ± 0.02
DLinear	1.0M	7.3M	12.2	0.27 ± 0.03	2.10 ± 0.32
FB-deep (Ours)	1.5M	11.0M	14.3	0.62 ± 0.03	2.67 ± 0.15
PatchTST	8.7M	357.1M	57.1	2.18 ± 0.12	13.48 ± 0.11
TimeKAN	526.9k	59.6M	15.7	7.30 ± 1.70	26.03 ± 0.68
TimeMixer	4.0M	477.3M	29.0	5.64 ± 0.86	24.87 ± 0.47
TimePro	238.7M	1.45G	925.5	3.57 ± 0.52	–

5.2.2 Computational Efficiency

We finally characterize the efficiency of the considered models in terms of learnable parameters, total number of multiply-accumulate operations (MACs), inference time (lower is better), and memory usage during inference at batch size 1 in FP32. Appendix D.3 reports resource usage during training.

Table 3 reports these quantities for the ETTm1 dataset for $L = 720$ and unitary batch size. We measure inference time and memory usage on both a workstation-class GPU (an NVIDIA RTX 3080 Ti) and an embedded-class platform (a Jetson Orin AGX device with a 15W power profile).

The results show that FB-LR requires $3.85\times$ fewer MACs than SparseTSF, a highly parameter-efficient architecture, despite having a slightly higher number of parameters. This difference arises because FB-LR leverages low-rank matrix products to reduce both parameter count and computation, whereas SparseTSF relies on weight sharing, which lowers the number of parameters but leaves the number of MACs unchanged. SparseTSF is still slightly faster in measured latency, but both models remain below 1 ms on the RTX 3080 Ti and below 2 ms on the Jetson device. FB-deep offers a compelling tradeoff between linear and deep models by making effective use of additional complexity. It uses about $32.5\times$ fewer MACs than PatchTST, reaches sub-millisecond RTX 3080 Ti inference latency, and runs about $5\times$ faster than PatchTST on the Jetson Orin AGX. This shows that the filter bank models are efficient also on a low-power edge platform.

6 Conclusion

This work studied low-rank structure in TSF theoretically and through practical architectures. Our analysis based on reduced-rank regression and kernel regression indicates that popular TSF benchmarks exhibit an intrinsically low-dimensional structure, thereby motivating rank-constrained forecasting models. Guided by these insights, we introduced the filter bank architecture, which expresses low-rank constraints through tunable hyperparameters and extends to deeper models. Empirically, filter bank models deliver competitive performance at substantially lower computational cost, with FB-LR performing strongly on the ETT datasets and FB-deep on the larger benchmarks.

Our analysis also shows how the kernel eigenspectrum shapes the filter-bank structure, complementing Gaussian-process interpretations of closed-form linear models [Yugay et al., 2026]. Low-rank structures are often used for compression or adaptation [Idelbayev and Carreira-Perpinán, 2020, Hu et al., 2022], but our results suggest that they also improve generalization when trained from scratch.

Limitations. The rank analysis presented here holds exactly for linear models such as FB-LR. For the nonlinear FB-deep, the rank bound applies to the linear map that the model computes around each input, rather than to the model as a whole (Appendix E.2). Similarly, the kernel analysis assumes that training windows are independent samples, whereas real-world time series exhibit temporal dependence (Proposition 2). Moreover, both the analysis and the models treat each channel separately; capturing cross-channel interactions would require extending the notion of rank jointly over time and channels. Finally, this work does not investigate how dataset properties (e.g., spectrum and autocorrelation) relate to the optimal filter-bank capacity. Promising directions for future work include establishing this relation, integrating filter bank-style blocks with other backbones (e.g., attention, mixers, and state space models), and developing theory-guided hyperparameter selection.

Acknowledgments and Disclosure of Funding

This work was supported by the Research Council of Finland (formerly Academy of Finland) under grants no. 354322 and no. 355577. We acknowledge CSC – IT Center for Science, Finland, for computational resources and for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through CSC. The authors declare no competing interests.

References

Theodore W Anderson. *The statistical analysis of time series*. John Wiley & Sons, 2011.

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36, 2022.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Henrik Brink, Joseph Richards, and Mark Fetherolf. *Real-world machine learning*. Simon and Schuster, 2016.

David Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Rates of convergence for sparse variational Gaussian process regression. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871. PMLR, Jun 2019.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. doi: 10.1007/s10208-009-9045-5.

Abhimanyu Das, Weihao Kong, Andrew Leach, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam Nguyen, Wesley M Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *Advances in Neural Information Processing Systems*, 37:74147–74181, 2024.

Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007. ISBN 978-0-262-07262-6.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Yaping He, Linhao Jiang, and Di Wu. Convolutional neural network compression based on low-rank decomposition, 2024. URL https://arxiv.org/abs/2408.16289.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2106.09685.

Songtao Huang, Zhen Zhao, Can Li, and Lei Bai. TimeKAN: KAN-based frequency decomposition learning architecture for long-term time series forecasting. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2502.06910. arXiv preprint arXiv:2502.06910.

Yerlan Idelbayev and Miguel A Carreira-Perpinán. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8049–8059, 2020.

Alan Julian Izenman. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5(2):248–264, 1975.

Xuebo Jin, Xinghong Yu, Xiaoyi Wang, Yuting Bai, Tingli Su, and Jianlei Kong. Prediction for time series with CNN and LSTM. In *Proceedings of the 11th international conference on modelling, identification and control (ICMIC2019)*, pages 631–641. Springer, 2020.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vladimir Koltchinskii and Evarist Giné. Random matrix approximation of spectra of integral operators. *Bernoulli*, 6(1):113–167, 2000.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. MCUNet: Tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems*, 33, 2020.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. SparseTSF: modeling long-term time series forecasting with 1k parameters. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and C. L. Philip Chen. SparseTSF: Lightweight and robust time series forecasting via sparse modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(1):170–183, 2026. doi: 10.1109/TPAMI.2025.3602445.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations (ICLR 2024)*, 2024. URL https://arxiv.org/abs/2310.06625.

Aitian Ma, Dongsheng Luo, and Mo Sha. MixLinear: Extreme low resource multivariate time series forecasting with 0.1K parameters. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2410.02081.

Xiaowen Ma, ZhenLiang Ni, Shuai Xiao, and Xinghao Chen. TimePro: Efficient multivariate long-term time series forecasting with variable- and time-aware hyper-state. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, volume 267 of *Proceedings of Machine Learning Research*, pages 42096–42111, 2025. URL https://proceedings.mlr.press/v267/ma25p.html.

J. Mercer. Functions of positive and negative type, and their connection the theory of integral equations. *Philosophical Transactions of the Royal Society of London.*, 209:415–446, 1909.

Gilles Meyer, Silère Bonnabel, and Rodolphe Sepulchre. Linear regression under fixed-rank constraints: A Riemannian approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA, USA, 2011.

Mehryar Mohri. *Foundations of Machine Learning*. MIT Press, Cambridge, MA, 2018. ISBN 9780262039406.

Ashin Mukherjee. *Topics on Reduced Rank Methods for Multivariate Regression.* PhD thesis, 2013.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Aileen Nielsen. *Practical time series analysis: Prediction with statistics and machine learning*. O'Reilly Media, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *European Signal Processing Conference (EUSIPCO)*, pages 606–610, 2007.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004. ISBN 978-0521813974. doi: 10.1017/CBO9780511809682.

Yang Sui, Miao Yin, Wanzhao Yang, Yu Gong, Jinqi Xiao, Huy Phan, Ding Ding, Xiaozhong Xu, Shan Liu, Zhenzhong Chen, and Bo Yuan. ELRT: Towards efficient low-rank training for compact neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=TC39w69m8bB.

Cheng Tai, Tong Xiao, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization. In *International Conference on Learning Representations (ICLR), poster track*, 2016. URL https://arxiv.org/abs/1511.06067.

William Toner and Luke Darlow. An analysis of linear time series forecasting models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, July 2024. URL https://arxiv.org/abs/2403.14587.

Joel A. Tropp and Robert J. Webber. Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications. *arXiv preprint arXiv:2306.12418*, 2023. URL https://arxiv.org/abs/2306.12418.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=7oLshfEIC2.

Zheng Wang, Kaixuan Zhang, Wanfang Chen, Xiaonan Lu, Longyuan Li, and Tobias Schlagenhauf. Characteristic root analysis and regularization for linear time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.23597.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is Mamba effective for time series forecasting? *Neurocomputing*, 619:129178, 2025. doi: 10.1016/j.neucom.2024.129178.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling time series with 10k parameters. *arXiv preprint arXiv:2307.03756*, 2023. URL https://arxiv.org/abs/2307.03756.

Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. FilterNet: Harnessing frequency filters for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2411.01623.

Mingzhou Yin and Matthias A. Müller. Low-rank matrix regression via least-angle regression. *arXiv preprint arXiv:2503.10569*, 2025. URL https://arxiv.org/abs/2503.10569.

Aleksandr Yugay, Hang Cui, Changhua Pei, and Alexey Zaytsev. Strong linear baselines strike back: Closed-form linear models as Gaussian process conditional density estimators for TSAD, 2026. URL https://arxiv.org/abs/2602.00672.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, pages 11121–11128, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai 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.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022.

A Proof of Proposition 2

Proof. Without loss of generality, we denote the n-th input sample at time i, i.e., $x_{i,n}$, by simply x_n, and the n-th prediction sample at time i, i.e., $\hat{y}_{i,n}$, by $\hat{y}_n$. Mercer's theorem states that kernel functions can be represented as follows [Mercer, 1909]:

$$\mathscr{K}(x, x') = \sum_{k=1}^{K} \mu_k \phi_k(x) \phi_k(x') \tag{13}$$

where $\{\mu_k\}_{k=1}^K$ and $\{\phi_k(\cdot)\}_{k=1}^K$ are the eigenvalues and orthonormal eigenfunctions of the integral operator $\mathscr{I}_{\mathscr{K}}[\cdot]$:

$$\mathscr{I}_{\mathscr{K}}[f(x')] = \int f(x) \mathscr{K}(x, x') p(x) dx,$$

with $p(\cdot)$ being the distribution from which the inputs are assumed to be drawn. We construct the kernel matrix $\mathbf{K_{xx}}$, where $\mathbf{x} \in \mathbb{R}^L$ is drawn from $p(\cdot)$ and $[\mathbf{K_{xx}}]_{nm} = \mathscr{K}(x_n, x_m)$. For a bounded kernel function $\mathscr{K}(\cdot, \cdot)$, the sequence of eigenvalues-eigenvectors of $1/L\mathbf{K_{xx}}$ converges to the sequence of eigenvalues-eigenfunctions of the operator $\mathscr{I}_{\mathscr{K}}[\cdot]$ [Koltchinskii and Giné, 2000].

We now recall the model in Eq. 5, where the function class of interest $\mathscr{H}$ is associated with the kernel function $\mathscr{K}(\cdot, \cdot)$, and vectors $\mathbf{x}$ and $\mathbf{y}$ are assumed to be independently drawn from the distribution $p(\cdot)$. This i.i.d. abstraction follows the standard supervised-regression setting used in classical RKHS analyses and recent analyses of linear TSF models [Shawe-Taylor and Cristianini, 2004, Mohri, 2018, Toner and Darlow, 2024]; here it serves as an idealized model of the forecasting task rather than a literal claim about temporal independence. For this setting, Matern kernels with inputs uniformly sampled can be considered [Burt et al., 2019]. Let us denote the eigenvalues-eigenfunctions of the corresponding integral operator $\mathscr{I}_{\mathscr{K}}[\cdot]$ by $\{\mu_k\}_k$–$\{\phi_k(\cdot)\}_k$. Then, as stated above, the eigenvalues of $\mathbf{K_{xx}}$ converge to $\{L\,\mu_k\}_k$, and the eigenvalues of $(\mathbf{K_{xx}} + \lambda\mathbf{I})^{-1}$ converge to $\{1/(L\,\mu_k + \lambda)\}_k$. Moreover, the eigenvectors of $\mathbf{K_{xx}}$, denoted by $\{\mathbf{u}_k\}_k$, are the same as the eigenvectors of $(\mathbf{K_{xx}} + \lambda\mathbf{I})^{-1}$, and converge to $\{\frac{1}{\sqrt{L}}\phi_k(\cdot)\}_k$. Using these facts and Eq. 13, we can write:

$$\left[(\mathbf{K_{xx}} + \lambda\mathbf{I})^{-1}\right]_{m,l} := \sum_{k=1}^{K} (L\mu_k + \lambda)^{-1} [\mathbf{u}_k^\intercal]_m [\mathbf{u}_k]_l := \frac{1}{L} \sum_{k=1}^{K} (L\mu_k + \lambda)^{-1} \phi_k(x_m) \phi_k(x_l).$$

According to Eq. 13, we also obtain:

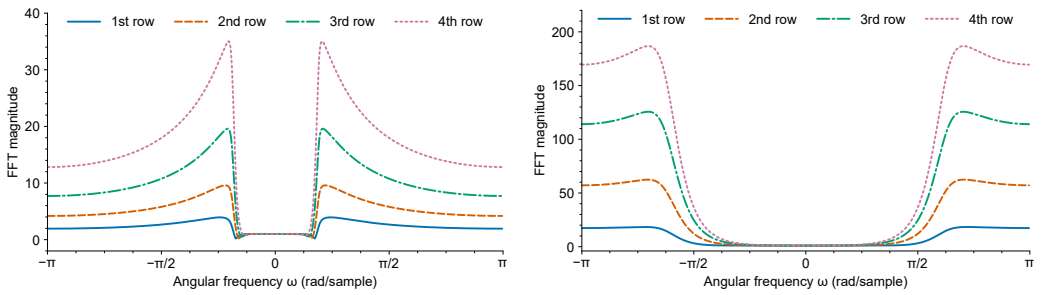
$$\left[\mathbf{K_{yx}}\right]_{n,m} = \mathscr{K}(y_n, x_m) = \frac{1}{L} \sum_{k=1}^{K} L\,\mu_k \phi_k(y_n) \phi_k(x_m).$$

The cross-kernel matrix $\mathbf{K_{yx}}$ involves the future values y_n. The operator $\mathbf{H}$ below is therefore a theoretical representer construction used to analyze the structure of the prediction operator, not an oracle procedure that uses future values at test time. As shown below, the future values enter only through the horizon-dependent coefficients $\gamma_{k,n}$ in Eq. (17). In the kernel-based model of Problem $\mathrm{P_{ker}}$ in Eq. (8), these coefficients and the filters are learned from the training data, so that at inference time the forecasting models use only the observed look-back window $\mathbf{x}_i$. Therefore, for the kernel-based model with $\mathbf{H} = \mathbf{K_{yx}}(\mathbf{K_{xx}} + \lambda\mathbf{I})^{-1}$, we obtain:

$$\begin{aligned}
\left[\mathbf{H}\right]_{n,l} &= \sum_{m=1}^{L} [\mathbf{K_{yx}}]_{n,m} \left[(\mathbf{K_{xx}} + \lambda\mathbf{I})^{-1}\right]_{m,l} = \frac{1}{L} \sum_{m=1}^{L} \sum_{k=1}^{K} \mu_k \phi_k(y_n) \phi_k(x_m) \sum_{k'=1}^{K} (L\mu_{k'} + \lambda)^{-1} \phi_{k'}(x_m) \phi_{k'}(x_l) \\
&= \frac{1}{L^2} \sum_{k=1}^{K} \sum_{k'=1}^{K} \mu_k (\mu_{k'} + \lambda/L)^{-1} \phi_k(y_n) \phi_{k'}(x_l) \sum_{m=1}^{L} \phi_k(x_m) \phi_{k'}(x_m)
\end{aligned}$$

Therefore, we obtain the following expression for the n-th prediction:

$$\begin{aligned}
\hat{y}_n &= \sum_{l=1}^{L} [\mathbf{H}]_{n,l}\, x_l = \sum_{l=1}^{L} [\mathbf{H}]_{n,L-l+1}\, x_{L-l+1} \\
&= \frac{1}{L^2} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{k'=1}^{K} \mu_k (\mu_{k'} + \lambda/L)^{-1} \phi_k(y_n) \phi_{k'}(x_{L-l+1}) x_{L-l+1} \sum_{m=1}^{L} \phi_k(x_m) \phi_{k'}(x_m)
\end{aligned} \tag{14}$$



(a) (b)

Figure 4: Row-wise visualization of $\mathbf{H}$ in the frequency domain for $L = 600$ and $H = 4$, obtained with RBF kernel $\mathscr{K}(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/(2L_{\mathrm{RBF}}^2))$ and RQ kernel $\mathscr{K}(\mathbf{x}, \mathbf{x}') = (1+\|\mathbf{x}-\mathbf{x}'\|^2/(a_{\mathrm{RQ}}L_{\mathrm{RQ}}^2))^{-a_{\mathrm{RQ}}}$. Each curve can be regarded as a linear combination of several *row-invariant* filters. The examples were obtained with: (a) two RBF kernels with hyper-parameters $L_{\mathrm{RBF},1} = 10$, $L_{\mathrm{RBF},2} = 15$ and (b) one RQ kernel with hyper-parameters $L_{\mathrm{RQ}} = 6$, $a_{\mathrm{RQ}} = 4$.

To simplify this expression further, we need to characterize $\sum_{m=1}^{L} \phi_k(x_m)\phi_{k'}(x_m)$ in Eq. 14. First, note that

$$\mathbb{E}\left\{\frac{1}{L}\sum_{m=1}^{L}\phi_k(x_m)\phi_{k'}(x_m)\right\} = \int \phi_k(x)\phi_{k'}(x)p(x)dx = \delta_{k,k'},$$

where $\delta_{k,k'}$ is Kronecker delta function, and the last equality holds due to the orthonormality of $\{\phi_k(\cdot)\}_k$. By Hoeffding's inequality, which applies here because the samples are treated as independent and the eigenfunctions are uniformly bounded, we then have

$$\Pr\left(\left|\frac{1}{L}\sum_{m=1}^{L}\phi_k(x_m)\phi_{k'}(x_m) - \delta_{k,k'}\right| \geq t\right) \leq 2\exp\left(-\frac{L\,t^2}{B_0^4}\right), \tag{15}$$

where t is the confidence parameter, and B_0 is the maximum upper bound of the eigenfunctions of the operator $\mathscr{I}_{\mathscr{K}}$ (or equivalently that of eigenvalues of $\mathbf{K_{xx}}$), i.e., $|\phi_k(x)| \leq B_0$. We can then propagate the upper bound in Eq. 15 to establish an approximation rate for the prediction $\hat{y}_n$ in Eq. 14. Accordingly, we obtain:

$$\Pr\left(\left|\hat{y}_n - \sum_{l=1}^{L}\sum_{k=1}^{K}\gamma_{k,n}\, f_{k,l}\, x_{L-l+1}\right| \geq s\right) \leq 2\exp\left(-\frac{L^3\, s^2}{B_0^6\, g(\mathbf{x}, K)}\right), \tag{16}$$

where

$$\gamma_{k,n} := \mu_k(\mu_k + \lambda/L)^{-1}\phi_k(y_n), \qquad f_{k,l} := \frac{1}{L}\phi_k(x_{L-l+1}), \tag{17}$$

and $g(\mathbf{x}, K) := K^2\sum_{l=1}^{L} x_l$ is a function that depends on the input data. To derive Eq. 16, we use the following expression:

$$\sum_{k=1}^{K}\sum_{k'=1}^{K}\mu_k(\mu_{k'} + \lambda/L)^{-1}\phi_k(y_n)\phi_{k'}(x_{L-l+1}) = \sum_{k=1}^{K}\mu_k(\mu_k + \lambda/L)^{-1}\phi_k(y_n)\phi_k(x_{L-l+1})$$

This proves the statement. □

We further provide a numerical evaluation of this proposition. Fig. 4 illustrates the matrix $\mathbf{H} = \mathbf{K_{yx}}(\mathbf{K_{xx}} + \lambda\mathbf{I})^{-1}$ for several kernel functions.

B Impact of n_f

The regularization coefficient λ in Eq. 5 determines the capacity of the function class $\mathscr{H}$ within the RKHS. Recall the definition of $\gamma_{k,n} =: \frac{\mu_k}{\mu_k+\lambda/L}\phi_k(y_n)$ in Eq.17, and assume that the eigenvalues $\{\mu_k\}_{k=1}^{K}$ are sorted. As λ increases, $\gamma_{k,n}$ decreases and can eventually become negligible.

Specifically, there exists an index n_f such that for $k > n_f$, we have $\frac{\mu_k}{\mu_k + \lambda/L} \ll 1$. In the extreme case, $n_f = K$. Accordingly, n_f controls the RKHS norm constraint of the model in Eq. 7. Faster eigenvalue decay implies that this transition occurs earlier, so the effective rank can remain small even when K is large. This is the sense in which the practical hyperparameter n_f approximates the number of non-negligible kernel components in the learned predictor.

C Experimental Setup

This section details the datasets, hyperparameter search space, tuning setup, hardware, and training time estimates.

Table 4: Summary statistics of considered TSF datasets.

Dataset	Weather	Electricity	Traffic	ETTh1	ETTh2	ETTm1	ETTm2
Channels	21	321	862	7	7	7	7
Timesteps	52696	26304	17544	17420	17420	69680	69680
Total size	1106616	8443584	15122928	121940	121940	487760	487760

We evaluate the models on widely used real-world time series datasets, namely Electricity Transformer Temperature (ETTh1, ETTh2, ETTm1, ETTm2), Traffic, Electricity, and Weather [Zhou et al., 2021], summarized in Table 4. These datasets cover diverse application domains, are openly available, support reproducible experimentation, and enable direct comparison with prior work that uses the same benchmarks. We train separate models for prediction horizons in $\{96, 192, 336, 720\}$ samples and optimize the architectural hyperparameters using the Tree-structured Parzen Estimator algorithm [Bergstra et al., 2011].

The hyperparameter search space is reported in Tables 5a and 5b. For both filterbank and filterbank-deep, we tune the learning rate, the number of filters n_f, and the filter length (convolution kernel size) K. For filterbank-deep, we additionally tune the keep fraction k_f and reduce the search range of n_f to keep the computations feasible. The keep fraction k_f allows ResBlock modules to operate on a longer input, which facilitates learning, and gradually relaxes the rank constraint as k_f increases.

Our experimental pipeline is as follows. The reported benchmark experiments span seven datasets, four prediction horizons, and separate proposed-model configurations for each dataset–horizon pair. For each proposed architecture, dataset, and prediction horizon, we run about 20 trials with train–validation splits and tune hyperparameters using the Tree-structured Parzen Estimator algorithm. Additional trials did not yield noticeable gains. To avoid unnecessary computation, we prune trials early when their performance falls below the median of completed trials. After all trials finish, we retain the best hyperparameters for each configuration. The selected numbers of filters and keep fractions are listed in Table 12, and all hyperparameters and the corresponding results are provided in the `compiled_results/` directory of the public code repository.

In Table 2, the benchmark results for DLinear, PatchTST, and SparseTSF are taken from the conference version of SparseTSF [Lin et al., 2024, Appendix D]. SparseTSF uses a look-back window of $L = 720$, whereas DLinear and PatchTST use their official look-back windows. On ETTh1, ETTh2, Electricity, and Traffic, the DLinear and PatchTST results come from the reproduction by Xu et al. [2023], which re-evaluated these models after fixing a longstanding bug that affected several works starting from Informer [Zhou et al., 2021]. The results for TimeKAN, TimeMixer, iTransformer, and TimePro are instead sourced from the TimeKAN paper Huang et al. [2025] and all use $L = 96$. This preserves broad coverage of strong published baselines rather than enforce a single unified re-evaluation. Appendix D.2 repeats the comparison with a unified look-back window of $L = 720$. We also employ reversible instance normalization [Kim et al., 2021], as in SparseTSF and earlier work, to normalize inputs and then denormalize outputs to obtain the final predictions.

We train all models on MI250x GPUs. As representative training-time measurements, for the ETTm1 dataset with $L = H = 720$, one training epoch takes about 14 seconds for filterbank and about 20 seconds for filterbank-deep. We train for up to 100 epochs with an exponentially decaying learning rate ($\beta = 0.8$) and the Adam optimizer [Kingma and Ba, 2014]. Training stops early if the validation loss does not improve for 5 consecutive epochs or if the learning rate falls below 10^{-6}. All models are implemented in PyTorch [Paszke et al., 2019].

Table 5: (Hyper)parameters employed for (a) FB-LR and (b) FB-deep.

(a)

Parameter	Value(s)
Learning rate	[0.001, 0.03]
Number of filters n_f	{2,. . . ,80}
Convolution kernel size	{25,. . . ,L}
Reversible instance normalization	True
Keep fraction k_f	1/L

(b)

Parameter	Value(s)
Learning rate	[0.001, 0.03]
Number of filters n_f	{1,. . . ,10}
Filter length	{25,. . . ,L}
Reversible instance normalization	True
Keep fraction k_f	[1/L,1]
Hidden layer size	512
Dropout rate	0
Layer normalization	False

Table 6: MSE (the lower the better) for long-term TSF over seven datasets. Methods are grouped by model family: Low-rank, Low-complexity, KAN, Mixer, Transformers, and State-space. Best results in **bold**, second-best on a shaded background.

		Ours		Low-complexity models				KAN	Mixer	Transformer-based models			State-space models	
Dataset	Horizon	FB-deep (Ours)	FB-LR (Ours)	SparseTSF (ICML'24)	MixLinear (ICLR'26)	FITS (ICLR'24)	DLinear (AAAI'23)	TimeKAN (ICLR'25)	TimeMixer (ICLR'24)	iTransformer (ICLR'24)	PatchTST (ICLR'23)	Fedformer (ICML'22)	TimePro (ICML'25)	S-Mamba (Neurocomp'25)
Weather	96	0.148	0.167	0.172	0.170	**0.145**	0.176	0.162	0.163	0.174	0.149	0.217	0.166	0.165
	192	0.192	0.209	0.215	0.212	**0.188**	0.218	0.207	0.211	0.221	0.194	0.276	0.216	0.214
	336	0.241	0.254	0.260	0.257	**0.236**	0.262	0.263	0.263	0.278	0.245	0.339	0.273	0.274
	720	0.319	0.318	0.318	0.321	**0.308**	0.323	0.338	0.344	0.358	0.314	0.403	0.351	0.350
Electricity	96	**0.128**	0.134	0.138	0.138	0.145	0.140	0.174	0.153	0.148	0.129	0.188	0.139	–
	192	**0.148**	0.153	0.151	0.154	0.159	0.153	0.182	0.166	0.162	0.149	0.197	0.156	–
	336	**0.159**	0.169	0.166	0.170	0.175	0.169	0.197	0.185	0.178	0.166	0.212	0.172	–
	720	**0.196**	0.209	0.205	0.209	0.212	0.204	0.236	0.224	0.225	0.210	0.244	0.209	–
Traffic	96	**0.360**	0.387	0.389	0.389	0.398	0.413	–	0.462	0.395	0.366	0.573	–	–
	192	**0.381**	0.417	0.398	0.403	0.409	0.423	–	0.473	0.417	0.388	0.611	–	–
	336	**0.395**	0.425	0.411	0.416	0.421	0.437	–	0.498	0.433	0.398	0.621	–	–
	720	**0.430**	0.463	0.448	0.452	0.457	0.466	–	0.506	0.467	0.457	0.630	–	–
ETTm1	96	**0.292**	0.315	0.314	0.308	0.305	0.299	0.322	0.317	0.334	0.293	0.379	0.326	0.333
	192	0.338	0.338	0.343	0.337	0.339	0.335	0.357	0.367	0.377	**0.333**	0.426	0.367	0.376
	336	0.373	0.371	0.369	**0.365**	0.367	0.369	0.382	0.391	0.426	0.369	0.445	0.402	0.408
	720	0.440	**0.414**	0.418	0.415	0.418	0.425	0.445	0.454	0.491	0.416	0.543	0.469	0.475
ETTm2	96	0.172	**0.161**	0.165	0.165	0.164	0.167	0.174	0.175	0.180	0.166	0.203	0.178	0.179
	192	0.231	**0.216**	0.218	0.219	0.217	0.221	0.239	0.240	0.250	0.223	0.269	0.242	0.250
	336	0.286	0.270	0.272	**0.269**	**0.269**	0.274	0.301	0.303	0.311	0.274	0.325	0.303	0.312
	720	0.367	0.360	0.350	0.351	**0.347**	0.368	0.395	0.392	0.412	0.362	0.421	0.400	0.411
ETTh1	96	0.400	0.371	0.362	**0.351**	0.382	0.384	0.367	0.385	0.386	0.385	0.375	0.375	0.386
	192	0.454	0.405	0.403	**0.395**	0.417	0.443	0.414	0.443	0.441	0.413	0.427	0.427	0.443
	336	0.441	0.441	0.434	**0.411**	0.436	0.446	0.445	0.512	0.487	0.440	0.459	0.472	0.489
	720	0.478	0.473	0.426	**0.423**	0.433	0.504	0.444	0.497	0.503	0.456	0.484	0.476	0.502
ETTh2	96	0.291	**0.271**	0.294	0.283	0.272	0.282	0.290	0.289	0.297	0.274	0.340	0.293	0.296
	192	0.376	0.346	0.339	0.336	**0.333**	0.350	0.375	0.378	0.380	0.338	0.433	0.367	0.376
	336	0.382	0.391	0.359	**0.355**	**0.355**	0.414	0.423	0.432	0.428	0.367	0.508	0.419	0.424
	720	0.451	0.422	0.383	0.380	**0.378**	0.588	0.443	0.464	0.427	0.391	0.480	0.427	0.426

D Additional Results

This section collects supplementary empirical results that complement the main-text evaluation. It broadens the comparison with the state of the art, repeats it with the same look-back window for all models, reports training-resource measurements, and further examines whether the reported gains persist under pruning-based baselines and across multiple random seeds.

D.1 Broader Comparison with the State of the Art

Table 6 reports a broader comparison with several established methods from different architecture families for long-term TSF. In addition to the models discussed in the main text, this table includes FITS [Xu et al., 2023], a lightweight frequency-domain forecaster; FedFormer [Zhou et al., 2022], a frequency-enhanced transformer; and S-Mamba [Wang et al., 2025], a state-space model based on selective state-space blocks. The additional results for FITS and FedFormer are also sourced from the SparseTSF paper Lin et al. [2024], with FITS evaluated at $L = 720$ and FedFormer at its official look-back window; those for MixLinear [Ma et al., 2026] and S-Mamba are instead taken from their original publications [Ma et al., 2026, Wang et al., 2025]. Methods are grouped by architecture family to make the comparison easier to interpret across low-rank, low-complexity, KAN, mixer, transformer-based, and state-space approaches.

Table 7: Test MSE for long-term TSF with the same look-back window $L = 720$ for all models. TimePro is omitted, and – marks unavailable results. Best results in **bold**, second-best are shaded.

Dataset	Horizon	FB-deep (Ours)	FB-LR (Ours)	SparseTSF (ICML'24)	DLinear (AAAI'23)	TimeKAN (ICLR'25)	TimeMixer (ICLR'24)	iTransformer (ICLR'24)	PatchTST (ICLR'23)
Weather	96	0.148	0.167	0.169	0.169	**0.143**	0.151	0.180	0.149
	192	0.192	0.209	0.214	0.214	**0.185**	0.195	0.226	0.191
	336	0.241	0.254	0.257	0.256	**0.237**	0.258	0.286	0.245
	720	0.319	0.318	0.322	0.315	**0.310**	0.325	0.353	0.315
Electricity	96	**0.128**	0.134	0.139	0.136	0.133	0.136	0.133	0.130
	192	0.148	0.153	0.151	0.150	0.148	0.154	0.152	**0.147**
	336	**0.159**	0.169	0.166	0.166	0.165	0.175	0.169	0.162
	720	0.196	0.209	0.207	0.200	0.200	0.205	**0.194**	0.201
Traffic	96	**0.360**	0.387	0.389	0.396	–	0.387	0.362	0.370
	192	0.381	0.417	0.399	0.406	–	0.405	**0.375**	0.382
	336	0.395	0.425	0.411	0.419	–	0.421	**0.390**	0.395
	720	0.430	0.463	0.449	0.459	–	0.455	**0.423**	0.434
ETTm1	96	**0.292**	0.315	0.314	0.307	0.300	0.333	0.314	0.294
	192	0.338	0.338	0.343	0.347	**0.333**	0.341	0.349	0.334
	336	0.373	0.371	0.369	0.367	0.378	0.407	0.381	**0.362**
	720	0.440	0.414	0.418	0.415	**0.413**	0.468	0.448	0.416
ETTm2	96	0.172	**0.161**	0.165	0.163	0.169	0.186	0.181	0.162
	192	0.231	**0.216**	0.218	0.223	0.222	0.225	0.238	**0.216**
	336	0.286	0.270	0.272	0.291	0.275	0.304	0.295	**0.268**
	720	0.367	0.360	**0.350**	0.407	0.385	0.381	0.382	0.351
ETTh1	96	0.400	0.371	**0.362**	0.381	0.371	0.400	0.396	0.375
	192	0.454	0.405	**0.404**	0.419	0.406	0.433	0.430	0.418
	336	0.441	0.441	**0.435**	0.454	0.449	0.460	0.480	0.462
	720	0.478	0.473	**0.426**	0.499	0.507	0.505	0.700	0.495
ETTh2	96	0.291	**0.271**	0.294	0.302	0.274	0.294	0.311	0.290
	192	0.376	0.346	**0.340**	0.411	0.341	0.343	0.392	0.382
	336	0.382	0.391	**0.360**	0.542	0.415	0.384	0.415	0.421
	720	0.451	0.422	**0.383**	0.888	0.420	0.472	0.425	0.404

D.2 Comparison at a Look-Back Window of $L = 720$

As described in Appendix C, the baselines in Table 2 use different look-back windows. Table 7 repeats this comparison with the baselines evaluated at $L = 720$, the look-back window used by our models. For SparseTSF, DLinear, iTransformer, and PatchTST, we use the most recent published results with $L = 720$, namely those of the journal version of SparseTSF [Lin et al., 2026], which evaluates all four models with $L = 720$. The journal version covers Weather, Electricity, Traffic, ETTh1, and ETTh2, but does not report results on ETTm1 and ETTm2. For these two datasets, we therefore take the SparseTSF results from the conference version [Lin et al., 2024], which uses the same look-back window $L = 720$, and rerun DLinear, iTransformer, and PatchTST ourselves with $L = 720$. TimeKAN and TimeMixer are rerun with $L = 720$ on all datasets. Apart from the look-back window, we use the official or reported hyperparameters of each baseline rather than a new hyperparameter search for every method, dataset, and horizon. TimePro could not be executed on our computing infrastructure because of hardware and software incompatibilities and is therefore omitted. With the same look-back window, FB-deep remains the best or second-best model at all horizons on Electricity and Traffic, whereas FB-LR and SparseTSF are each among the two best models in 9 of the 16 settings on the ETT datasets, with SparseTSF achieving the lowest error more often (8 settings versus 3 for FB-LR). TimeKAN achieves the lowest error at all horizons on Weather, and SparseTSF on ETTh1. On ETTm1 and ETTm2, PatchTST is among the two best models at seven of the eight horizons. Compared with DLinear at the same look-back window, FB-LR achieves a lower MSE in 19 of the 28 dataset–horizon pairs.

D.3 Training Cost

Table 8 reports direct training-step profiling measurements on an NVIDIA RTX 3080 Ti for ETTm1 with look-back window $L = 720$ and batch size 32. Both proposed models retain a modest train-

Table 8: Training-step cost of the proposed models on ETTm1 with look-back window $L = 720$ and batch size 32. Lower values are better.

Model	Per-step latency (ms)	Peak CUDA memory (MiB)	Peak system RAM (MiB)
FB-LR	8.306 ± 0.454	50.754	9889.160
FB-deep	7.736 ± 0.782	80.307	17826.305

Table 9: Test MSE (the lower the better) for DLinear magnitude-pruned to match the parameter count of FB-LR, reported across forecast horizons. Results are averaged over five random seeds.

Dataset	Pruned DLinear			
	96	192	336	720
Weather	0.170	0.212	0.257	0.315
ETTm1	0.308	0.338	0.371	0.422
ETTm2	0.164	0.222	0.284	0.381
ETTh1	0.376	0.415	0.446	0.500
ETTh2	0.324	0.381	0.479	0.726

ing footprint in this setting: the average per-step latency stays below 10 ms, and the peak CUDA allocation remains below 100 MiB. FB-deep uses more memory than FB-LR, as expected from its additional depth, but both variants remain far from the memory regimes typically associated with heavier transformer and state-space baselines. The system-RAM numbers further indicate that the training overhead stays manageable even on a standard workstation.

D.4 Comparison with Pruned DLinear

To disentangle the impact of parameter count from that of the proposed structure, we compare FB-LR with a magnitude-pruned DLinear baseline. For each considered dataset and prediction horizon, we prune DLinear to match the parameter count of the corresponding FB-LR model. This removes 84.7% of the original weights on average. The pruned model is then fine-tuned for 10 epochs at fixed sparsity on the training set. We purposely consider this pruning-based comparison as a tighter parameter-budget control than standard regularization baselines such as ridge penalties or dropout, because pruning directly enforces a parameter count comparable to FB-LR rather than merely encouraging smoother full-rank solutions.

Comparing Table 9 with the corresponding FB-LR results in Table 6, FB-LR achieves the best accuracy in 16 out of 20 cases, whereas Pruned DLinear performs better in only four cases, all by a small margin. In the remaining cases, FB-LR outperforms the pruned baseline, often substantially. These results confirm that the efficiency advantage of FB-LR is not merely a consequence of having fewer parameters; rather, structured decomposition extracts more useful features per parameter than standard compression.

D.5 Variation Across Random Seeds

To assess the variation induced by random initialization and training stochasticity, we re-run the ETT benchmarks reported in Tables 1 and 2 using five random seeds. Table 10 reports the mean test MSE and the corresponding standard deviation for both FB-LR and FB-deep.

The standard deviations are small relative to the mean MSE values in most settings, indicating that the main ETT conclusions are stable across random seeds. The higher variability of FB-deep on ETTh1 at the 720-sample horizon reflects the greater sensitivity of the deeper variant on some smaller datasets, but it does not change the overall pattern: FB-LR is generally preferred on the ETT benchmarks, whereas FB-deep remains competitive when additional model capacity is beneficial.

Table 10: Mean test MSE $\pm$ standard deviation over five random seeds on the ETT benchmarks. Lower values are better.

Dataset	Horizon	FB-LR	FB-deep
ETTh1	96	0.3700 ± 0.0032	0.4044 ± 0.0073
	192	0.4069 ± 0.0066	0.4455 ± 0.0168
	336	0.4382 ± 0.0118	0.4513 ± 0.0208
	720	0.4777 ± 0.0084	0.5248 ± 0.0594
ETTh2	96	0.2764 ± 0.0035	0.2909 ± 0.0059
	192	0.3391 ± 0.0030	0.3756 ± 0.0055
	336	0.3770 ± 0.0105	0.3943 ± 0.0092
	720	0.4254 ± 0.0147	0.4383 ± 0.0240
ETTm1	96	0.3234 ± 0.0202	0.2927 ± 0.0037
	192	0.3411 ± 0.0043	0.3459 ± 0.0069
	336	0.3701 ± 0.0029	0.3774 ± 0.0029
	720	0.4174 ± 0.0055	0.4336 ± 0.0041
ETTm2	96	0.1618 ± 0.0005	0.1781 ± 0.0038
	192	0.2180 ± 0.0024	0.2348 ± 0.0033
	336	0.2704 ± 0.0009	0.2898 ± 0.0037
	720	0.3568 ± 0.0028	0.3594 ± 0.0019

Table 11: Test MSE on the financial datasets with look-back window $L = 720$. Best results in **bold**, second-best are shaded.

Dataset	Horizon	FB-deep (Ours)	FB-LR (Ours)	SparseTSF (ICML'24)	DLinear (AAAI'23)	TimeKAN (ICLR'25)	TimeMixer (ICLR'24)	iTransformer (ICLR'24)	PatchTST (ICLR'23)
NASDAQ	96	1.127	**0.643**	1.213	0.784	0.735	1.609	1.947	1.190
	192	1.773	**1.611**	2.252	2.168	3.383	2.482	3.622	1.689
	336	5.456	**3.470**	4.641	6.412	4.828	4.199	6.559	4.352
	720	17.590	**7.374**	10.102	21.868	8.968	15.921	9.774	9.729
SP500	96	1.556	**0.996**	1.670	2.037	1.600	2.002	2.521	1.695
	192	3.377	**2.374**	3.468	2.586	4.897	5.004	5.691	4.914
	336	6.841	**4.424**	6.568	22.806	9.517	7.235	10.496	9.710
	720	8.598	**8.197**	11.998	70.967	11.540	11.265	13.860	10.936

D.6 Financial Datasets

To assess the proposed models beyond the standard long-term TSF benchmark, we evaluate them on two daily financial datasets. NASDAQ[1] comprises 3,914 trading days and 12 market, economic, and commodity channels, namely open, high, low, and close prices, trading volume, interest rate, exchange rate, VIX, TED spread, effective federal funds rate, gold, and oil. SP500 comprises 8,077 trading days and five daily split- and dividend-adjusted price and volume channels of the SPDR S&P 500 ETF Trust (SPY), an exchange-traded fund that tracks the S&P 500 index, obtained from Yahoo Finance.[2] We split each dataset chronologically into $60\%/20\%/20\%$ training, validation, and test partitions, as for the ETT datasets, fit the standardization statistics on the training partition, and forecast and evaluate all channels with look-back window $L = 720$. We evaluate SparseTSF, DLinear, TimeKAN, TimeMixer, iTransformer, and PatchTST using their released implementations and select their configurations by validation MSE. FB-LR and FB-deep follow the training protocol in Appendix C.

Table 11 reports the results. FB-LR achieves the lowest MSE in all eight dataset–horizon settings. FB-deep does not improve on FB-LR on these datasets, which suggests that the lower-rank architecture is better suited to them, whereas the relative performance of the nonlinear baselines depends on the dataset and the horizon. The errors are larger than on the standard benchmark and grow quickly with the horizon, reflecting the chronological distribution shift between the training and test periods of these nonstationary series.

[1] https://www.kaggle.com/datasets/sai14karthik/nasdq-dataset

[2] https://finance.yahoo.com/quote/SPY/history/

E Rank of the Trained Filter Bank Models

This section reports the validation-selected number of filters n_f and keep fraction k_f for the FB-LR and FB-deep results in Table 2, relates them to the optimal RRR ranks in Fig. 1b, and characterizes the nonlinear FB-deep models through the effective rank of their local linearizations. Table 12 summarizes these quantities.

E.1 Rank of FB-LR

The FB-LR prediction in Eq. (9) can be written as $\hat{\mathbf{y}}_i = \mathbf{\Gamma}^\top \mathbf{F}\, \mathbf{x}_i$, where the rows of $\mathbf{F} \in \mathbb{R}^{n_f \times L}$ are the (zero-padded) convolution filters. Hence, the prediction matrix $\mathbf{W} = \mathbf{\Gamma}^\top \mathbf{F}$ satisfies $\operatorname{rank}(\mathbf{W}) \leq n_f$, and n_f directly upper-bounds the rank of the learned predictor (Corollary 4). The selected values satisfy $n_f \leq 80 < \min(H, L)$ in all cases, so that the rank constraint is active at every horizon; for $H \geq 192$, n_f is at most about 40% of the full rank $\min(H, L)$.

The selected n_f exceeds the optimal RRR rank r^* in Fig. 1b in all cases. This is expected, for two reasons. First, n_f is an upper bound on the rank rather than an estimate of r^*: both the RRR validation loss (Fig. 1a) and the FB-LR test loss (Fig. 3b) are nearly flat beyond their optimum, so the validation-based search, which tunes n_f jointly with the convolution kernel size and the learning rate, can select larger values of n_f at little cost in validation loss. Second, the two problems are not identical: FB-LR restricts the filter support to the convolution kernel size, uses reversible instance normalization, and is trained by gradient descent, whereas r^* is obtained from the closed-form RRR solution. We therefore interpret r^* as a reference for the intrinsic rank of each task rather than as a target value for n_f. On Electricity and Traffic, the search selects values in the upper part of the search range ($59 \leq n_f \leq 80$), consistent with the observation in Section 5.1 that these larger datasets benefit from additional capacity.

E.2 Effective Rank of FB-deep

Remark 5 bounds the rank of FB-deep after removing the ReLU activation. Since the actual FB-deep model is nonlinear, it has no single global prediction matrix $\mathbf{W}$, and its algebraic rank is not well defined. We therefore characterize FB-deep locally, through the Jacobian $\mathbf{J}(\mathbf{x}) := \partial \hat{\mathbf{y}} / \partial \mathbf{x} \in \mathbb{R}^{H \times L}$ of the trained model at an input $\mathbf{x}$.

Rank of the local linearizations. Let $\mathbf{C}_k \in \mathbb{R}^{s \times L}$ denote the linear map that convolves the input with filter $\mathbf{f}_k$ and retains the last s convolution samples, so that $\mathbf{z}_{i,k} = \mathbf{C}_k \mathbf{x}_i$. At a fixed input, the ReLU gates in Eq. (10) are fixed, and we collect them in the diagonal matrix $\mathbf{D}_k(\mathbf{x}) \in \{0,1\}^{d_k \times d_k}$, whose j-th diagonal entry is one if $(\mathbf{A}_{k,1}\mathbf{C}_k\mathbf{x} + \mathbf{b}_{k,1})_j > 0$ and zero otherwise. Differentiating Eq. (11) then gives, wherever the ReLU is differentiable,

$$\mathbf{J}(\mathbf{x}) = \sum_{k=1}^{n_f} \big(\mathbf{A}_{k,2}\mathbf{D}_k(\mathbf{x})\mathbf{A}_{k,1} + \mathbf{A}_{k,\text{skip}}\big)\mathbf{C}_k, \qquad \operatorname{rank}\big(\mathbf{J}(\mathbf{x})\big) \leq \min(n_f s, H, L) =: r_{\text{ub}}, \quad (18)$$

since each term of the sum factors through $\mathbf{C}_k$ and thus has rank at most s. Hence, the bound of Remark 5 holds for every local linearization of the nonlinear model, not only for the model without ReLU. The ReLU activation does not remove this rank bottleneck; it makes the active linear map input dependent, so that different inputs activate different local linear maps within the same architecture-level bound. The convolution kernel size determines which input samples each branch can access, but does not affect r_{ub}, since each branch retains only an s-dimensional vector before its residual block. With reversible instance normalization, the input normalization composes with $\mathbf{J}$ and cannot increase its rank, whereas the instance-dependent rescaling and shift of the output add at most two rank-one terms.

Effective rank. The algebraic rank counts all nonzero singular values, regardless of their magnitude, and therefore does not reflect how many directions the local map effectively uses. We instead measure the entropic effective rank [Roy and Vetterli, 2007]. Let $\sigma_1 \geq \cdots \geq \sigma_q \geq 0$, with $q = \min(H, L)$, be the singular values of $\mathbf{J}(\mathbf{x})$, and let $p_j = \sigma_j / \sum_{l=1}^{q} \sigma_l$. The effective rank is

$$r_{\text{eff}}\big(\mathbf{J}(\mathbf{x})\big) = \exp\Big(-\sum_{j=1}^{q} p_j \log p_j\Big), \qquad 1 \leq r_{\text{eff}}\big(\mathbf{J}(\mathbf{x})\big) \leq \operatorname{rank}\big(\mathbf{J}(\mathbf{x})\big) \leq r_{\text{ub}},$$

Table 12: Selected hyperparameters and rank measures of the models in Table 2 ($L = 720$). For FB-LR, n_f upper-bounds the rank of the prediction matrix, and r^* is the optimal RRR rank from Fig. 1b (–: not evaluated). For FB-deep, $s = \lfloor k_f L \rfloor$ is the number of retained convolution samples per branch, $r_{\mathrm{ub}} = \min(n_f s, H, L)$ is the rank bound of Eq. (18), and $\bar{r}_{\mathrm{eff}}$ is the mean effective rank of the local Jacobians (Appendix E.2).

		FB-LR		**FB-deep**				
Dataset	**Horizon**	n_f	r^*	n_f	k_f	s	r_{ub}	$\bar{r}_{\mathrm{eff}}$
Weather	96	19	6	4	0.1080	77	96	35.1
	192	29	6	1	0.3268	235	192	45.7
	336	32	7	1	0.7892	568	336	175.0
	720	79	7	3	0.0328	23	69	16.5
Electricity	96	79	–	5	0.3005	216	96	76.6
	192	64	–	5	0.2608	187	192	142.7
	336	59	–	3	0.2009	144	336	179.7
	720	63	–	3	0.4491	323	720	251.5
Traffic	96	80	–	10	0.9764	703	96	75.0
	192	78	–	9	0.7310	526	192	140.3
	336	77	–	5	0.7301	525	336	177.7
	720	60	–	8	0.4284	308	720	310.9
ETTm1	96	64	18	6	0.0534	38	96	48.8
	192	74	15	7	0.0096	6	42	23.1
	336	24	14	6	0.0107	7	42	31.6
	720	24	20	4	0.0148	10	40	31.8
ETTm2	96	31	23	3	0.3178	228	96	72.3
	192	72	24	4	0.6294	453	192	170.3
	336	51	6	1	0.2236	160	160	133.4
	720	59	18	1	0.5902	424	424	332.2
ETTh1	96	17	12	4	0.3912	281	96	87.3
	192	27	14	2	0.1506	108	192	131.8
	336	55	7	6	0.2653	191	336	221.2
	720	36	9	10	0.9744	701	720	469.6
ETTh2	96	62	22	8	0.3023	217	96	70.7
	192	68	6	10	0.3564	256	192	160.3
	336	47	17	6	0.3656	263	336	252.0
	720	72	3	1	0.4282	308	308	182.1

with the convention $0 \log 0 = 0$. The inequality $r_{\mathrm{eff}} \leq \mathrm{rank}$ holds with equality if and only if all nonzero singular values are equal, and r_{eff} decreases as the singular values become more concentrated.

Measurement protocol. For each of the 28 trained FB-deep configurations in Table 2, we compute the same-channel Jacobian $\mathbf{J}^{(c)}(\mathbf{x}) \in \mathbb{R}^{H \times L}$ of the complete trained model, including reversible instance normalization, for 100 evenly spaced test windows and three representative channels c (8,400 Jacobians in total). Table 12 reports their mean effective rank $\bar{r}_{\mathrm{eff}}$.

Results. All 8,400 measured effective ranks lie below their corresponding r_{ub}, with a maximum ratio $r_{\mathrm{eff}}/r_{\mathrm{ub}}$ of 0.914. Across the 28 configurations, $\bar{r}_{\mathrm{eff}}$ closely follows the capacity ordering predicted by r_{ub}, with a Pearson correlation of 0.898 and a Spearman correlation of 0.940. Configurations with a larger rank capacity thus generally use a higher effective rank, which empirically connects the linearized capacity analysis to the trained nonlinear model. Here, r_{ub} depends jointly on n_f and k_f, as well as on the dimensional ceiling $\min(H, L)$: increasing k_f relaxes the capacity constraint only until $r_{\mathrm{ub}} = \min(H, L)$, which all Electricity and Traffic configurations reach.

The trained models typically use only part of their rank capacity, with $\bar{r}_{\mathrm{eff}}/r_{\mathrm{ub}}$ ranging from 0.24 to 0.91. The effective rank is lowest on Weather, where $\bar{r}_{\mathrm{eff}}$ remains below 50 at three of the four horizons, consistent with the small optimal RRR ranks of this dataset in Fig. 1b. On ETTm1, the selected keep fractions yield $r_{\mathrm{ub}} \in \{40, 42\}$ for $H \geq 192$, which is about two to three times the optimal RRR ranks $r^* \in \{14, 15, 20\}$. Since the effective rank is a local measure evaluated at test inputs, these results characterize how the trained models use their capacity, rather than providing a global rank or generalization guarantee for the nonlinear model.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's claims and scope, and the contribution list points to the supporting theoretical analysis, architecture design, and empirical evaluation in Sections 3, 4, and 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations and scope throughout the analysis and evaluation: Section 3.3 notes the practical limitations of RRR and kernel methods, Section 5 discusses dataset-dependent behavior and computational scaling, and Section 6 highlights limitations and future work on linking dataset properties to optimal filter-bank capacity.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 3 states the assumptions for the reduced-rank regression and kernel-regression results. The RRR result is a standard proposition with the relevant references, while Proposition 2 includes a proof sketch in the main text and a full proof in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 5 provides a link to the publicly available source code. Appendix C reports the datasets, prediction horizons, hyperparameter search spaces, tuning procedure, optimizer, early stopping criterion, hardware, and training-time estimates needed to reproduce the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available at `https://github.com/ashutoshvai1/FilterBanks`, together with the experimental pipeline, hyperparameter settings, and compiled results needed to reproduce the main results reported in the paper. The experiments use openly available long-term time series forecasting benchmark datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix C reports the datasets, prediction horizons, hyperparameter search spaces, tuning procedure, optimizer, learning-rate schedule, early-stopping criterion, hardware, and implementation framework used in the experiments. The public code repository further provides the retained hyperparameters and compiled results for each configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Figure 3 reports mean values over five random seeds with standard deviations shown as shaded regions where non-negligible. Table 3 reports inference time as median $\pm$ standard deviation over 10,000 forward passes, and Appendix Table 10 reports mean test MSE $\pm$ standard deviation over five random seeds for the ETT benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C reports the training hardware, hyperparameter-search budget, stopping criteria, and representative training-time estimates. Appendix D.3 reports measured training-step latency and memory usage for the proposed models.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is methodological work on time-series forecasting and uses public benchmark datasets without human-subject data, personally identifiable information, or deployment claims. We have reviewed the NeurIPS Code of Ethics and are not aware of any aspects of the work that conflict with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive application domains such as energy systems, healthcare, finance, and resource management, and reports efficiency benefits that may reduce deployment cost. It does not discuss negative societal impacts because the work is a generic methodological contribution to efficient time-series forecasting and is not tied to a particular application or deployment pathway. This matches the guideline that such generic algorithmic improvements need not enumerate indirect negative applications without a direct path to harm.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models or scraped datasets with plausible misuse concerns; it studies efficient architectures for time-series forecasting using standard benchmark datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper credits the original sources for the benchmark datasets and baseline implementations used in the experiments, and uses them in accordance with their stated terms and licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper documents the introduced model, experimental protocol, and implementation details needed to reproduce the reported results, and these materials are provided alongside the submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.