UNICORN: UNIVERSAL CORRELATION MODELING FOR HIGH-DIMENSIONAL TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern time series architectures face a fundamental trade-off: channel-independent models scale well across datasets but ignore inter-channel dependencies, while many channel-dependent models learn correlations within individual datasets and face computational or transfer challenges when channel sets change. To bridge this gap, we introduce **Unicorn** (Universal Correlation Network), a framework for scalable, multi-dataset pretraining on high-dimensional time series. At the core of Unicorn is a *continuous prototype bank* that decouples correlation modeling from specific channel identities. By routing heterogeneous channels through a shared latent space, Unicorn learns *identity-agnostic, reusable interaction patterns* with linear interaction cost in the number of channels, enabling joint pretraining across different channel sets. Experiments on financial and public forecasting benchmarks show competitive forecasting accuracy and strong data efficiency during finetuning, offering a scalable path toward multivariate time series foundation models.

1 INTRODUCTION

High-dimensional time series forecasting is at the heart of many real-world applications, ranging from financial market analysis to large-scale energy system management and climate modeling. In these domains, systems must jointly reason over hundreds or even thousands of correlated variables (*i.e.*, input channels). Despite recent advances in time series foundation models, scaling modern architectures to such high-dimensional settings remains challenging. This challenge arises from the fundamental conflicts between **domain scalability** and **correlation modeling**. Channel-independent (CI) models (Shi et al., 2026; Das et al., 2024) process channels (assets in our Kronos setting) independently with shared parameters, which scales well across heterogeneous datasets but discards inter-channel interactions. Channel-dependent (CD) models (Liu et al., 2024a; 2025a) explicitly capture interactions among variables, but are commonly trained separately for each dataset. Some designs use channel-indexed parameters; others, such as iTransformer, have parameters independent of channel count but still compute dense $C \times C$ attention. Parameter sharing alone does not ensure that correlations learned on one dataset transfer effectively to another.

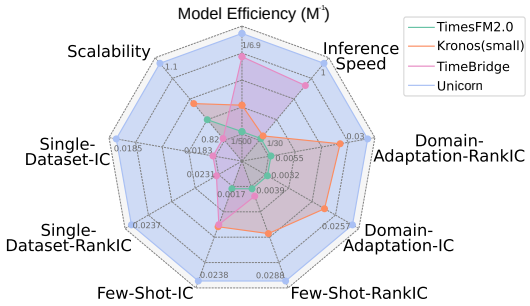


Figure 1: **Performance on a 587-stock A-share dataset.** The radar chart benchmarks model performance across four key dimensions: efficiency, generalization, scalability, and few-shot adaptability. Scalability is evaluated by the performance delta between full-channel and 25% channel configurations during finetuning and inference. Few-shot performance is measured by finetuning on a restricted 25% subset of the training data.

Despite ignoring inter-channel dependencies, empirical studies show that CI models can outperform CD counterparts in multivariate forecasting tasks (Nie et al., 2023). This observation suggests a key challenge for CD approaches: learning reusable dependencies rather than fitting correlations specific to a single dataset. This issue is particularly pronounced when many channels are observed over a short history, leaving limited temporal evidence for learning their relationships. In some datasets,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the channel count C even exceeds the total sequence length T, as in Wiki-People and the A-share pretraining data (Appendix A). Here, scarcity concerns the available training history and windows, not the length I of a single input window. Sharing temporal and relational structure across datasets is therefore particularly useful in this many-channel, short-history setting.

These observations motivate a central hypothesis: the limitation of CD models lies **NOT** in modeling correlations per se, but in the **scale and manner** in which they are learned. Building on this insight, we propose *Unicorn* (Universal Correlation Network), a unified framework that decouples correlation modeling from explicit channel identities to enable scalable pretraining across heterogeneous datasets. Our design starts with a basic assumption: while physical channel identities vary across domains, some interaction patterns (*e.g.*, sector-level co-movements across stock markets) can be shared and reused. Unicorn organizes important predictive dependencies in high-dimensional time series through a learnable **Continuous Prototype Bank**, whose anchors represent shared interaction patterns across channels. To align heterogeneous channels with these anchors, we introduce a **Spectral Global Guidance** module that uses learned periodic features to provide shared context. Rather than constructing dense *channel-to-channel* attention, Unicorn routes dependencies through a *channel-to-prototype* interaction mechanism. By treating prototypes as an intermediate "interaction bottleneck", the model jointly learns a reusable basis and input-dependent relations across datasets with different channel counts, combining scalable pretraining with explicit inter-channel modeling.

Experiments demonstrate that Unicorn outperforms TimesFM (Das et al., 2024) and Kronos (Shi et al., 2026) on the evaluated financial finetuning tasks and is competitive with specialized CD architectures on public forecasting benchmarks. Channel-subset and limited-data experiments further support its scalability and data efficiency, while controlled analyses examine the contribution and reuse of the learned prototype space.

2 Problem Formalization

We consider multi-dataset pretraining for high-dimensional time series forecasting. Given an input sequence $\mathbf{X} \in \mathbb{R}^{C \times I}$ spanning a lookback window of length I across C variables (channels), our objective is to predict the future values $\mathbf{Y} \in \mathbb{R}^{C \times O}$ for a horizon of length O. This high-dimensional setting poses two fundamental challenges: Dense channel attention incurs $O(C^2)$ interaction cost. Separately, short training histories can make reliable correlation learning difficult when C is large. Let T denote the total number of observed timestamps in a dataset; $C > T$ describes a particularly wide dataset, whereas $C > I$ alone does not establish a shortage of training samples. In multi-dataset pretraining, both channel identities and counts vary ($C \in \{C_1, C_2, \dots\}$). Channel-indexed parameters require adaptation when the channel set changes; even with C-independent parameters, efficient interaction and transfer of learned relations remain separate challenges. Our objective is to learn one shared interaction module across these datasets while retaining channel-aligned forecasts.

3 Related Work

Temporal modeling. Neural time series methods cover recurrent forecasting (Sagheer & Kotb, 2019), probabilistic prediction (Salinas et al., 2020; Rangapuram et al., 2018; Wang et al., 2019; Park et al., 2022; Rasul et al., 2021), and missingness-aware sequence modeling (Che et al., 2018). Temporal designs include efficient attention (Li et al., 2019; Zhou et al., 2021), decomposition and periodic/frequency modeling (Wu et al., 2021; Zhou et al., 2022; Dai et al., 2024; Yi et al., 2023), multi-scale or multi-period representations (Wang et al., 2023; Wu et al., 2023; Wang et al., 2024a), stationarity handling (Liu et al., 2022), and lightweight forecasting (Das et al., 2023; Lin et al., 2024). Unicorn builds on temporal modeling, with learned periodic features serving as guidance for cross-channel interaction across different datasets.

Channel interaction. Channel-independent (CI) models such as DLinear (Zeng et al., 2023) and PatchTST (Nie et al., 2023) process variates separately with shared parameters. Channel-dependent (CD) models exchange information through graphs (Wu et al., 2020), variate attention (Liu et al., 2024a), cointegration-aware attention (Liu et al., 2025a), or exogenous-variable cross-attention (Wang et al., 2024b). Compact interaction mechanisms include routers (Zhang & Yan, 2023), aggregate–redistribute cores (Han et al., 2024), sparse channel relations (Qiu et al., 2025; Zhou et al., 2024), and channel clustering (Chen et al., 2024). CCM also supports zero-shot transfer, while Set Trans-

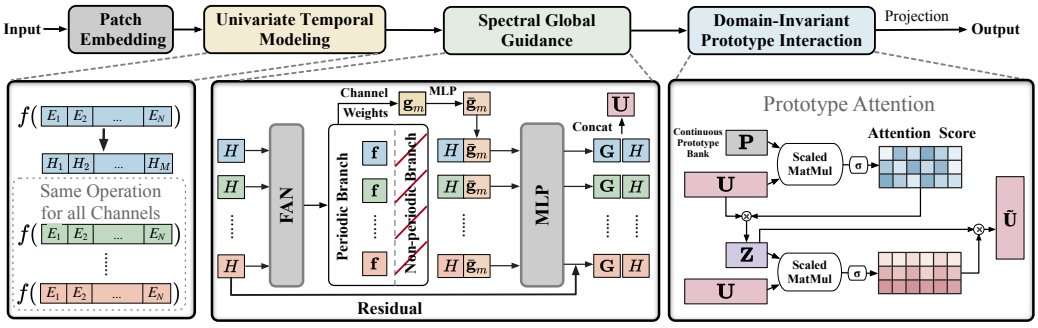


Figure 2: **The Unicorn architecture.** *Temporal Extraction (Left):* Scalable univariate modeling via shared patch embedding. *Frequency Guidance (Middle):* A Fourier Analysis Network scores latent periodic features to form global context ($\bar{g}_m$); it does not compute a temporal FFT. *Prototype Interaction (Right):* Channel-to-prototype cross-attention (U, P) produces inter-channel features ($\tilde{U}$) decoupled from channel identities.

former (Lee et al., 2019) introduced inducing-point attention for linear-cost set interaction. Efficient interaction and cross-dataset reuse are distinct design goals. Unicorn addresses sparse predictive dependencies in high-dimensional time series through a small set of shared interaction patterns, using a continuous prototype bank independent of channel identity (Appendix E.1).

Time series pretraining. Pretraining research includes adapting language models (Zhou et al., 2023; Cao et al., 2024), dedicated forecasting models (Liu et al., 2024b; Shi et al., 2025; Liu et al., 2025b; Garza et al., 2023), and general-purpose time series models (Goswami et al., 2024). TimesFM (Das et al., 2024) and Chronos (Ansari et al., 2024) primarily model individual series; Kronos (Shi et al., 2026) jointly tokenizes one instrument's fields without cross-asset interaction in our setting (C denotes assets). CD foundation models include Moirai (Woo et al., 2024) with any-variate attention and Toto (Cohen et al., 2025) with factorized temporal and variate attention. Unicorn focuses on jointly pretraining a shared correlation basis across changing channel sets, with linear interaction cost in C for many-channel, short-history data.

4 Method

To enable domain scalability, Unicorn is founded upon a guiding principle: *inter-channel dependencies should be modeled in a shared latent interaction space rather than via explicit channel-to-channel parameterization.* The core idea is to organize important predictive dependencies through shared interaction patterns represented by K learnable prototypes, with far fewer prototypes than channels ($K \ll C$). This forms a shared correlation vocabulary for pretraining across datasets. Channels selectively exchange features through this shared basis, while their relations remain input-dependent.

Unicorn consists of three components: (i) a channel-wise encoder that extracts univariate temporal representations (Section 4.1); (ii) a global feature module that uses learned periodic features to guide representation learning (Section 4.2); and (iii) a latent interaction module that models channel-to-prototype dependencies in the latent space (Section 4.3). We further detail the training pipeline in Section 4.4 and analyze computational complexity in Section 4.5.

4.1 Temporal Modeling

Given $\mathbf{X} \in \mathbb{R}^{C \times I}$, we partition each channel into non-overlapping patches of length S, resulting in $N = \lceil I/S \rceil$ patches. Each temporal patch is linearly projected into a D-dimensional token, producing patch tokens $\mathbf{E} \in \mathbb{R}^{C \times N \times D}$. Unicorn adopts TimeBridge (Liu et al., 2025a) as the temporal backbone for channel-independent modeling, and applies Integrated Attention to patches *within each channel* to capture intra-channel temporal dependencies, while mitigating the effects of non-stationarity over short time scales.

To model long-term dependencies, patch tokens are further downsampled along the temporal dimension from N to M tokens ($M < N$), producing long-term tokens $\mathbf{H} \in \mathbb{R}^{C \times M \times D}$ with $M \ll I$.

Each slice $\mathbf{H}_{:,m}$ corresponds to a coarse temporal abstraction at index m and serves as the input to subsequent inter-channel modules. Up to this stage, all operations are channel-independent, which supports pretraining across datasets with heterogeneous channel sets.

4.2 Spectral Global Guidance

Recurring periodic patterns can provide useful cues for domain-reusable channel modeling. We use learned periodic features to construct shared inter-channel context through the proposed Spectral Global Guidance module.

Periodic feature extraction. To represent periodic structure, we employ a Fourier Analysis Network (FAN) (Dong et al., 2025) as a shared feature extractor. Given $\mathbf{H} \in \mathbb{R}^{C\times M\times D}$ from the channel-wise encoder, we retain FAN's learned sinusoidal branch outputs for channel weighting and exclude its non-periodic branch from the guidance descriptor. These mappings act on latent feature vectors rather than performing a temporal FFT. For each channel and long-term index, FAN produces a periodicity-aware feature descriptor $\mathbf{f}_{c,m}$:

$$\mathbf{f}_{c,m} = \mathrm{FAN}(\mathbf{H}_{c,m,:}) \in \mathbb{R}^{d_f}, \tag{1}$$

where $\mathrm{FAN}(\cdot)$ denotes the retained periodic-feature output, with parameters shared across channels and datasets. The original temporal features remain available through the residual and concatenation paths below. Explicit temporal-axis FFT is used only in the spectral loss (Eq. (8)).

Aggregate-redistribute mechanism. Building on these descriptors, we synthesize a global inter-channel context with an aggregate-redistribute mechanism, without introducing dataset-specific model parameters. For each long-term index m, we compute normalized channel importance scores $a_{c,m}$:

$$s_{c,m} = \mathbf{w}^\top \mathbf{f}_{c,m}, \quad a_{c,m} = \frac{\exp(s_{c,m})}{\sum_{j=1}^{C}\exp(s_{j,m})}, \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^{d_f}$ is learnable and $\sum_{c=1}^{C} a_{c,m} = 1$ for each m. These scores determine the relative contribution of each channel to a shared global representation at each index m:

$$\mathbf{g}_m = \sum_{c=1}^{C} a_{c,m}\, \mathbf{H}_{c,m,:} \in \mathbb{R}^{D}. \tag{3}$$

The aggregated global token $\mathbf{g}_m$ is refined via a dimension-wise MLP denoted by $\phi(\cdot)$, yielding the refined context $\bar{\mathbf{g}}_m = \phi(\mathbf{g}_m) \in \mathbb{R}^D$. This context is subsequently redistributed to individual channels through a residual fusion process:

$$\mathbf{G}_{c,m,:} = \mathbf{H}_{c,m,:} + \psi\big([\mathbf{H}_{c,m,:}; \bar{\mathbf{g}}_m]\big) \in \mathbb{R}^{D}, \tag{4}$$

where $[\cdot\,;\cdot]$ denotes concatenation along the feature dimension, and $\psi(\cdot)$ is an MLP shared across channels. This stage concludes by enriching the original CI features with spectral-aware global context. For each m, we construct the augmented representation:

$$\mathbf{U}_{:,m,:} = [\mathbf{H}_{:,m,:}; \mathbf{G}_{:,m,:}] \in \mathbb{R}^{C\times 2D}. \tag{5}$$

By preserving univariate temporal features while adding periodicity-aware global context, $\mathbf{U}$ provides the input for subsequent prototype-mediated interaction.

4.3 Channel-Prototype Interaction

To learn transferable inter-channel dependencies, Unicorn introduces a continuous prototype bank and models interactions within a shared latent space. This architecture replaces dense $C \times C$ attention with (i) *a set of latent channel interaction prototypes* and (ii) *an input-adaptive channel-prototype aggregation and redistribution mechanism.*

Continuous prototype bank. Given the spectral-temporal representations $\mathbf{U}_{:,m,:} \in \mathbb{R}^{C\times 2D}$ from the previous stage, we maintain a learnable set of latent prototypes $\mathbf{P} \in \mathbb{R}^{K\times 2D}$, where K is fixed and independent of the channel dimension ($K \ll C$). Each prototype is a continuous interaction anchor shared across datasets, while channels are softly assigned based on current content. No quantization, discrete token indices, or commitment loss is used. For each long-term index m, inter-channel interaction follows a cascaded aggregate-redistribute attention mechanism.

Channel-to-prototype aggregation. The prototypes act as queries to *read* from channel-wise representations, forming prototype-specific summaries:

$$\begin{aligned}\mathbf{Z}_m &= \mathrm{Attn}(\mathbf{P}, \mathbf{U}_{:,m,:}, \mathbf{U}_{:,m,:}) \\ &= \mathrm{softmax}\left(\frac{\mathbf{P}\mathbf{U}_{:,m,:}^{\top}}{\sqrt{2D}}\right)\mathbf{U}_{:,m,:} \in \mathbb{R}^{K\times 2D}.\end{aligned} \tag{6}$$

The assignment matrix $\mathbf{A}_m = \mathrm{softmax}(\mathbf{P}\mathbf{U}_{:,m,:}^{\top}/\sqrt{2D}) \in \mathbb{R}^{K\times C}$ normalizes over channels, producing K input-dependent summaries. The prototype parameters are shared across datasets; the summaries $\mathbf{Z}_m$ are not fixed across inputs.

Prototype-to-channel redistribution. Once the channel dependencies are computed within the compact prototype space, this stage redistributes the refined signals back to the original channels. We reuse spectral-temporal augmented representations to query the prototype summaries and retrieve interaction-enhanced features:

$$\begin{aligned}\tilde{\mathbf{U}}_{:,m,:} &= \mathrm{Attn}(\mathbf{U}_{:,m,:}, \mathbf{Z}_m, \mathbf{Z}_m) \\ &= \mathrm{softmax}\left(\frac{\mathbf{U}_{:,m,:}\mathbf{Z}_m^{\top}}{\sqrt{2D}}\right)\mathbf{Z}_m \in \mathbb{R}^{C\times 2D}.\end{aligned} \tag{7}$$

The redistribution matrix $\mathbf{B}_m = \mathrm{softmax}(\mathbf{U}_{:,m,:}\mathbf{Z}_m^{\top}/\sqrt{2D}) \in \mathbb{R}^{C\times K}$ normalizes over prototypes and propagates the summaries back to the original channels. The resulting features are integrated with the original representations via residual connections and a dimension-wise MLP. A shared forecasting head then produces O future values per channel. Outputs retain the input channel order; Appendix E.1 details this property.

4.4 Training Scheme

Spectral-temporal objective. To ensure the model captures both point-wise temporal fidelity and global periodic structures, Unicorn is supervised through a dual-domain objective. Given the ground-truth sequence $\mathbf{Y} \in \mathbb{R}^{C\times O}$ and the prediction $\hat{\mathbf{Y}}$, we define the forecasting loss as:

$$\mathcal{L}_{\text{pred}} = (1-\alpha)\frac{1}{B}\sum_{i=1}^{B}\left|Y_i - \hat{Y}_i\right| + \alpha\frac{1}{B}\sum_{i=1}^{B}\left|\mathrm{FFT}(Y_i) - \mathrm{FFT}(\hat{Y}_i)\right|, \tag{8}$$

where $\mathrm{FFT}(\cdot)$ is applied along the prediction-time axis separately for each channel, B is the batch size, and absolute errors are reduced over the channel and time/frequency entries. The weight α balances the two terms.

Multi-dataset pretraining. During multi-dataset pretraining, Unicorn optimizes Eq. (8) across datasets with varying channel counts. All model components, including Spectral Global Guidance and the Continuous Prototype Bank (with fixed size K), share parameters across channels. No interaction parameter is indexed by a named channel. This permits joint training across different channel sets without resizing the interaction module.

Regularized finetuning. During target-domain finetuning, all model parameters are updated unless an ablation explicitly freezes them. To encourage reuse of the pretrained interaction basis, we regularize prototype drift. Let $\mathbf{P}^{(0)}$ be the pretrained prototype matrix; the objective is:

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda\left\|\mathbf{P} - \mathbf{P}^{(0)}\right\|_F^2, \tag{9}$$

where λ governs the trade-off between preservation and adaptation. This encourages preservation of the prototype basis while allowing the encoder, assignments, and forecasting head to adapt.

4.5 Complexity and Scalability

The *Channel-Prototype Interaction* module replaces dense $C \times C$ attention with a factorized $C \to K \to C$ pathway. We analyze routing cost in terms of C, D, and K. Substituting Eq. (6) into Eq. (7) yields the following composite form: $\tilde{\mathbf{U}}_{:,m,:} = \mathbf{B}_m\mathbf{A}_m\mathbf{U}_{:,m,:}$. Thus, Unicorn applies a factorized interaction operator rather than computing a dense $C \times C$ affinity matrix:

- *Dense channel attention:* All-pairs attention costs $\mathcal{O}(C^2D)$ per long-term index. This cost comes from evaluating a separate similarity score for each channel pair, using shared feature projections.

- *Unicorn:* Factorized routing costs $\mathcal{O}(CKD)$ per index, or $\mathcal{O}(MCKD)$ over all M indices, without materializing $\mathbf{B}_m\mathbf{A}_m$. This composite operator has rank at most K; the bound concerns the routing operator, not the full nonlinear model.

For domain scalability, Unicorn combines interaction parameters shared across datasets with assignments that adapt to the current input:

- *Universal interaction vocabulary:* The prototype bank $\mathbf{P}$ provides continuous interaction anchors shared across datasets, without channel-ID embeddings or a learned $C \times C$ table.
- *Efficient domain transfer:* The matrices $\mathbf{A}_m$ and $\mathbf{B}_m$ map current channel features to and from the shared basis. Pretraining learns reusable anchors across channel sets, and full-model finetuning adapts features and routing to each target. Appendix E.6 reports training time, inference speed, and peak memory consumption.

5 Experiments

5.1 Experimental Setups

We evaluate Unicorn in two settings: (i) financial pretraining followed by target-market finetuning, and (ii) training on individual public forecasting benchmarks, with a separate comparison to financial pretraining followed by finetuning.

Setting I: Cross-domain pretraining. This protocol evaluates reuse of channel relations across financial datasets. Unicorn follows a two-stage process:

- *Pretraining:* Unicorn is pretrained on a multi-market financial dataset comprising 14,386 stocks from the A-share, U.S., and Hong Kong markets.
- *Adaptation:* All parameters are finetuned on the target training split and evaluated on (i) 587 A-share stocks and (ii) 88 NASDAQ-100 constituent stocks.

The evaluation period spans from July 1, 2024, to October 20, 2025. We use standard metrics in quantitative finance, including the Information Coefficient (IC) and RankIC. We compare CD models (iTransformer (Liu et al., 2024a), TimeBridge (Liu et al., 2025a), SOFTS (Han et al., 2024)) and pretrained baselines (TimesFM 2.0 (Das et al., 2024), Kronos (Shi et al., 2026)). PatchTST, DLinear, and SOFTS use our financial corpus; foundation baselines use released checkpoints followed by full-model finetuning. Appendix D details these distinctions.

Setting II: Domain-specific training. This setting first compares models trained from scratch on individual benchmarks (Traffic, Crime-Chicago, Electricity, and Wiki-People). We also evaluate financial pretraining followed by full-model finetuning on each benchmark's training split. To assess cross-frequency robustness, we also include Traffic-Daily and ECL-Daily, which are generated via temporal aggregation. The lookback window is fixed at $I = 96$. We forecast $O = 96$ future steps for most datasets and $O = 12$ for the monthly Crime-Chicago dataset.

Comprehensive details on data preparation and metrics are provided in Appendices A–B.

5.2 Results on Cross-Domain Pretraining and Adaptation

Table 1 presents financial pretraining and finetuning results, where Unicorn leads the compared models on both target markets. Unicorn jointly pretrains on complete market panels with different channel counts through shared parameters and linear-cost interaction. Its prototype bank learns reusable patterns across the three markets for adaptation to target stock pools. Compared with the *CI foundation models* (TimesFM, Kronos) that are also finetuned on the target benchmarks, Unicorn achieves better performance by flexibly modeling inter-channel correlations. Among models pretrained on the same financial corpus, Unicorn delivers the largest improvements (Δ). Together with prototype freezing and reinitialization (Appendix E.3), these results demonstrate effective reuse of learned interaction structure.

5.3 Results on Domain-Specific Learning

Table 2 reports both target-only training and financial pretraining followed by target finetuning. Without pretraining, Unicorn is competitive overall and improves on TimeBridge on Crime-Chicago,

Table 1: **Cross-domain scalability results on high-dimensional financial benchmarks.** IC/RankIC are in 10^{-2} (higher is better), with mean and standard deviation over three seeds. $^{\ominus}$ denotes target-only training. The gain is $\Delta = (x_{\text{pre}} - x^{\ominus}_{\text{scratch}}) \times 10^3$. Baseline training details are in Appendix D.

Model	Pretrain Data	Performance (10^{-2})				Promotion Δ (10^{-3})			
		A-share		NASDAQ-100		A-share		NASDAQ-100	
		IC	RankIC	IC	RankIC	ΔIC	ΔRankIC	ΔIC	ΔRankIC
iTransformer$^{\ominus}$	None	-0.92±0.03	-1.17±0.04	-0.48±0.02	-0.85±0.03	–	–	–	–
TimeBridge$^{\ominus}$	None	1.83±0.04	2.31±0.03	0.72±0.03	1.06±0.05	–	–	–	–
PatchTST$^{\ominus}$	None	0.65±0.02	0.87±0.03	0.44±0.01	0.51±0.01	–	–	–	–
PatchTST	Financial	0.67±0.03	0.93±0.03	0.39±0.01	0.44±0.02	0.2	0.6	-0.5	-0.7
DLinear$^{\ominus}$	None	-2.79±0.06	-2.26±0.07	-3.60±0.04	-3.09±0.05	–	–	–	–
DLinear	Financial	-2.74±0.04	-2.29±0.05	-3.52±0.10	-2.94±0.08	0.5	-0.3	0.8	1.5
SOFTS$^{\ominus}$	None	-1.58±0.03	-1.39±0.02	-2.25±0.04	-2.07±0.05	–	–	–	–
SOFTS	Financial	-1.03±0.02	-0.89±0.04	-1.94±0.02	-1.56±0.03	5.5	5.0	3.1	5.1
TimesFM 2.0	Time series	0.32±0.01	0.55±0.00	-0.17±0.00	-0.31±0.01	–	–	–	–
Kronos	Global financial	1.83±0.02	2.31±0.05	1.22±0.04	1.89±0.02	–	–	–	–
Unicorn$^{\ominus}$	None	1.85±0.03	2.37±0.04	1.39±0.01	1.95±0.04	–	–	–	–
Unicorn	Financial	**2.57±0.04**	**3.01±0.03**	**2.38±0.02**	**2.51±0.02**	**7.2**	**6.4**	**9.9**	**5.6**

Table 2: **Forecasting performance on public benchmarks.** We report mean MSE and MAE (lower is better) over three random seeds. Baselines and Unicorn$^{\ominus}$ are trained from scratch. Unicorn uses financial pretraining followed by full-model finetuning on the target training split. Pretraining generally improves results, with small metric-dependent variations.

Model	Traffic		Traffic-Daily		Crime		Electricity		ECL-Daily		Wiki-People	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
iTransformer	0.392	0.269	0.650	0.401	1.202	0.667	0.148	0.239	0.289	0.351	2.449	0.893
PatchTST	0.360	0.249	0.582	0.351	1.230	0.670	0.132	0.227	0.275	0.339	2.651	0.972
DLinear	0.410	0.282	0.838	0.515	1.268	0.730	0.141	0.236	0.281	0.340	2.510	0.916
SOFTS	0.377	0.250	0.623	0.390	1.094	0.631	0.143	0.233	0.290	0.355	2.226	0.853
TimeBridge	**0.341**	0.241	0.514	0.300	1.116	0.624	0.122	0.215	0.258	0.306	2.239	0.868
Unicorn$^{\ominus}$	0.343	0.239	0.513	0.301	0.925	0.527	**0.120**	0.212	0.265	0.308	2.107	0.831
Unicorn	**0.341**	**0.238**	**0.497**	**0.294**	**0.919**	**0.526**	0.121	**0.211**	**0.241**	**0.295**	**2.001**	**0.816**

Electricity, and Wiki-People, with the largest MSE reductions on Crime-Chicago and Wiki-People. This target-only comparison separates the complete architecture's contribution from the benefit of financial pretraining. Module ablations are provided below.

We further examine financial pretraining followed by full-model finetuning on non-financial data. The gains are generally modest, consistent with differences in source and target distributions. Pretraining improves most reported metrics, with larger gains on ECL-Daily and Wiki-People. Joint pretraining that also includes the target training split yields clearer gains on Traffic and Electricity (Appendix E.5); validation and test data remain held out.

5.4 Ablation Studies

We conduct multiple ablation studies to quantify the contribution of each modular component in Unicorn under the domain-specific training regime (Setting II). Following the modular architecture presented in Section 4.2–4.4, we evaluate three key components of the framework:

- *w/o Spectral guidance:* Removes the FAN-based spectral extractor and the aggregate-redistribute pathway. The model relies solely on univariate features in the prototype interaction module.
- *w/o Prototype interaction:* Replaces the prototype bank and cascaded attention with direct channel-level interaction; it does not remove all cross-channel computation.

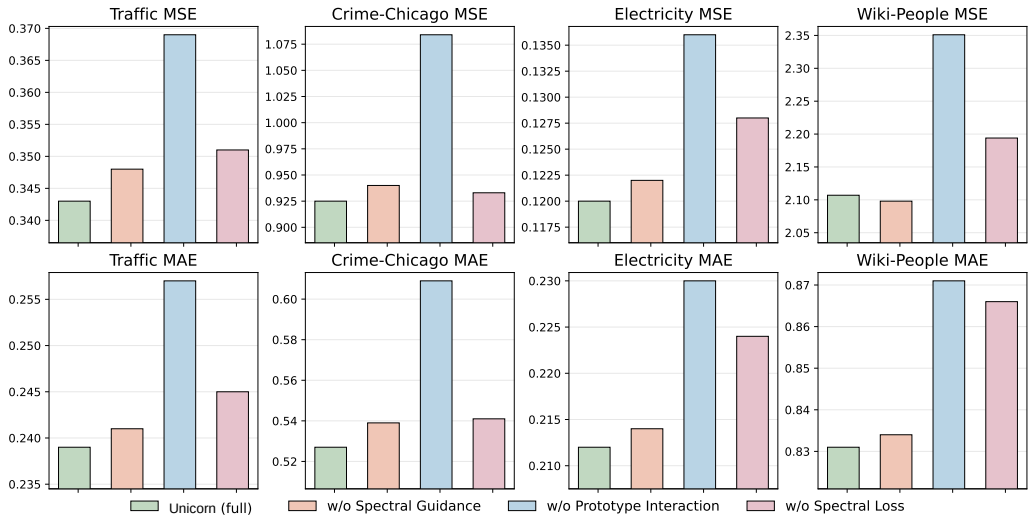


Figure 3: **Ablation studies under the domain-specific training setting.** Replacing Prototype Interaction with direct channel interaction leads to the most significant performance drop, validating its role for channel correlation modeling. Both Spectral Guidance and the Spectral Loss provide complementary gains in forecasting accuracy.

- *w/o Spectral loss:* Maintains the full architecture but trains exclusively with a time-domain objective, dropping the frequency-domain term from our hybrid loss function.

The results in Figure 3 support the complementary contributions of the three components. Replacing *Prototype Interaction* with direct interaction produces the largest drop, indicating the importance of the prototype bottleneck in these settings. Without *Spectral Guidance*, forecasting error generally increases, consistent with the benefit of global context. Omitting *Spectral Loss* increases error on the evaluated datasets, supporting complementary temporal and spectral supervision.

5.5 Sensitivity to Prototype Size

The number of latent prototypes K controls the capacity of the shared interaction space. The default is $K = 16$ for all experiments; only this sensitivity study varies K, with all other components and training settings unchanged. Table 3 shows that Unicorn exceeds Kronos on all four metrics for $K \geq 4$ among the tested values, while $K = 2$ is insufficient on RankIC. The best value differs between markets. For the A-share market ($C = 587$), performance metrics (IC and RankIC) improve steadily as K increases. This suggests that larger, more heterogeneous stock pools require a higher-capacity latent space to capture the diverse array of interaction roles present in the market. Conversely, on NASDAQ-100 ($C = 88$), peak performance occurs at $K = 8$, although the default remains $K = 16$ for all other experiments.

5.6 Scalability and Data Efficiency

In Table 4, we evaluate Unicorn on the A-share dataset (Setting I) under three challenging regimes: channel scalability, robustness to missing data, and few-shot transferrability.

Channel scalability. We examine the scalability of Unicorn by varying the number of available variables during both adaptation and inference (we use the same subset of channels in both phases). Unicorn presents a clear positive scaling effect: as the channel count increases, its performance improves consistently, with IC rising from 0.0232 (at 25% capacity) to 0.0257 (full set). In contrast, Kronos peaks at 50% channels on IC and 25% on RankIC, with both metrics lower at full capacity. These trends support the usefulness of additional channels for Unicorn, but changing the stock subset also changes evaluation composition, so they do not establish an intrinsic performance ceiling for CI models. Unicorn accommodates these larger variable sets without the quadratic overhead of dense channel attention.

Robustness to channel omission. We evaluate the robustness of our model to missing channels at inference time, which is a frequent challenge in real-world deployment. Unicorn demonstrates

Table 3: **Impact of prototype size K under the cross-domain pretraining setting.** Values are raw IC/RankIC (higher is better). Only this study varies K; all other experiments use $K = 16$. All remaining components and training settings are held fixed. The best tested prototype size differs between the two target markets.

# Prototypes	A-share		NASDAQ-100	
	IC	RankIC	IC	RankIC
Kronos	0.0183	0.0231	0.0122	0.0189
2	0.0187	0.0194	0.0185	0.0170
4	0.0214	0.0236	0.0231	0.0248
8	0.0249	0.0289	**0.0241**	**0.0260**
16	0.0257	0.0301	0.0238	0.0251
32	**0.0259**	**0.0308**	0.0226	0.0247

Table 4: **Channel scalability, robustness, and few-shot analyses on the A-share dataset.** See text for detailed configurations used in the target-domain finetuning phase and the test phase.

Evaluation	Unicorn		Kronos	
	IC	RankIC	IC	RankIC
Channel scalability: Train = Test				
25% channels	0.0232	0.0274	0.0195	**0.0243**
50% channels	0.0239	0.0292	**0.0197**	0.0236
100% channels	**0.0257**	**0.0301**	0.0183	0.0231
Channel robustness: Train = full, Test = partial				
50% channels	0.0244	0.0297	0.0187	0.0229
25% channels	0.0238	0.0280	0.0171	0.0215
Few-shot finetuning				
Full FT data	0.0257	0.0301	0.0183	0.0231
50% FT data	0.0241	0.0295	0.0146	0.0185
25% FT data	0.0238	0.0288	0.0108	0.0160

remarkable stability. Even when 75% of channels are missing at test time, it retains an IC of 0.0238, which is significantly higher than its CI counterpart. Despite lacking explicit inter-channel modeling, Kronos also scores lower on both metrics at 25% channel availability than with all channels, although its IC rises slightly at 50%. This suggests that the performance degradation in both models is not solely a result of losing relational information, but also reflects the stochasticity and variance inherent in the channel sampling process.

Few-shot adaptation and data efficiency. This final regime evaluates the model's capacity to adapt under severe data scarcity. As the finetuning data is reduced to 25%, Unicorn maintains an IC of 0.0238, representing only a marginal drop from its full-data performance. In contrast, the CI baseline (Kronos) suffers a significant performance collapse, with its IC reducing from 0.0183 to 0.0108. This disparity highlights that, compared to CI models, the latent channel-prototype interaction mechanism serves as a robust structural inductive bias. The pretrained interaction basis provides a useful starting point under limited target supervision. All parameters are still finetuned; the prototype-freezing and reinitialization controls in Appendix E.3 provide more direct evidence of basis reuse.

5.7 Additional Analyses

We provide supplementary analyses in Appendix E. These include: (i) structural fidelity assessments using Corr-MAE, ACF-MAE, and PSD-L1 to evaluate the preservation of system dynamics (Appendix E.2); (ii) an investigation into prototype adaptation and drift during finetuning (Appendix E.3); (iii) an analysis of cross-domain prototype reuse and functional specialization (Appendix E.4); and (iv) an expanded study on pretraining robustness under distribution shift (Appendix E.5).

6 Conclusions and Limitations

We introduced Unicorn, a unified framework for scalable, multi-dataset pretraining on high-dimensional time series. By modeling inter-channel correlations through a shared prototype space and using periodicity-aware global guidance, Unicorn combines linear-cost channel interaction with joint pretraining across different channel sets. Experiments across financial and public forecasting benchmarks demonstrate effective reuse of learned interaction structure, competitive accuracy, robustness to missing channels, and data-efficient finetuning, suggesting a path toward multivariate time series foundation models.

Transfer depends on the compatibility of source and target data: financial-only pretraining yields modest gains on non-financial benchmarks. The low-rank routing bottleneck may also limit interactions that require larger capacity. This work focuses on offline forecasting of continuous-valued series. Broader pretraining coverage, streaming observations, and mixed-type variables remain important directions for future work.

AI USE STATEMENT

We used LLM tools solely for text editing, grammar correction, language polishing, and manuscript formatting. The authors remain responsible for the final manuscript, including all claims, results, and references.

ETHICS STATEMENT

Unicorn is evaluated on historical financial data and public time series benchmarks. Historical data can reflect biases and changes in the underlying systems. Deployment in financial or public-safety applications requires independent validation, assessment of distribution shift and potential harms, and appropriate human oversight. The reported forecasting metrics measure predictive performance rather than deployment safety or realized financial returns. Any downstream reuse or release of the data should respect the original data providers' licenses, access conditions, and applicable privacy requirements.

REPRODUCIBILITY STATEMENT

Sections 4.1–4.4 describe the model components and training objectives, and Section 4.5 analyzes computational complexity. Appendix A describes the datasets and temporal evaluation setting, while Appendix B defines the evaluation metrics. Appendices C and D document baseline configurations, preprocessing, training protocols, hyperparameters, and hardware. Appendix E.1 provides further details of the channel-prototype interaction and its permutation equivariance. Anonymized code is provided in the supplementary material.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Defu Cao, Furong Jia, Sercan Ö. Arık, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In *International Conference on Learning Representations*, 2024.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1):6085, 2018.

Jialin Chen, Jan Eric Lenssen, Aosong Feng, Weihua Hu, Matthias Fey, Leandros Tassiulas, Jure Leskovec, and Rex Ying. From similarity to superiority: Channel clustering for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130635–130663, 2024.

Ben Cohen, Emaad Khwaja, Youssef Doubli, Salahidine Lemaachi, Chris Lettieri, Charles Masson, Hugo Miccinilli, Elise Ramé, Qiqi Ren, Afshin Rostamizadeh, Jean Ogier du Terrail, Anna-Monica Toon, Kan Wang, Stephan Xie, Zongzhe Xu, Viktoriya Zhukova, David Asker, Ameet Talwalkar, and Othmane Abou-Amal. This time is different: An observability perspective on time series foundation models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 50907–50951, 2025.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *International Conference on Learning Representations*, 2024.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167, 2024.

Yihong Dong, Ge Li, Yongding Tao, Xue Jiang, Kechi Zhang, Jia Li, Jinliang Deng, Jing Su, Jun Zhang, and Jingjing Xu. Fan: Fourier analysis networks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 66267–66296, 2025.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152, 2024.

Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. Softs: Efficient multivariate time series forecasting with series-core fusion. In *Advances in Neural Information Processing Systems*, volume 37, pp. 64145–64175, 2024.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, 2019.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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*, volume 235 of *Proceedings of Machine Learning Research*, pp. 30211–30226, 2024.

Peiyuan Liu, Beiliang Wu, Yifan Hu, Naiqi Li, Tao Dai, Jigang Bao, and Shu-Tao Xia. Timebridge: Non-stationarity matters for long-term time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39815–39840, 2025a.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9881–9893, 2022.

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*, 2024a.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32369–32399, 2024b.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39295–39317, 2025b.

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.

Youngsuk Park, Danielle Maddix, François-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8127–8150, 2022.

Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. DUET: Dual clustering enhanced multivariate time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 1185–1196, 2025.

Syama Sundar Rangapuram, Matthias W. Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8857–8868, 2021.

Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations*, 2025.

Yu Shi, Zongliang Fu, Shuo Chen, Bohan Zhao, Wei Xu, Changshui Zhang, and Jian Li. Kronos: A foundation model for the language of financial markets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(30):25366–25373, 2026.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *International Conference on Learning Representations*, 2023.

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 *International Conference on Learning Representations*, 2024a.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *Advances in Neural Information Processing Systems*, volume 37, pp. 469–498, 2024b.

Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6607–6617, 2019.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020.

Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Ning An, Defu Lian, Longbing Cao, and Zhendong Niu. Frequency-domain mlps are more effective learners in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 76656–76679, 2023.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):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 *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286, 2022.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43322–43355, 2023.

Xin Zhou, Weiqing Wang, Wray Buntine, Shilin Qu, Abishek Sriramulu, Weicong Tan, and Christoph Bergmeir. Scalable transformer for high dimensional multivariate time series forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 3515–3526, 2024.

APPENDIX

A DATASET DETAILS

Pretraining dataset. The pretraining corpus consists of large-scale financial time series collected from three major equity markets: the Chinese A-share market, the U.S. stock market, and the Hong Kong stock market. The dataset spans from January 1, 2010 to December 31, 2023, covering multiple market regimes and economic cycles. Series are aligned on trading days within each market and contain open, high, low, close, and volume (OHLCV) fields for each stock. The market-specific sequence lengths are reported in Table 5. The dataset is split chronologically for training and validation to avoid any temporal leakage.

Evaluation datasets. Evaluations are conducted on both financial and public multivariate forecasting benchmarks. For financial forecasting, we consider two test sets: (i) an A-share subset containing 587 actively traded stocks, and (ii) the NASDAQ-100 constituent stocks with 88 equities. Both test sets use the same evaluation window from July 1, 2024 to October 20, 2025, covering 341 trading days in total. Models are trained on historical data before this period and evaluated strictly on the held-out interval. In addition, we evaluate on widely used public multivariate benchmarks, including Traffic, Electricity (ECL), Crime-Chicago, and Wiki-People. Traffic and ECL are hourly datasets, while Crime-Chicago and Wiki-People are monthly and daily datasets, respectively. For the public benchmarks, we follow the chronological training, validation, and test splits used in prior work.

Daily-resolution variants. To examine robustness under different temporal granularities, we further construct Traffic-Daily and ECL-Daily by aggregating the original hourly data into daily frequency. Specifically, values within each day are aggregated by averaging, resulting in sequences whose lengths are approximately one twenty-fourth of the original hourly series while preserving the same channel dimensionality.

Table 5: **Summary of datasets used in pretraining and evaluation.** "Dim" denotes the number of channels. "Length" refers to the total number of time steps. Traffic-Daily and ECL-Daily are obtained by aggregating the original hourly data to daily frequency.

Dataset	Domain	Dim	Length	Frequency
A-share (pretrain)	Finance	4932	3,168	Trading day
U.S. stocks (pretrain)	Finance	7036	3,273	Trading day
Hong Kong stocks (pretrain)	Finance	2418	3,206	Trading day
A-share (test)	Finance	587	341	Trading day
NASDAQ-100	Finance	88	341	Trading day
Traffic	Transportation	862	17,545	Hourly
Traffic-Daily	Transportation	862	729	Daily
Electricity (ECL)	Energy	321	26,304	Hourly
ECL-Daily	Energy	321	1,096	Daily
Crime-Chicago	Public Safety	1,155	260	Monthly
Wiki-People	Web Traffic	6,107	550	Daily

B EVALUATION METRICS

Overview. This paper considers two evaluation settings: (i) high-dimensional forecasting on public multivariate benchmarks (Setting II), and (ii) cross-market stock return prediction under domain adaptation (Setting I). Accordingly, we report error-based forecasting metrics (MSE/MAE) and correlation-based financial metrics (IC/RankIC).

Notation. For metric definitions, write the future sequence in time-first order as $\mathbf{Y} \in \mathbb{R}^{O\times C}$, with horizon O and C channels, and let $\hat{\mathbf{Y}} \in \mathbb{R}^{O\times C}$ be the corresponding predictions. We denote the value at time step t and channel c as $Y_{t,c}$ and $\hat{Y}_{t,c}$, respectively.

Mean Squared Error (MSE). For Setting II, we evaluate forecasting accuracy using the Mean Squared Error (MSE), defined as

$$\mathrm{MSE} = \frac{1}{OC} \sum_{t=1}^{O} \sum_{c=1}^{C} \left(Y_{t,c} - \hat{Y}_{t,c} \right)^2 . \tag{10}$$

MSE serves as the primary metric for numerical forecasting accuracy, where lower values indicate better performance.

Mean Absolute Error (MAE). We additionally report the Mean Absolute Error (MAE), given by

$$\mathrm{MAE} = \frac{1}{OC} \sum_{t=1}^{O} \sum_{c=1}^{C} \left| Y_{t,c} - \hat{Y}_{t,c} \right| . \tag{11}$$

Compared with MSE, MAE is less sensitive to large deviations and provides a complementary assessment of prediction quality.

Financial correlation metrics. For Setting I, the model predicts a sequence of length O for each stock. Only when computing IC and RankIC do we convert the predicted and ground-truth sequences into scalar returns over the entire output window. Let $\hat{\mathbf{r}} \in \mathbb{R}^C$ and $\mathbf{r} \in \mathbb{R}^C$ denote these predicted and realized window returns across C stocks, respectively. Both terms of the spectral-temporal loss in Eq. (8) operate on the full sequences, with FFT applied along the prediction-time axis, not on the scalar returns used for evaluation.

Information Coefficient (IC). We measure the linear association between predicted signals and realized returns using the Information Coefficient (IC), defined as the Pearson correlation:

$$\mathrm{IC} = \mathrm{Corr}(\hat{\mathbf{r}}, \mathbf{r}) = \frac{\sum_{c=1}^{C} (\hat{r}_c - \bar{\hat{r}})(r_c - \bar{r})}{\sqrt{\sum_{c=1}^{C} (\hat{r}_c - \bar{\hat{r}})^2} \sqrt{\sum_{c=1}^{C} (r_c - \bar{r})^2}}. \tag{12}$$

A higher IC indicates stronger predictive consistency between the model outputs and future returns.

Rank Information Coefficient (RankIC). To further evaluate whether the model preserves the correct cross-sectional ordering, we report the Rank Information Coefficient (RankIC), computed as the Spearman rank correlation:

$$\mathrm{RankIC} = \mathrm{Corr}\big(\mathrm{rank}(\hat{\mathbf{r}}), \mathrm{rank}(\mathbf{r})\big). \tag{13}$$

RankIC is particularly important in quantitative finance, as many trading strategies rely on accurate return ranking rather than exact value prediction. At each daily forecast origin, we calculate cross-sectional correlation using returns over the corresponding output window, then average over forecast origins.

Metric interpretation. All forecasting error metrics (MSE and MAE) are *lower-is-better*, while financial correlation-based metrics (IC and RankIC) are *higher-is-better*.

C BASELINES

We compare Unicorn with *channel-independent models*, *channel-dependent models*, and *time-series foundation models*, following the evaluation protocols described in the main paper.

C.1 CHANNEL-INDEPENDENT FORECASTING MODELS

Channel-independent (CI) models process each time series independently using shared parameters, without explicitly modeling inter-channel dependencies. Parameter sharing allows the same model to process datasets with different numbers of channels.

DLinear (Zeng et al., 2023). DLinear uses a moving average to extract the trend and subtracts it from the input to obtain the seasonal component. Separate linear projections forecast the two components, and their outputs are summed. Each channel is modeled independently, without explicit inter-channel interaction.

PatchTST (Nie et al., 2023). PatchTST adopts a patch-based Transformer architecture where each channel is segmented into temporal patches and processed independently with shared Transformer layers. Attention over these patches models temporal dependencies within each channel.

C.2 Channel-Dependent Forecasting Models

Channel-dependent (CD) models exchange information among channels through attention, graph propagation, or global aggregation. Their interaction cost, dependence on channel-indexed parameters, and ability to reuse learned relations are distinct properties.

iTransformer (Liu et al., 2024a). iTransformer treats channels as tokens and applies attention across them. Its shared input and output projections are independent of C, while dense attention constructs a $C \times C$ activation map. Its reported training is dataset-specific, and we evaluate the model separately on each target dataset.

TimeBridge (Liu et al., 2025a). TimeBridge models inter-channel correlations through cointegrated attention over temporally aligned representations. Its released implementation uses dataset-specific `enc_in` for axial layout and padding, and we evaluate this implementation with target-specific training. Unicorn retains TimeBridge's channel-wise temporal backbone but changes its cross-channel pathway.

SOFTS (Han et al., 2024). SOFTS aggregates channels into a fixed-dimensional core and redistributes context using shared MLPs. Its parameters are independent of C and its interaction cost is linear in C; it therefore supports the variable-channel pretraining evaluated in Table 1. Unicorn instead maintains multiple persistent prototypes and uses two-way attention to exchange channel features through them.

C.3 Time-Series Foundation Models

Time-series foundation models learn from large time series corpora and include both per-series and multivariate architectures. We evaluate their released pretrained checkpoints after finetuning on each target training split.

TimesFM 2.0 (Das et al., 2024). TimesFM 2.0 is a decoder-only foundation model pretrained on large-scale time series data. It processes multivariate inputs in a channel-independent manner, sharing parameters across all channels and focusing on learning universal temporal patterns. While it scales effectively to datasets with varying channel dimensions, the model does not explicitly model inter-channel correlations during either pretraining or finetuning.

Kronos (Shi et al., 2026). Kronos is pretrained on over 12 billion K-line records from 45 exchanges. Its tokenizer jointly represents price and trading-activity fields within an instrument using discrete tokens. In our financial setting, channels are assets: Kronos processes assets independently, whereas Unicorn exchanges features across assets through continuous prototypes. Modeling multiple fields within one instrument is distinct from modeling cross-asset interactions.

D Implementation Details

Data preprocessing. For all datasets, input time series are normalized using z-score normalization based on statistics computed from the training split only. The same normalization parameters are applied to validation and test sets. Financial inputs contain the five OHLCV fields with a lookback of $I = 96$. The forecasting targets are full future sequences of length O; the scalar returns over these windows are used only to compute IC and RankIC. No information from the test period is used during training or validation.

Channel and feature axes. For financial data, C counts stocks, while the five OHLCV fields form a separate feature axis: $\mathbf{X}^{\text{raw}} \in \mathbb{R}^{C \times I \times 5}$. A shared learnable projection $\phi_{\text{fin}} : \mathbb{R}^5 \to \mathbb{R}$ is applied independently at each stock and timestamp to produce $\mathbf{X} \in \mathbb{R}^{C \times I}$ for Unicorn. This projection does not mix stocks. Public benchmarks already provide one scalar per channel and timestamp.

Table 6: **Hyperparameters**.

Category	Hyperparameter	Value
Architecture	hidden size D	512
	patch length S	16 (hourly) / 8 (daily)
	prototype size K	16
	dropout	0.1
Training	optimizer	AdamW
	batch size	32
	learning rate	1×10^{-4}
	spectral-temporal weight α (Eq. (8))	0.3
	prototype regularization λ (Eq. (9))	1×10^{-2}

Training protocol (Setting II). For public multivariate forecasting benchmarks (Setting II), baselines and Unicorn$^{\ominus}$ are trained from scratch using a sliding-window strategy with a fixed input length $I = 96$. Prediction horizons are set to $O = 96$ for Traffic, Electricity, and Wiki-People, and $O = 12$ for Crime-Chicago, consistent with Section 5.1. Unicorn uses the spectral-temporal loss in Eq. (8), and model selection is performed based on validation MSE. Final results are reported on the test split using MSE and MAE. The separate Unicorn row in Table 2 starts from financial pretraining and updates all parameters on the same target training split using Eq. (9). Inputs, horizons, chronological splits, and validation-based model selection are unchanged.

Pretraining and finetuning protocol (Setting I). For cross-market transfer experiments (Setting I), Unicorn is pretrained on the multi-market financial corpus described in Appendix A. Pretraining optimizes the spectral-temporal objective in Eq. (8) over heterogeneous channel dimensions. For downstream adaptation to a target stock pool, all parameters are finetuned using the regularized objective in Eq. (9), where the pretrained prototype bank $\mathbf{P}^{(0)}$ is used as the reference. All reported financial results are evaluated on the held-out test window using IC and RankIC, as defined in Appendix B. The target markets are represented in the pretraining corpus; this is a chronologically held-out evaluation, not a leave-one-market-out experiment.

Baseline pretraining. "Financial" in Table 1 denotes our 2010–2023, three-market corpus, shared by PatchTST, DLinear, SOFTS, and Unicorn. "Global financial" denotes Kronos's separate upstream corpus; its raw data are not public, so we use the released checkpoint. Foundation baselines use their native input/tokenization interfaces and full-model finetuning on the same target inputs, labels, and splits. Their upstream corpora and model sizes differ, so these comparisons are not matched-pretraining-budget experiments.

Target-only CD baselines. We train the released iTransformer and TimeBridge implementations separately on each target dataset. iTransformer's dense channel attention makes complete pretraining panels of 2,418–7,036 assets computationally impractical in our setup; TimeBridge's released axial layout and padding use a fixed `enc_in`. The target-only rows marked "None" in Table 1 reflect these implementation and computational constraints.

Optimization details. All experiments are implemented in PyTorch and run on NVIDIA GeForce RTX 4090 hardware. Models are trained using the AdamW optimizer. A cosine learning-rate schedule with linear warm-up is employed. Early stopping is used based on validation performance, and the model with the best validation metric is selected for testing.

Hyperparameter configuration. Table 6 summarizes the hyperparameters used in our implementation. These values are fixed across datasets unless explicitly stated. In particular, $K = 16$ is the default throughout; only Table 3 varies K. Full-model experiments use $\alpha = 0.3$; the loss ablation sets $\alpha = 0$.

E Additional Analyses

We examine the shared interaction basis, structural fidelity, prototype reuse, pretraining under domain shift, and computational efficiency.

E.1 Shared Interaction Basis and Channel-Aligned Outputs

What is shared. The prototype parameters $\mathbf{P}$ are shared across datasets; the attention weights $\mathbf{A}_m$, summaries $\mathbf{Z}_m$, redistribution weights $\mathbf{B}_m$, and effective routing operator $\mathbf{B}_m\mathbf{A}_m$ all depend on the current input. Thus "universal" refers to a reusable interaction basis across changing channel sets, with relations adapted to each input.

Channel-aligned outputs. With shared row-wise projections and no channel-ID embeddings, permuting the input channels permutes the output channels. For a permutation matrix $\mathbf{\Pi}$, let $\mathbf{U}'_m = \mathbf{\Pi}\mathbf{U}_m$. Row-wise softmax in Eqs. (6)–(7) gives

$$\mathbf{A}'_m = \mathbf{A}_m\mathbf{\Pi}^\top, \qquad \mathbf{Z}'_m = \mathbf{Z}_m, \qquad \mathbf{B}'_m = \mathbf{\Pi}\mathbf{B}_m, \qquad \tilde{\mathbf{U}}'_m = \mathbf{\Pi}\tilde{\mathbf{U}}_m. \tag{14}$$

The shared temporal encoder, global aggregation, row-wise fusion, and forecasting head preserve this property. Identity-agnostic parameters therefore retain channel-specific predictions. Changing the number of channels also leaves the parameter shapes unchanged, although the resulting summaries and forecasts can change.

Relation to compact interaction methods. Set Transformer's inducing-point attention (Lee et al., 2019) and Crossformer's routers (Zhang & Yan, 2023) use a small latent set to collect and return information. SOFTS (Han et al., 2024) provides a shared aggregate–redistribute core with linear cost. CCM (Chen et al., 2024) learns prototype assignments and uses them to combine cluster-specific predictor weights, including for zero-shot transfer. Unicorn designs its continuous prototype bank for sparse predictive correlations in high-dimensional time series: FAN-based guidance aligns channel representations, and two-way attention selectively exchanges their features through shared interaction anchors at each long-term index. The same bank is jointly pretrained across datasets with different channel identities and counts. DUET (Qiu et al., 2025) constructs a channel mask from pairwise frequency distances; Unicorn instead models these interactions through a fixed-K feature-routing bottleneck. This design supports reusable correlation modeling and scalable cross-dataset pretraining within one forecasting architecture.

E.2 Structure-Aware Validation Beyond Point-wise Errors

While MSE and MAE are standard metrics in multivariate forecasting, they do not directly measure whether a model preserves inter-channel dependency structure, temporal dynamics, or spectral characteristics. To complement the main results, we evaluate Unicorn and five strong baselines on three structure-aware metrics over the non-financial benchmarks, using the corresponding models and evaluation settings from Table 2.

Inter-channel correlation preservation. We define the correlation error as

$$\text{Corr-MAE} = \frac{1}{C(C-1)} \sum_{i \neq j} \left| R_{ij}(Y) - R_{ij}(\hat{Y}) \right|, \tag{15}$$

where $R(\cdot)$ denotes the channel-wise Pearson correlation matrix. This measures predictive co-movement, not recovery of a causal graph.

Temporal dynamics preservation. We measure the discrepancy in autocorrelation structure by

$$\text{ACF-MAE} = \frac{1}{CL} \sum_{c=1}^{C} \sum_{\ell=1}^{L} \left| \rho_c^{(Y)}(\ell) - \rho_c^{(\hat{Y})}(\ell) \right|, \tag{16}$$

where L is the maximum evaluated lag and $\rho_c(\ell)$ is the autocorrelation of channel c at lag ℓ.

Spectral consistency. We further quantify the discrepancy in normalized power spectra:

$$\text{PSD-L1} = \frac{1}{C} \sum_{c=1}^{C} \left\| \tilde{S}_c(Y) - \tilde{S}_c(\hat{Y}) \right\|_1, \tag{17}$$

where $\tilde{S}_c(\cdot)$ denotes the power spectrum of channel c, normalized by its total power across frequency bins.

Table 7: **Structure-aware validation on non-financial benchmarks.** We report Corr-MAE, ACF-MAE, and PSD-L1, where lower is better. Unicorn consistently outperforms all baselines in preserving inter-channel correlations, temporal dynamics, and spectral characteristics.

Model	Traffic			Crime-Chicago			Electricity		
	Corr-MAE	ACF-MAE	PSD-L1	Corr-MAE	ACF-MAE	PSD-L1	Corr-MAE	ACF-MAE	PSD-L1
iTransformer	0.126	0.102	0.159	0.177	0.128	0.198	0.090	0.081	0.124
PatchTST	0.121	0.098	0.154	0.181	0.132	0.203	0.087	0.079	0.120
DLinear	0.139	0.114	0.173	0.190	0.141	0.216	0.095	0.085	0.129
SOFTS	0.112	0.094	0.147	0.152	0.113	0.178	0.086	0.078	0.118
TimeBridge	0.108	0.091	0.140	0.161	0.119	0.184	0.081	0.074	0.111
Unicorn	**0.101**	**0.087**	**0.133**	**0.134**	**0.101**	**0.161**	**0.074**	**0.069**	**0.103**

Table 7 shows that Unicorn consistently achieves the best performance across all three structure-aware metrics on Traffic, Crime-Chicago, and Electricity. This indicates that the gain of Unicorn is not limited to lower point-wise forecasting error; it also better preserves the underlying inter-channel dependency structure, temporal evolution patterns, and spectral behavior of multivariate time series.

E.3 Prototype Drift and Reassignment During Finetuning

To examine whether finetuning reuses the pretrained interaction basis, we measure both changes in the prototype bank and changes in channel-to-prototype assignments, and evaluate freezing and reinitialization controls.

Recall from Section 4.3 that, for each sample x and temporal index m, Unicorn redistributes prototype summaries to channels. Given $U^{(x,d)}_{:,m,:}$ and $Z^{(x,d)}_m$, the redistribution matrix is

$$B^{(x,d)}_m = \mathrm{softmax}\left(\frac{U^{(x,d)}_{:,m,:}\left(Z^{(x,d)}_m\right)^\top}{\sqrt{2D}}\right) \in \mathbb{R}^{C_d \times K}, \tag{18}$$

where d denotes the target domain, C_d is the number of channels in that domain, and K is the number of latent prototypes. Each row of $B^{(x,d)}_m$ specifies how a target-domain channel combines the prototype summaries. Therefore, $B^{(x,d)}_m$ directly characterizes the channel-to-prototype alignment induced by the current model.

Let $P^{(0)} \in \mathbb{R}^{K\times 2D}$ denote the pretrained prototype bank before target-domain adaptation, and let $P^{(d)}$ denote the bank after finetuning on domain d. We define the *prototype drift* as

$$\delta^{(d)}_P = \frac{\left\|P^{(d)} - P^{(0)}\right\|_F}{\left\|P^{(0)}\right\|_F}. \tag{19}$$

This quantity measures how much the shared prototype basis itself changes during finetuning.

To describe changes in channel assignments, we define the *reassignment drift*. Let S_d be the evaluation set of target domain d, and let M be the number of long-term temporal tokens in Section 4.1. For each sample $x \in S_d$ and token index m, we compare the redistribution matrix before and after finetuning:

- $B^{(x,d,0)}_m$: computed by the pretrained model *before* target-domain finetuning;
- $B^{(x,d,\mathrm{ft})}_m$: computed by the adapted model *after* finetuning on domain d.

We then define

$$\delta^{(d)}_B = \frac{1}{|S_d|M}\sum_{x\in S_d}\sum_{m=1}^{M}\frac{\left\|B^{(x,d,\mathrm{ft})}_m - B^{(x,d,0)}_m\right\|_1}{C_d}. \tag{20}$$

Here, $\|\cdot\|_1$ denotes the entrywise ℓ_1 norm, summing absolute differences over all channel–prototype pairs. Dividing by C_d gives the average ℓ_1 change in a channel's assignment distribution, allowing

Table 8: **Prototype reuse under domain adaptation.** The first three result columns report raw IC. Full FT updates all parameters; Frozen P fixes only the pretrained prototypes; Reinit P reinitializes them before finetuning. The two drift measures have different scales and are reported descriptively.

Target	Full FT	Frozen P	Reinit P	$\delta_P^{(d)}$	$\delta_B^{(d)}$
A-share	0.0257	0.0251	0.0203	0.07	0.41
NASDAQ-100	0.0238	0.0232	0.0181	0.09	0.47

comparison across domains with different channel counts. Thus, $\delta_B^{(d)}$ measures how much channel-to-prototype alignment changes during adaptation.

In addition to standard finetuning (*Full FT*), we consider two controlled variants:

- **Frozen** P: keep the pretrained prototype bank fixed during finetuning;
- **Reinit** P: randomly reinitialize the prototype bank before finetuning.

Table 8 supports reuse of the pretrained basis. *Frozen* P nearly preserves full-finetuning IC, while *Reinit* P reduces it on both markets. The measured prototype drift is small and assignments change after finetuning. However, δ_P and δ_B use different normalizations, so their magnitudes cannot quantify the relative contribution of these two adaptation mechanisms. Encoder and head parameters also change during full finetuning.

E.4 Cross-domain Reuse and Semantic Coherence of Prototypes

We measure cross-domain prototype usage and within-prototype statistical coherence.

For each channel c in domain d, we first compute its average prototype-assignment profile:

$$\bar{b}_c^{(d)} = \frac{1}{|S_d|M} \sum_{x \in S_d} \sum_{m=1}^{M} B_m^{(x,d)}(c,:) \in \mathbb{R}^K, \tag{21}$$

and define its dominant prototype by

$$k_c^{(d)} = \arg\max_k \bar{b}_c^{(d)}(k). \tag{22}$$

We summarize domain-level prototype usage by

$$q_d(k) = \frac{1}{|S_d|MC_d} \sum_{x \in S_d} \sum_{m=1}^{M} \sum_{c=1}^{C_d} B_m^{(x,d)}(c,k), \tag{23}$$

and define the cross-domain usage similarity as

$$\mathrm{Cos}(d,d') = \frac{q_d^\top q_{d'}}{\|q_d\|_2 \|q_{d'}\|_2}. \tag{24}$$

To test whether channels grouped under the same prototype are statistically coherent, we consider three raw-series statistics: dominant period, volatility, and lag-1 autocorrelation. For each statistic s_c, we define

$$\rho_s = \frac{\frac{1}{K}\sum_{k=1}^{K} \mathrm{Var}(s_c \mid k_c^{(d)} = k)}{\mathrm{Var}_{\mathrm{rand}}(s_c)}, \tag{25}$$

where $\mathrm{Var}_{\mathrm{rand}}(s_c)$ is the average within-group variance under random channel assignments that preserve the number of channels in each prototype group.

Finally, to test whether prototypes collapse into domain labels, we define

$$\mathrm{Purity} = \frac{1}{K} \sum_{k=1}^{K} \max_d p(d \mid k), \tag{26}$$

Table 9: **Cross-domain reuse and semantic coherence of learned prototypes.** Higher Cos indicates more similar prototype usage across domains. Lower ρ_s indicates stronger within-prototype coherence relative to a matched random baseline.

Metric	Value
Cos(A-share, NASDAQ-100)	0.87
Purity	0.58
ρ_{period}	0.68
ρ_{vol}	0.74
ρ_{acf1}	0.79

where $p(d \mid k)$ is the fraction of channels assigned to prototype k from domain d.

Table 9 describes prototype usage and statistical coherence. The usage similarity of 0.87 indicates similar aggregate usage across markets, but alone does not rule out uniform assignments. Purity depends on market composition and should be read descriptively. The three variance ratios below one indicate greater within-group coherence than the matched random partitions. Together with the freezing and reinitialization controls, these results support reusable interaction roles, not a one-prototype–one-named-pattern interpretation.

E.5 Expanded Pretraining Study Under Domain Shift

In the main paper, we observe that financial-only pretraining yields limited gains on non-financial benchmarks such as Traffic and Electricity. To better understand this phenomenon, we conduct an additional experiment that expands the pretraining corpus by including the training split of the target dataset itself.

We compare three settings:

- **None**: train from scratch on the target dataset;
- **Financial**: pretrain on the financial corpus and then finetune on the target dataset;
- **Financial + Target**: jointly pretrain on the financial corpus and the training split of the target dataset before finetuning.

All finetuning settings update all model parameters on the target training split. Inputs, horizons, chronological splits, and validation-based model selection are identical; target validation and test data are excluded from gradient-based training. In "Financial + Target", only the target training split also enters joint pretraining. This setting tests the benefit of joint training, not transfer to an unseen target dataset.

Table 10 shows modest, metric-dependent changes from financial-only pretraining: Traffic improves, while Electricity MSE changes from 0.120 to 0.121 and MAE from 0.212 to 0.211. Including target training data in joint pretraining gives clearer gains on both datasets, supporting effective adaptation when target-relevant structure is represented during pretraining.

Table 10: **Expanded pretraining under domain shift.** Every pretrained model is subsequently finetuned on the target training split. Financial-only pretraining has modest, metric-dependent effects; joint pretraining with target training data gives clearer gains.

Pretrain	Target dataset	MSE	MAE
None	Traffic	0.343	0.239
Financial	Traffic	0.341	0.238
Financial + Traffic	Traffic	**0.335**	**0.234**
None	Electricity	0.120	0.212
Financial	Electricity	0.121	0.211
Financial + Electricity	Electricity	**0.119**	**0.205**

E.6 End-to-End Efficiency on High-Dimensional Financial Data

We profile the complete 587-stock A-share panel from Setting I on NVIDIA GeForce RTX 4090 hardware. Training time is measured per target-domain epoch, and each inference request forecasts one complete panel. Relative latency is normalized to Unicorn; relative throughput is its reciprocal under the same panel workload. The models follow the training and finetuning protocols in Appendix D.

Table 11: **End-to-end efficiency on the 587-stock A-share panel.** Lower training time, latency, and peak memory are better; higher throughput is better. Relative latency and throughput use Unicorn as the reference.

Model	Training time (s/epoch)	Relative latency	Relative throughput	Peak memory (GiB)	
				Training	Inference
TimesFM 2.0	520	≈ 30.0	≈ 0.033	23.2	4.2
Kronos-small	260	≈ 15.0	≈ 0.067	17.4	2.3
TimeBridge	31	≈ 1.4	≈ 0.714	15.2	1.2
Unicorn	**22**	**1.0**	**1.000**	**11.4**	**0.9**

Table 11 shows that Unicorn has the lowest training time, inference latency, and peak memory among the compared models on this workload. Absolute inference latency is 9.8 ms per panel for TimeBridge and 7.0 ms for Unicorn, corresponding to a 28.6% reduction. Unicorn also reduces peak inference memory from 1.2 to 0.9 GiB, a 25.0% reduction. These measurements complement the routing complexity analysis in Section 4.5 with end-to-end computational savings on a high-dimensional panel.