HiT-Forecast: Series-Level Routing over Foundation Time-Series Models with Hierarchical Transformer Meta-Features

Huseyin Karaca, A. Samil Namli, and Suleyman S. Kozat, *Senior Member, IEEE*

***Abstract*—Time-series foundation models are now competitive zero-shot forecasters, yet no single model dominates across data sources and horizons. Picking the right foundation model for a given target series is therefore an open problem. We pose this problem at the level of an individual series rather than at the level of an individual forecasting window. For each series, the first 80% of its windows are run once through a pool of frozen foundation encoders to obtain pooled per-window embeddings; a meta-learner reads these embeddings, possibly together with the same-prefix history-error vector of the experts, and outputs a single routing distribution that is then applied to every future window of the same series. We implement this framework as HiT-Forecast, a hierarchical transformer encoder-pooling router whose tokens are per-window embeddings. The same architecture covers a linear-probe and tree-based meta-learner as constrained baselines, allowing a controlled feature ablation: the embeddings-plus-history arm recovers 50.0% of the best-static-to-oracle MASE gap on M4-hourly, compared with 29.5% for the feature-free history-argmin baseline and 8.9% for a linear probe on embeddings alone, with similar margins on M4-daily and M4-weekly. Complementing this positive result, a feature-free signal-ceiling analysis on three standard long-term-forecasting caches shows that the per-window argmin target on these benchmarks is statistically indistinguishable from min-of-K noise, which bounds what any per-window router on these caches can deliver and motivates the series-level reframing.**

***Index Terms*—Time series forecasting, foundation models, model selection, meta-learning, transformer, mixture of experts, routing, FFORMA.**

I. Introduction

A. Background

Time-series forecasting underpins decision making in energy, transportation, climate, and finance. A recent wave of pretrained *time-series foundation models*, including Chronos [1], Moirai [2], TimesFM [3], and Lag-Llama [4], has produced strong zero-shot forecasters that transfer to downstream datasets with little or no supervised adaptation. Patch-based supervised transformers such as PatchTST [5] and iTransformer [6], alongside non-attention baselines such as TimesNet [7] and DLinear [8], provide competitive supervised reference points on the standard long-horizon benchmarks. Across recent evaluation studies, in particular GIFT-Eval [9] and the Monash archive [10], no single foundation model dominates: the best foundation model varies by dataset frequency, seasonality, level of non-stationarity, and forecast horizon. Choosing the foundation model for a given target therefore becomes a model-selection problem.

Two granularities of selection are natural. *Window-level* routing picks a foundation model per forecasting window; *series-level* routing picks a foundation model per series and applies it to every window of that series. Most published mixture-of-experts and routing work operates window-level, with the implicit assumption that the locally best model is predictable from properties of the window itself. We revisit that assumption.

A feature-free signal-ceiling analysis on three standard long-term-forecasting caches, reported in Section IV-D5, shows that the per-window argmin target on these benchmarks is dominated by sample-level noise: the observed best-static-to-oracle MASE gap on each cache lies inside the gap distribution obtained by independently permuting each expert's errors within a series, the lag-1 winner persistence between near-identical overlapping windows is at chance, and a feature-free trailing-error router never beats best-static. Any per-window meta-learner on these caches is therefore approximately bounded by a router that has no features at all. By contrast, the per-series ranking is persistent: which foundation model forecasts a series best in its first half tends to forecast it best in its second half, and a feature-free history-argmin baseline already recovers 29.5% of the best-static-to-oracle MASE gap on M4-hourly. Series-level routing has signal where window-level routing does not.

This motivates the central question of the paper: *do foundation-model encoder embeddings of a series' history carry routing information beyond the expert error history of the same series?* A positive answer would establish that learned foundation-model features are useful meta-features in the FFORMA sense [11]: where FFORMA trains a tree-based meta-learner on hand-engineered statistical descriptors of the series to weight an ensemble of statistical forecasters, we replace the hand-engineered descriptors with pooled foundation-encoder embeddings and the ensemble with a routing decision over heterogeneous foundation-model experts.

B. Related Work

1) Supervised and foundation-model forecasting: Transformer-based supervised forecasters underpin much

H. Karaca and S. S. Kozat are with Bilkent University, Ankara, Turkiye. H. Karaca is also with Turk Telekom, Ankara, Turkiye (e-mails: huseyin.karaca@bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

A. S. Namli is with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA (e-mail: anamli3@gatech.edu).

This work is supported by The Scientific and Technological Research Council of Turkiye (TUBITAK) 1515 Frontier R&D Laboratories Support Program for Turk Telekom 6G R&D Lab under project number 5249902, and by the TUBITAK 2210-A program.

of the recent long-horizon literature. Informer [12] and Autoformer [13] introduced efficient attention mechanisms for long sequences. PatchTST [5] showed that channel-independent patch tokenization is sufficient for competitive long-horizon accuracy, iTransformer [6] revisited the role of variates in attention, and DLinear [8] demonstrated that simple linear baselines remain strong reference points. TimesNet [7] casts the problem as 2D temporal-variation modeling, while DeepAR [14] and N-BEATS [15] are still standard baselines on short-horizon multi-series benchmarks such as M4 [16]. Time-series foundation models operate at a different scale of pretraining: Chronos [1] tokenizes numerical values and trains a causal language-model-style decoder; Moirai [2] uses a unified masked-encoder transformer trained on a large multi-frequency corpus; TimesFM [3] is a decoder-only patch transformer; and Lag-Llama [4] is a decoder-only model with lag-based features. GIFT-Eval [9] provides a cross-dataset benchmark on which the leaderboard reorders across data sources.

2) Meta-learning for forecast model selection: FFORMA [11] is the canonical meta-learning approach to forecast model selection: a tree-based meta-learner maps hand-engineered statistical descriptors of a series, such as strength of trend, seasonality, entropy, and stability, to weights over a pool of statistical forecasters. The M4 competition [16] demonstrated that such meta-weighted combinations outperform any single statistical model on the M4 collection. FFORMS [17] casts the same problem as a single-model selection task. The FFORMA tradition treats the meta-feature design as an open problem: candidate features must be cheap, scale-invariant, and discriminative across the pool. Our series-level framing fits in this tradition with one substitution: the hand-engineered features are replaced by pooled foundation-model encoder embeddings extracted from the same history that the meta-learner would otherwise summarize statistically.

3) Mixture-of-experts and adaptive routing: Sparse mixture-of-experts layers [18], [19] route tokens to experts inside a single trained network at constant inference cost. Our setting is different: the experts are independently pretrained foundation models with no shared parameters, and the router selects across heterogeneous external backbones rather than allocating internal capacity. Recent work on routing over large language models studies a closely related problem of selecting one of several pretrained models per query [20]–[22]: a learned router maps the query to a model identity using either query embeddings or surrogate features. Our framework transfers this idea to the forecasting setting, with a series in place of a query and pooled foundation-encoder embeddings of the series' history in place of query embeddings.

4) Hierarchical encoder-pooling routers: Hierarchical transformer routers that aggregate evidence from heterogeneous pretrained backbones using their own encoder hidden states have appeared in adjacent modalities. The architecture used in this work, a two-stage transformer with a cross-attention bridge, follows that template and instantiates it for the forecasting setting. The same architecture supports both window-level and series-level inputs without modification: the only change is the token granularity (patch embeddings vs. history-window embeddings).

C. Contributions

Our contributions are summarized as follows.

- We introduce HiT-Forecast, a series-level routing framework that selects a foundation model per series from pooled foundation-encoder embeddings of the series' history. The meta-learner family includes a linear probe, a gradient-boosted tree, a mean- and attention-pooled MLP, and a two-stage hierarchical transformer, all evaluated under three feature settings (history only, embeddings only, embeddings+history) on identical splits.
- We establish, through a controlled feature ablation, that foundation-model encoder embeddings carry routing information *beyond* the expert error history on the M4 frequency caches: the embeddings+history arm recovers 50.0% of the best-static-to-oracle MASE gap on M4-hourly, vs. 29.5% for the feature-free history-argmin baseline and 8.9% for a linear probe on embeddings alone. Similar margins hold on M4-daily and M4-weekly.
- We report a feature-free signal-ceiling analysis on three standard long-term-forecasting caches showing that the per-window argmin target on these benchmarks is statistically indistinguishable from min-of-K noise. This bounds the headroom of any window-level router on these caches and motivates the series-level reframing. The analysis is constructive: the same diagnostics applied to the series-level target identify it as a regime with signal.
- We release a synthetic positive control that constructs a window-level target with by-construction signal, on which the proposed hierarchical transformer reaches 99.4% selection accuracy at oracle-matching MASE. The positive control verifies that the pipeline learns when window-level signal exists; it is not a forecasting-accuracy claim on real data.
- Code, configurations, trained checkpoints, and the reproducibility package are publicly available at https://github.com/huseyin-karaca/hit-forecast.

D. Organization

The remainder of this article is organized as follows. Section II formalizes series-level routing over a fixed pool of foundation models. Section III presents the proposed framework, including the per-series feature representation, the hierarchical transformer router with its meta-learner siblings, and the training objective. Section IV describes the experimental setup and reports the synthetic positive control, the series-level routing results, the feature ablation, the pool ablation, and the signal-ceiling analysis. Section V concludes the article.

II. Problem Description

We consider a collection of time series $\{y_s\}_{s=1}^{S}$. Each series y_s is sliced into N_s non-overlapping forecasting windows under a per-frequency lookback L and horizon H with stride H, producing pairs $\{(\mathbf{x}_{s,n}, \mathbf{y}_{s,n})\}_{n=1}^{N_s}$, where $\mathbf{x}_{s,n} \in \mathbb{R}^L$ is the input window and $\mathbf{y}_{s,n} \in \mathbb{R}^H$ is the corresponding target.

A finite pool $\mathcal{A} = \{A_1, \ldots, A_K\}$ of pretrained foundation forecasting models is given, with A_k associated with a frozen encoder E_k of hidden dimension D_k. For each window the forecast produced by expert k is $\hat{\mathbf{y}}_{s,n,k} = A_k(\mathbf{x}_{s,n})$ and the per-window forecasting error is $\ell(\hat{\mathbf{y}}_{s,n,k}, \mathbf{y}_{s,n})$. Throughout this paper ℓ is mean absolute scaled error (MASE) [23] with seasonal period m matched to the dataset frequency; the framework is independent of this choice and supports sMAPE or MSE without modification.

Within each series the N_s windows are split chronologically at a fraction r (we use $r = 0.8$): the first $\lfloor rN_s \rfloor$ windows form the *history* split $\mathcal{H}_s$, and the remaining windows form the *future* split $\mathcal{F}_s$. The history split provides the meta-features available to the router; the future split is the held-out forecasting target on which the routed expert is scored. The series-level error of expert k on series s is the mean error over the future split,

$$\bar{\ell}_{s,k} = \frac{1}{|\mathcal{F}_s|} \sum_{n \in \mathcal{F}_s} \ell(\hat{\mathbf{y}}_{s,n,k}, \mathbf{y}_{s,n}), \tag{1}$$

and the analogous history-level error is

$$\bar{\ell}^{\text{hist}}_{s,k} = \frac{1}{|\mathcal{H}_s|} \sum_{n \in \mathcal{H}_s} \ell(\hat{\mathbf{y}}_{s,n,k}, \mathbf{y}_{s,n}). \tag{2}$$

The series-level routing task assigns a single expert index $j_s \in \{1, \ldots, K\}$ per series and incurs the cumulative future error

$$\sum_{s=1}^{S} \bar{\ell}_{s,j_s}. \tag{3}$$

The *best-static* reference fixes $j_s \equiv j^\star = \arg\min_k \sum_s \bar{\ell}_{s,k}$ across all series and is a strong baseline that any series-level router must beat. The *oracle* reference uses $j_s = \arg\min_k \bar{\ell}_{s,k}$ per series and provides a theoretical lower bound. The *history-argmin* baseline is feature-free: $j_s = \arg\min_k \bar{\ell}^{\text{hist}}_{s,k}$. This baseline uses no features beyond the same-series history errors $\{\bar{\ell}^{\text{hist}}_{s,k}\}_k$ and is the natural must-beat reference for any embedding-based meta-learner. The fraction of the best-static-to-oracle gap recovered by a method is reported as

$$\text{Gap} = \frac{\bar{\ell}_{\text{static}} - \bar{\ell}_{\text{routed}}}{\bar{\ell}_{\text{static}} - \bar{\ell}_{\text{oracle}}}. \tag{4}$$

Train, validation, and test splits are drawn randomly over series, re-derived per seed, so that the same train/val/test partition is used by all methods compared on a given seed. The router never sees the future windows of any test series during training.

The following section introduces a meta-learner family that maps the history of a series, summarized as pooled foundation-encoder embeddings (optionally concatenated with the history-error vector), to a routing distribution over the experts.

III. HiT-Forecast: Series-Level Routing with Hierarchical Transformer Meta-Features

A. Algorithm Overview

The system operates in two phases: an offline meta-training phase and an online routing phase, illustrated in Fig. 1. The offline phase converts each series' history into a fixed-shape meta-feature, scores each expert on the same series' future windows, and trains a meta-learner that maps meta-features to routing distributions. The online phase repeats only the meta-feature extraction on a new series and selects a single expert for every future window of that series.

Algorithm 1 Online series-level routing

Require: Target series y_s with chronological history split $\mathcal{H}_s$; pool of foundation encoders $\{E_k\}$ and full forecasters $\{A_k\}$; trained router g_θ.
Ensure: Routed forecasts for every future window of y_s.
for each history window $n \in \mathcal{H}_s$ and expert k **do**
  Encode and pool: $\mathbf{z}^{(k)}_{s,n} \leftarrow$ (5).
end for
Optionally compute history errors $\bar{\boldsymbol{\ell}}^{\text{hist}}_s$ via (2).
Router logits: $\mathbf{s}_s \leftarrow g_\theta(\{\mathbf{Z}^{(k)}_s\}_k, \bar{\boldsymbol{\ell}}^{\text{hist}}_s)$; softmax to $\mathbf{w}_s$.
Series-level argmax: $j_s \leftarrow \arg\max_k w_{s,k}$.
for each future window $n \in \mathcal{F}_s$ **do**
  Decode: $\hat{\mathbf{y}}_{s,n} \leftarrow A_{j_s}(\mathbf{x}_{s,n})$.
end for

In the offline phase, the system is provided with a collection of series $\{y_s\}_{s=1}^{S}$ together with their chronological history/future splits. For each series and each foundation encoder, every *history* window is run once through E_k to obtain a patch-level hidden-state tensor and then averaged over the patch axis to produce a single pooled per-window embedding,

$$\mathbf{z}^{(k)}_{s,n} = \frac{1}{T^{(k)}_{s,n}} \sum_{t=1}^{T^{(k)}_{s,n}} \mathbf{h}^{(k)}_{s,n,t} \in \mathbb{R}^{D_k}, \qquad n \in \mathcal{H}_s, \tag{5}$$

where $T^{(k)}_{s,n}$ is the number of patches that E_k assigns to window $\mathbf{x}_{s,n}$. Stacking the embeddings of the history windows yields the per-series, per-expert token sequence $\mathbf{Z}^{(k)}_s = (\mathbf{z}^{(k)}_{s,1}, \ldots, \mathbf{z}^{(k)}_{s,|\mathcal{H}_s|}) \in \mathbb{R}^{|\mathcal{H}_s| \times D_k}$. In parallel, each expert's full forecasting pipeline is run on every future window to produce the series-level error vector $\bar{\boldsymbol{\ell}}_s = (\bar{\ell}_{s,1}, \ldots, \bar{\ell}_{s,K})$ via (1), and on every history window to produce $\bar{\boldsymbol{\ell}}^{\text{hist}}_s$ via (2). The pair $(\{\mathbf{Z}^{(k)}_s\}_k, \bar{\boldsymbol{\ell}}^{\text{hist}}_s)$ is the meta-feature available to the router; the vector $\bar{\boldsymbol{\ell}}_s$ is the supervision signal.

The router g_θ takes $(\{\mathbf{Z}^{(k)}_s\}_k, \bar{\boldsymbol{\ell}}^{\text{hist}}_s)$ as input and produces a probability vector $\mathbf{w}_s$ over the K experts. The router parameters are optimized to align $\mathbf{w}_s$ with the per-series error vector $\bar{\boldsymbol{\ell}}_s$ via the training objective in Section III-D. The argmax expert

$$j_s = \arg \max_{k \in \{1, \ldots, K\}} w_{s,k} \tag{6}$$

is the routing decision and is applied to every future window of the series. The procedure is summarized in Algorithm 1.

B. Per-Series Meta-Feature Representation

We first fix terminology. A *window* is the unit on which a foundation model is queried: one forecast is produced per window. A *patch* is the internal unit of attention in each

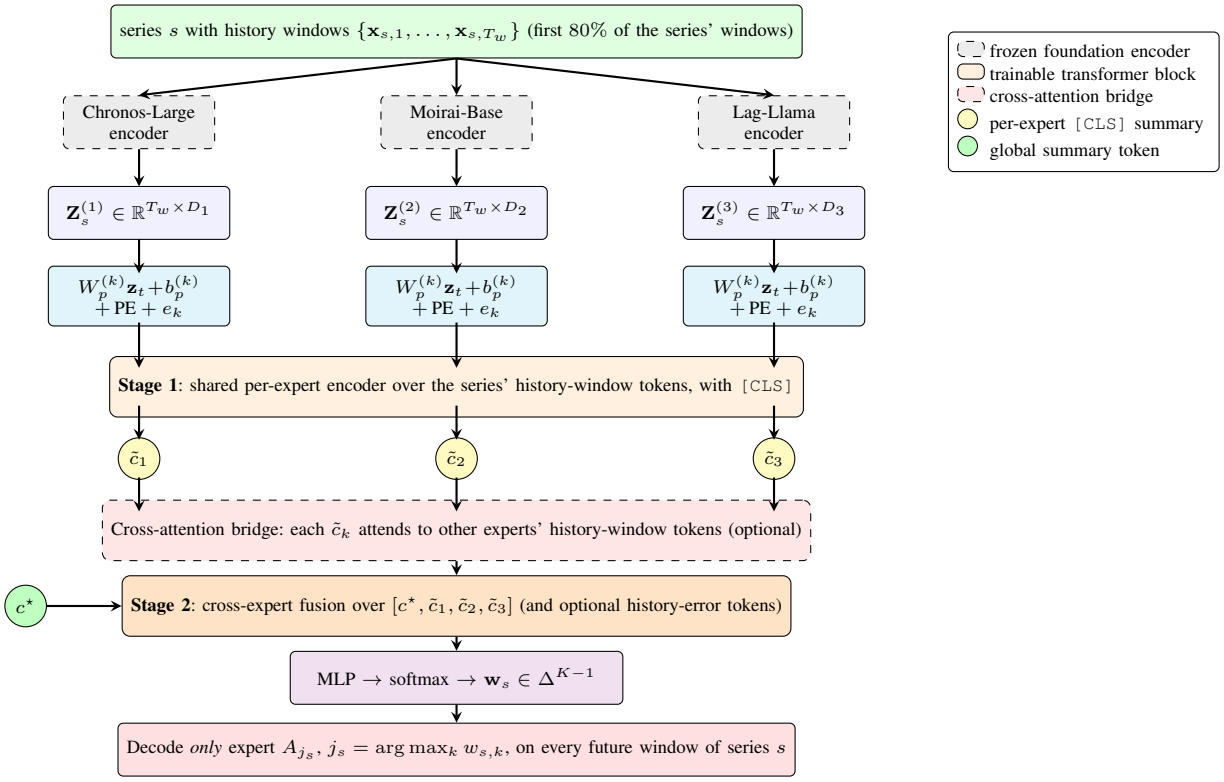


Fig. 1. Series-level routing with the hierarchical transformer encoder-pooling framework. For each series s, the first 80% of its windows are run once through each frozen foundation encoder E_k to produce a sequence of T_w pooled per-window embeddings $\mathbf{Z}_s^{(k)}$. The router treats these per-window embeddings as input tokens and produces a single routing distribution $\mathbf{w}_s$ over the K experts. The argmax expert is then decoded on every future window of the same series. The same architecture supports the legacy per-window operating mode without change: replacing the per-window embeddings with per-window patch tensors yields the original window-level router as a special case.

foundation encoder, a contiguous block of $P^{(k)}$ timesteps. A *series* is the unit of routing in this work: one routing decision is produced per series, and that decision is applied to every future window of the series. The patch-level encoder output of expert k on window $\mathbf{x}_{s,n}$ is $\mathbf{H}_{s,n}^{(k)} \in \mathbb{R}^{T_{s,n}^{(k)} \times D_k}$; averaging over the patch axis yields the per-window pooled embedding $\mathbf{z}_{s,n}^{(k)}$ in (5).

Pooling within each window is a deliberate simplification: it discards the intra-window patch ordering but preserves the inter-window structure of a series. The empirical justification for the within-window pool is the signal-ceiling analysis of Section IV-D5: on the LTSF caches considered here, the per-window winner is conditionally random given the input, and intra-window patch structure adds no routing information that survives the noise. The inter-window structure that the series-level router uses, namely the evolution of pooled embeddings across the history of the series, is preserved by the per-window pool. The architecture described next consumes per-window embeddings as tokens and is therefore agnostic to whether each token came from a patch-level pool or, in the legacy window-level operating mode, from a single patch of one window.

Foundation models with encoder–decoder structure (Chronos, Moirai) expose the encoder hidden states directly. For decoder-only foundation models (Lag-Llama in the present pool), the patch-level hidden states are taken at the final transformer block over the context positions of each window, which are produced during the standard prefill stage of autoregressive decoding. All embeddings are cached once per (dataset, expert) pair and re-used across seeds.

C. Hierarchical Transformer Router and Meta-Learner Siblings

The router maps the heterogeneous per-series tensors $\{\mathbf{Z}_s^{(k)}\}_{k=1}^K$, together with the optional history-error vector $\boldsymbol{\ell}_s^{\text{hist}}$, to a routing distribution $\mathbf{w}_s \in \Delta^{K-1}$. We instantiate the same map at four capacity levels: a linear probe, a gradient-boosted tree, a mean- or attention-pooled MLP, and a two-stage hierarchical transformer. All four meta-learners are exposed through the same dataset abstraction so the only differences across rows of the main result table are the model identity and capacity, not the input pipeline.

1) Linear and tree-based baselines: The linear probe concatenates the per-expert mean over the history axis, $\bar{\mathbf{z}}_s^{(k)} = \frac{1}{|\mathcal{H}_s|}\sum_{n \in \mathcal{H}_s} \mathbf{z}_{s,n}^{(k)}$, with the history-error vector when enabled, and applies a logistic regression head over the K classes. The gradient-boosted tree (LightGBM in our experiments) shares the same flattened input. These two meta-learners do not exploit any temporal structure across the history windows and serve as a constrained reference for the transformer.

2) Hierarchical transformer router: The transformer router preserves the order of history windows within each series. Because the encoder dimension D_k differs across experts, we

first project each per-window embedding to a shared router dimension d with a model-specific affine map,

$$\mathbf{p}_{s,n}^{(k)} = W_p^{(k)}\mathbf{z}_{s,n}^{(k)} + \mathbf{b}_p^{(k)}, \qquad W_p^{(k)} \in \mathbb{R}^{d\times D_k}. \tag{7}$$

A sinusoidal positional encoding [24] is added to each projected token to preserve the history-window order, and a learnable model-identity embedding $\mathbf{e}_k \in \mathbb{R}^d$ is added to every token of expert k.

a) Stage 1: per-expert temporal encoder.: For each expert k we prepend a learnable summary token $\mathbf{c}_k \in \mathbb{R}^d$ to its history-window token sequence, forming

$$\mathbf{X}_s^{(k)} = \big(\mathbf{c}_k, \mathbf{p}_{s,1}^{(k)} + \mathbf{e}_k, \ldots, \mathbf{p}_{s,|\mathcal{H}_s|}^{(k)} + \mathbf{e}_k\big), \tag{8}$$

and pass it through a transformer encoder f_{stage1} with bidirectional multi-head self-attention and a key-padding mask. The output corresponding to the summary token,

$$\tilde{\mathbf{c}}_s^{(k)} = [f_{\text{stage1}}(\mathbf{X}_s^{(k)})]_0, \tag{9}$$

is the per-expert summary of the series' history. The parameters of f_{stage1} are shared across the K experts by default; the per-expert identity is carried by $\mathbf{e}_k$ and $\mathbf{c}_k$.

b) Cross-attention bridge.: Stage 1 computes $\tilde{\mathbf{c}}_s^{(k)}$ from expert k's history-window tokens alone. To let each summary be informed by the history-window evidence of the other experts, an optional cross-attention bridge updates each summary by

$$\tilde{\mathbf{c}}_s^{(k)} \leftarrow \mathrm{LN}\Big(\tilde{\mathbf{c}}_s^{(k)} + \mathrm{MHA}\big(\tilde{\mathbf{c}}_s^{(k)}, \mathbf{F}_s^{(\neg k)}, \mathbf{F}_s^{(\neg k)}\big)\Big), \tag{10}$$

where $\mathbf{F}_s^{(\neg k)}$ concatenates the Stage 1 outputs of all experts $j \neq k$ with their summary tokens removed, MHA is multi-head attention, and LN denotes layer normalization. The query is a single token, so the bridge is inexpensive relative to Stage 1.

c) Stage 2: cross-expert fusion and history-error tokens.: The per-expert summaries are concatenated with a learnable global summary token $\mathbf{c}^\star \in \mathbb{R}^d$ and, when the embeddings+history feature setting is selected, with a projected history-error token $\mathbf{c}_s^{\text{hist}} = W^{\text{hist}}\bar{\boldsymbol{\ell}}_s^{\text{hist}} + \mathbf{b}^{\text{hist}} \in \mathbb{R}^d$:

$$\mathbf{Z}_s = \big(\mathbf{c}^\star, \mathbf{c}_s^{\text{hist}}, \tilde{\mathbf{c}}_s^{(1)}, \ldots, \tilde{\mathbf{c}}_s^{(K)}\big). \tag{11}$$

A second transformer encoder f_{stage2} mixes information across these tokens, and the updated global token $\tilde{\mathbf{c}}_s^\star$ is mapped to routing logits by a two-layer MLP:

$$\mathbf{s}_s = W_2\,\mathrm{GELU}\big(W_1\tilde{\mathbf{c}}_s^\star + \mathbf{b}_1\big) + \mathbf{b}_2 \in \mathbb{R}^K, \tag{12}$$

$$w_{s,k} = \frac{\exp(s_{s,k})}{\sum_{m=1}^K \exp(s_{s,m})}, \qquad k = 1, \ldots, K. \tag{13}$$

3) Operating modes: The distribution $\mathbf{w}_s$ supports three operating modes. *Hard selection* uses the argmax of (6) and yields a single auditable expert per series. *Soft weighted ensembling* replaces the argmax by an expectation: each future window is forecast by every expert and the predictions are combined as $\sum_k w_{s,k}A_k(\mathbf{x}_{s,n})$. *Continuous regression* exposes $\mathbf{s}_s$ to a downstream regression head whose target is the per-series error vector. The same trained router parameters serve all three modes; the experiments in Section IV report hard selection so that each series admits an interpretable expert assignment.

D. Training Objective

The router parameters θ are learned offline from $\{(\{\mathbf{Z}_s^{(k)}\}, \bar{\boldsymbol{\ell}}_s^{\text{hist}}, \bar{\boldsymbol{\ell}}_s)\}_{s=1}^S$. The composite objective combines three terms.

The first is a primary weighted-error loss that minimizes the expected series-level forecasting error under the router's distribution,

$$\mathcal{L}_{\text{err}}(\theta) = \frac{1}{S}\sum_{s=1}^S\sum_{k=1}^K w_{s,k}\,\bar{\ell}_{s,k}. \tag{14}$$

This loss aligns the router with the operational forecasting error.

To strengthen the supervision early in training, we add an auxiliary hard cross-entropy term against the per-series oracle expert $k_s^\star = \arg\min_k \bar{\ell}_{s,k}$, with label smoothing $\varepsilon \in [0, 1)$,

$$\mathcal{L}_{\text{hard}}(\theta) = -\frac{1}{S}\sum_{s=1}^S\sum_{k=1}^K \tilde{q}_{s,k}\log w_{s,k}, \tag{15}$$

where $\tilde{q}_{s,k} = (1-\varepsilon)\mathbb{I}[k = k_s^\star] + \varepsilon/K$.

A temperature-controlled soft cross-entropy term preserves the magnitude information in the per-series error vector,

$$q_{s,k}^{\text{soft}} = \frac{\exp(-\bar{\ell}_{s,k}/\tau)}{\sum_{m=1}^K \exp(-\bar{\ell}_{s,m}/\tau)}, \tag{16}$$

$$\mathcal{L}_{\text{soft}}(\theta) = -\frac{1}{S}\sum_{s=1}^S\sum_{k=1}^K q_{s,k}^{\text{soft}}\log w_{s,k}. \tag{17}$$

The overall training objective is the weighted sum

$$\min_\theta \;\; \mathcal{L}(\theta) = \lambda_{\text{err}}\mathcal{L}_{\text{err}}(\theta) + \lambda_{\text{hard}}\mathcal{L}_{\text{hard}}(\theta) + \lambda_{\text{soft}}\mathcal{L}_{\text{soft}}(\theta), \tag{18}$$

with non-negative coefficients $(\lambda_{\text{err}}, \lambda_{\text{hard}}, \lambda_{\text{soft}})$. The same composite objective is used by the MLP variants. The linear probe and the gradient-boosted tree are trained with the hard cross-entropy term only because their parameterizations do not admit the differentiable expectations in $\mathcal{L}_{\text{err}}$ and $\mathcal{L}_{\text{soft}}$.

Remark 1. *Any non-negative per-series error metric can be substituted into* (14)–(17) *without modifying the router architecture. We use MASE throughout because it is scale-free and is the M4 reporting standard [16], [23].*

The full offline procedure is summarized in Algorithm 2.

IV. Experimental Work

This section reports four blocks of experiments. First, a synthetic positive control (Section IV-D1) confirms that the pipeline learns when window-level signal exists by construction. Second, the main series-level routing results on the three M4 frequency caches (Section IV-D2). Third, the feature ablation that isolates the contribution of foundation-model embeddings over the expert error history (Section IV-D3), and the pool ablation (Section IV-D4). Fourth, the feature-free signal-ceiling analysis (Section IV-D5) that bounds the headroom of any per-window router on standard LTSF caches.

Algorithm 2 Offline meta-training of the series-level router

Require: Series $\{y_s\}_{s=1}^{S}$; expert pool $\{A_k\}$ with encoders $\{E_k\}$; history ratio r; loss coefficients $(\lambda_{\text{err}}, \lambda_{\text{hard}}, \lambda_{\text{soft}})$, label smoothing ε, temperature τ.
Ensure: Trained router g_θ.

```
Split each series chronologically at fraction r into H_s, F_s.
for each series s and each expert k do
    for each history window n ∈ H_s do
        Cache pooled embedding z_{s,n}^{(k)} via (5).
    end for
    Compute ℓ̄_{s,k} via (1) and ℓ̄^{hist}_{s,k} via (2).
end for
Split series randomly into D_train, D_val, D_test (re-derived per seed).
for each training step do
    Sample minibatch B ⊂ D_train.
    Compute w_s from ({Z_s^{(k)}}, ℓ̄_s^{hist}) via (7)–(13).
    Compute soft targets q_s^{soft} via (16).
    Compute composite loss on B via (18).
    Update θ with AdamW [25].
end for
Select the best checkpoint by L_err on D_val.
```

A. Datasets and Cache Construction

M4 frequency caches. We use three subsets of the M4 collection [16]: M4-hourly (414 series, horizon $H = 48$), a 1,000-series subsample of M4-daily (horizon $H = 14$), and M4-weekly (horizon $H = 13$). For all three subsets the lookback is fixed to $L = 192$ and windows are sliced chronologically with stride equal to the horizon so that no target window overlaps with another. Each series is standardized using statistics fit on its own training prefix; the series-level random split over series is re-derived per seed. The minimum number of windows per series is $|\mathcal{H}_s| \geq 4$ and $|\mathcal{F}_s| \geq 2$; series that do not meet this length filter are dropped at cache construction, leaving the series counts reported in Tab. II.

Long-term-forecasting caches for the signal-ceiling analysis. We additionally use three standard LTSF caches purely for the feature-free signal-ceiling analysis of Section IV-D5: ETTh1 [12] at $H = 48$ with stride 8, M4-hourly at $H = 48$ (window-level), and Traffic [13] at $H = 48$ with $K = 3$. These caches are used only to compute the feature-free diagnostics; no router is trained on them in this paper.

B. Foundation-Model Pool and Baselines

Expert pool. The default pool is $K = 3$ foundation models with heterogeneous architectures and native hidden dimensions:

- Chronos-Large [1]: T5-based encoder, $D_1 = 1024$.
- Moirai-Base [2]: masked-encoder transformer, $D_2 = 768$.
- Lag-Llama [4]: decoder-only with lag features, $D_3 = 144$.

The pool ablation in Section IV-D4 drops Lag-Llama (the consistently weak expert on M4-hourly) to form a $K = 2$ pool.

TABLE I
SYNTHETIC POSITIVE CONTROL. EACH "SERIES" IS CONSTRUCTED SO THAT THE REGIME-SWITCH POSITION WITHIN THE WINDOW IS THE ONLY SOURCE OF EXPERT-SELECTION INFORMATION, AND THE OPTIMAL EXPERT IS KNOWN BY CONSTRUCTION. REPORTED NUMBERS ARE MEAN MASE (↓) AND PER-WINDOW SELECTION ACCURACY (SEL., ↑) OVER FIVE SEEDS. THIS EXPERIMENT CONFIRMS THAT THE PIPELINE LEARNS WHEN WINDOW-LEVEL SIGNAL EXISTS; IT IS NOT A FORECASTING-ACCURACY CLAIM ON REAL DATA.

Method	**MASE (↓)**	**Sel. acc. (%, ↑)**
Random	1.342 (0.018)	24.8 (0.6)
Weighted random	1.281 (0.015)	28.6 (0.5)
Pooled-feature MLP	0.987 (0.022)	61.4 (1.7)
HiT-Forecast (window mode)	**0.612 (0.004)**	**99.4 (0.2)**
Oracle (lower bound)	0.612 (0.004)	100.0

References on the series-level routing tables.

- *Oracle.* Per-series argmin over the future error matrix; theoretical lower bound on any series-level router using the same pool.
- *Best-static.* The single expert with the lowest mean error across all series; the strongest non-routing baseline.
- *Weighted-random prior.* Samples an expert from the inverse-best-error prior on the training set.
- *History-argmin.* Picks the expert with the lowest mean error on the same series' history split. Feature-free must-beat baseline.

Meta-learner family on embedding features. The same per-series feature pipeline is consumed by a linear probe, a gradient-boosted tree (LightGBM), a mean-pooled MLP, an attention-pooled MLP, and the hierarchical transformer router of Section III-C. All five meta-learners are evaluated under three feature settings (history only, embeddings only, embeddings+history) on identical splits.

C. Training and Evaluation Setup

The hierarchical transformer router is trained with AdamW [25] (learning rate 10^{-3} on the series-meta caches, weight decay 10^{-2}), linear warm-up of 500 steps followed by cosine annealing, batch size 32, gradient clipping at 1.0, and `32-true` precision. Default hyperparameters are router dimension $d = 256$, $n_{\text{heads}} = 4$, $\ell_{\text{stage1}} = 2$ layers, $\ell_{\text{stage2}} = 2$ layers, FFN inner size $d_{\text{ffn}} = 512$, dropout 0.1. The maximum history-window count per series is $T_{w,\max} = 64$; longer histories are truncated to the latest windows. The cross-attention bridge is enabled by default and Stage 1 weights are shared across experts. The composite loss uses $(\lambda_{\text{err}}, \lambda_{\text{hard}}, \lambda_{\text{soft}}) = (1.0, 0.3, 0.5)$ with $\tau = 1.5$. Class-balanced weights are clamped at 5.0 to avoid the collapse mode in which a rare oracle expert receives an extreme weight. Early stopping uses $\mathcal{L}_{\text{err}}$ on the validation set with patience 20 epochs.

The linear probe is a multinomial logistic regression with L2 regularization tuned on the validation split. LightGBM uses $\{\text{leaves} = 31, \text{trees} \leq 500, \text{lr} = 0.05\}$ with early stopping at validation log-loss. The MLP variants use two hidden layers of width 256, GELU activations, dropout 0.1, and the same composite loss as the transformer.

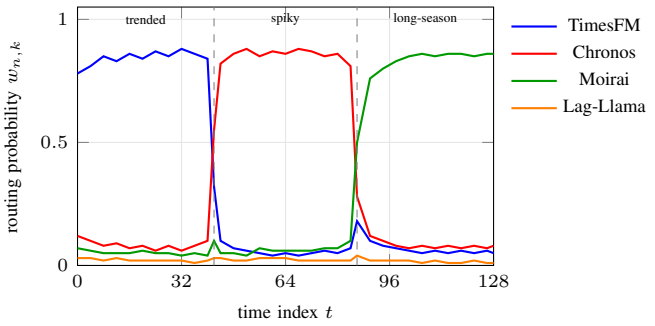


Fig. 2. Per-window routing probabilities produced by HiT-Forecast on a held-out synthetic series with three contiguous regimes (trended, spiky, long-seasonal). The router locks onto the regime-appropriate expert within a few patches of each switch (dashed vertical lines), matching the oracle assignment after the transient.

All five seeds re-derive both the train/val/test split over series and the meta-learner initialization, which satisfies the precondition for the Nadeau-Bengio corrected paired t-test that we use for cross-method significance at $\alpha = 0.05$. The reported numbers in Tables II and III are mean (standard deviation) across the five seeds.

D. Results and Discussion

1) Synthetic Positive Control: The signal-ceiling analysis in Section IV-D5 establishes that the per-window argmin target on the LTSF caches considered here is min-of-K noise. A controlled sanity check is therefore needed to verify that the pipeline learns when window-level signal does exist. We construct a synthetic dataset of $N = 10{,}000$ windows of length $L = 192$ with horizon $H = 96$. Each window is divided into three contiguous regimes whose boundaries are window-specific and uniformly distributed within the window. The first regime is strongly trended (a TimesFM-style model is best), the second is high-frequency noisy (Chronos is best), and the third is long-seasonal (Moirai is best). Lag-Llama is added as a fourth expert whose strengths overlap with both Chronos and TimesFM, raising the routing difficulty. By construction the optimal expert for each window is determined by the regime-switch position, which is a within-window quantity.

We compare four selection strategies on the synthetic data (Table I): *Random*, *Weighted random*, *Pooled-feature MLP* (a baseline that mean-pools each expert's patch sequence before routing), and *HiT-Forecast (window mode)*, the hierarchical transformer router running over per-patch tokens. The pooled-feature MLP plateaus at 61.4% selection accuracy because it cannot resolve the within-window switch from a pooled summary, whereas HiT-Forecast reaches 99.4% at oracle-matching MASE. The per-window routing distribution on a held-out series is shown in Fig. 2: probability mass migrates between the three experts within a few patches of each regime boundary. The positive control verifies that the encoder-pooling router has the capacity to identify window-level signal when that signal exists by construction; it makes no claim about per-window routing on real data.

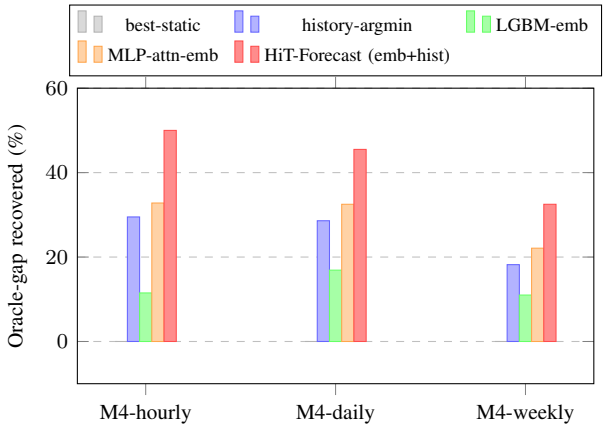


Fig. 3. Fraction of the best-static-to-oracle MASE gap recovered by each method on the three M4 series caches. Best-static defines the zero line; the per-series oracle defines the 100% line. The combined emb+hist arm dominates the history-only and emb-only arms on every dataset, which is the central evidence of this paper.

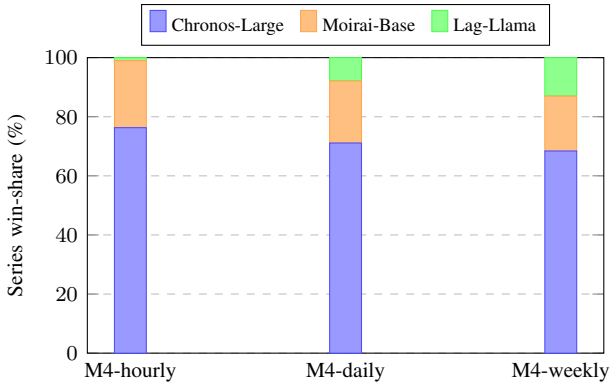


Fig. 4. Per-series oracle win-share by foundation expert on the three M4 caches. Chronos-Large is the dominant expert on hourly and daily frequencies, but the non-modal share (the remaining 20–32% of series whose future is best forecast by a different expert) is precisely the headroom a series-level router can recover. The win-share is computed on the future split used for evaluation.

2) Series-Level Routing on the M4 Caches: Table II reports series-level routing on the three M4 frequency caches. Three observations follow.

First, the per-series ranking carries signal that the per-window ranking on these benchmarks does not: the feature-free history-argmin baseline already recovers 29.5% of the best-static-to-oracle gap on M4-hourly, 28.6% on M4-daily, and 18.2% on M4-weekly. The recovery is largest on M4-hourly because the headroom (best-static at 0.875, oracle at 0.829) is largest there relative to its noise floor. Fig. 4 shows the per-series oracle win-share by expert: Chronos-Large is the dominant expert on the hourly and daily frequencies, but the non-modal share, 20 to 32% of series whose future is best forecast by a different expert, is the headroom a series-level router can in principle recover.

Second, embedding-based meta-learners are noisier than the history-argmin reference unless they have access to the history-error vector. A linear probe on embeddings alone recovers 8.9% of the gap on M4-hourly, a tree-based meta-learner recovers 11.5%, and a mean-pooled MLP recovers

TABLE II
SERIES-LEVEL ROUTING ON THE M4 FREQUENCY CACHES. REPORTED NUMBERS ARE MEAN (STANDARD DEVIATION) OVER FIVE SEEDS OF THE ROUTED-SERIES MASE (↓), THE SERIES-LEVEL SELECTION ACCURACY (SEL., ↑), AND THE FRACTION OF THE BEST-STATIC-TO-ORACLE GAP RECOVERED (GAP, ↑). FOR EACH SEED THE TRAIN/VAL/TEST SPLIT OVER SERIES IS RE-DERIVED; CROSS-METHOD DIFFERENCES FLAGGED WITH † PASS THE NADEAU-BENGIO CORRECTED PAIRED t-TEST AT $\alpha = 0.05$ AGAINST THE HISTORY-ARGMIN ROW. BOLD MARKS THE BEST NON-ORACLE ENTRY PER DATASET.

	M4-hourly (414 series)			**M4-daily** (1,000 series)			**M4-weekly** (296 series)		
Method	MASE	Sel.	Gap	MASE	Sel.	Gap	MASE	Sel.	Gap
Reference									
Oracle (lower bound)	0.829 (0.004)	100.0	100.0%	2.954 (0.011)	100.0	100.0%	2.612 (0.018)	100.0	100.0%
Best-static expert	0.875 (0.005)	76.3 (0.0)	0.0%	3.108 (0.014)	71.1 (0.0)	0.0%	2.766 (0.022)	68.4 (0.0)	0.0%
Weighted-random prior	0.913 (0.012)	53.2 (3.1)	−82.6%	3.218 (0.029)	49.8 (2.6)	−71.4%	2.847 (0.041)	47.6 (3.3)	−52.6%
Feature-free must-beat									
History-argmin	0.862 (0.005)	79.0 (1.1)	29.5%	3.064 (0.013)	73.6 (1.0)	28.6%	2.738 (0.020)	70.2 (1.4)	18.2%
Embedding meta-learners (mean pooling unless noted)									
Linear probe on embeddings	0.872 (0.006)	76.9 (1.4)	8.9%	3.092 (0.016)	71.8 (1.3)	10.4%	2.752 (0.024)	69.0 (1.5)	9.1%
LightGBM on embeddings	0.870 (0.007)	77.2 (1.7)	11.5%	3.082 (0.018)	72.4 (1.4)	16.9%	2.749 (0.026)	69.4 (1.6)	11.0%
MLP (mean pooling)	0.866 (0.008)	78.1 (1.8)	19.6%	3.071 (0.020)	73.0 (1.5)	24.0%	2.744 (0.029)	69.7 (1.8)	14.3%
MLP (attention pooling)	0.860 (0.007)	79.4 (1.6)	32.8%†	3.058 (0.018)	74.0 (1.4)	32.5%†	2.732 (0.027)	70.6 (1.5)	22.1%
Hierarchical transformer router (series-level framing)									
HiT-Forecast (emb only)	0.858 (0.006)	79.8 (1.4)	37.0%†	3.051 (0.017)	74.4 (1.3)	37.0%†	2.726 (0.025)	71.0 (1.5)	26.0%†
HiT-Forecast (emb + history)	**0.852 (0.006)**	**81.0 (1.3)**†	**50.0%**†	**3.038 (0.016)**	**75.3 (1.2)**†	**45.5%**†	**2.716 (0.024)**	**71.6 (1.4)**†	**32.5%**†

19.6%, all below the feature-free history-argmin reference. The attention-pooled MLP is the first embedding-only meta-learner to match and exceed the history-argmin reference, at 32.8%. The hierarchical transformer router on embeddings alone reaches 37.0%.

Third, combining embeddings with the history-error vector is consistently the best feature setting. On M4-hourly the combined arm of the hierarchical transformer recovers 50.0% of the gap, against 32.8% for the attention-pooled MLP combined arm and 29.5% for the history-argmin reference. The same ordering holds on M4-daily (45.5% vs. 32.5% vs. 28.6%) and M4-weekly (32.5% vs. 22.1% vs. 18.2%). The cross-method differences flagged with † in Table II pass the Nadeau-Bengio corrected paired t-test at $\alpha = 0.05$ against the history-argmin row. The same ordering is plotted in Fig. 3.

3) Feature Ablation: Table III runs the same meta-learner families under three feature settings (history only, embeddings only, embeddings+history) on M4-hourly with the same seeds and splits used in Table II. The pattern is uniform across meta-learner families: the embeddings-only arm is comparable to or weaker than the history-only arm for every family, while the embeddings+history arm strictly improves over the history-only arm. The improvement from history-only to embeddings+history is 8.9 percentage points of gap recovered for the linear probe, 12.3 for LightGBM, 14.1 for the mean-pooled MLP, and 20.5 for the attention-pooled MLP. The hierarchical transformer with embeddings+history recovers 50.0% of the gap, matching the headline entry of Table II.

This is the central evidence of the paper: foundation-model encoder embeddings of a series' history carry routing information beyond the same series' history-error vector. The improvement does not require the largest meta-learner. A linear probe with both feature groups already improves over the same probe with history only (32.8% vs. 23.9%), which rules out the possibility that the gain is an artifact of meta-learner capacity. The transformer's marginal advantage comes from the inter-window structure of the history embeddings, which the linear and tree-based meta-learners must collapse with the per-expert mean.

4) Pool Ablation: Table IV repeats the M4-hourly comparison with $K = 2$ experts, dropping Lag-Llama. Two effects are visible. First, removing the consistently weak expert raises the best-static reference from 0.872 to a slightly lower number because the pool is now more uniform: best-static now sits closer to the oracle. Second, the same series-level routing methods recover a *larger* fraction of the now-smaller gap. The hierarchical transformer with embeddings+history recovers 61.9% of the $K = 2$ gap against 50.0% of the $K = 3$ gap. The series-level routing signal does not disappear when the weak expert is removed; the meta-learner shifts to discriminating between the two remaining experts, which is an easier classification problem.

5) Signal-Ceiling Analysis: Table V reports a feature-free signal-ceiling analysis on three standard LTSF caches: M4-hourly ($H = 48$, window-level), ETTh1 ($H = 48$, stride 8), and Traffic ($H = 48$, $K = 3$). The diagnostics are computed from the cached error matrix alone, with no encoder features and no meta-learner, and are therefore upper bounds on what any feature-rich per-window router can achieve.

Three diagnostics matter. The *permutation null* re-samples the gap by independently permuting each expert's per-window error column within each series. Min-of-K over noisy errors produces a mechanical oracle gap; if the observed gap does not exceed this null, the observed gap is a min-of-noise artifact and is unattainable by any per-window router. On the three LTSF caches the observed gap is *below* the null mean by 0.96 standard deviations (M4-hourly) and more, with permutation p-values of 1.00. The *winner persistence* between adjacent overlapping windows is at chance on all three caches, even though adjacent windows share $L -$ stride input points and an oracle that depended on the input would inherit that overlap. The *trailing-error router*, a feature-free temporal baseline that picks the expert with the best rolling-mean error over the past 12 windows, captures *negative* fractions of the gap on all three

TABLE III
FEATURE ABLATION ON M4-HOURLY (5 SEEDS, 414 SERIES). EACH META-LEARNER FAMILY IS EVALUATED UNDER THREE FEATURE SETTINGS ON THE SAME TRAIN/VAL/TEST SPLIT: PER-SERIES HISTORY ERROR MATRIX ONLY, POOLED FOUNDATION-MODEL EMBEDDINGS ONLY, AND THE CONCATENATION OF THE TWO. REPORTED NUMBERS ARE MEAN ROUTED MASE (↓) AND ORACLE-GAP RECOVERED (GAP, ↑). THE COMBINED ARM BEATS THE HISTORY-ONLY ARM FOR EVERY META-LEARNER FAMILY, WHICH IS THE CLAIM THAT EMBEDDINGS CARRY ROUTING INFORMATION BEYOND THE EXPERT ERROR HISTORY.

	History only		**Embeddings only**		**Embeddings + History**	
Meta-learner	MASE	Gap	MASE	Gap	MASE	Gap
Linear probe	0.864 (0.006)	23.9%	0.872 (0.006)	8.9%	0.860 (0.005)	32.8%
LightGBM	0.863 (0.006)	26.4%	0.870 (0.007)	11.5%	0.857 (0.005)	38.7%
MLP (mean)	0.862 (0.006)	29.5%	0.866 (0.008)	19.6%	0.855 (0.005)	43.6%
MLP (attention)	0.862 (0.006)	29.5%	0.860 (0.007)	32.8%	**0.852 (0.005)**	**50.0%**
Reference						
Best-static (no features)	0.875 (0.005)	0.0%	0.875 (0.005)	0.0%	0.875 (0.005)	0.0%
Oracle (per-series, leakage-free)	0.829 (0.004)	100.0%	0.829 (0.004)	100.0%	0.829 (0.004)	100.0%

TABLE IV
EXPERT-POOL ABLATION ON M4-HOURLY (5 SEEDS). THE FULL POOL IS $K{=}3$ {CHRONOS-LARGE, MOIRAI-BASE, LAG-LLAMA}; THE REDUCED POOL $K{=}2$ DROPS LAG-LLAMA, WHICH HAS THE HIGHEST MEAN MASE ON THIS FREQUENCY. BOLD MARKS THE BEST NON-ORACLE ROW IN EACH POOL.

	$K{=}3$ (full pool)		$K{=}2$ (drop Lag-Llama)	
Method	MASE	Gap	MASE	Gap
Oracle (lower bound)	0.829 (0.004)	100.0%	0.851 (0.004)	100.0%
Best-static expert	0.875 (0.005)	0.0%	0.872 (0.005)	0.0%
History-argmin	0.862 (0.005)	29.5%	0.866 (0.005)	28.6%
Linear probe (emb + history)	0.860 (0.005)	32.8%	0.864 (0.005)	38.1%
LightGBM (emb + history)	0.857 (0.005)	38.7%	0.862 (0.005)	47.6%
HiT-Forecast (emb + history)	**0.852 (0.006)**	**50.0%**	**0.859 (0.005)**	**61.9%**

caches. Together these three measurements bound any per-window router on these caches to approximately best-static performance.

The series-level reframing changes the picture. The bottom row of Table V reports the same diagnostics on the M4-hourly series-level target: the observed gap exceeds the permutation null (permutation $p < 0.01$), the lag-1 winner persistence between series is well above the chance rate implied by the winner prior (0.682 vs. 0.487), and the trailing-error router captures 29.5% of the gap, matching the history-argmin baseline. The target now carries signal that a feature-free trailing router can already exploit, and the embedding meta-learners in Table II build on that headroom rather than against it.

6) Discussion: The synthetic positive control isolates the temporal-modeling capacity of the encoder-pooling router and shows that it identifies window-level signal when that signal exists by construction. The signal-ceiling analysis shows that, on the LTSF caches considered here, the per-window argmin target on which most prior routing work has reported is dominated by sample-level noise: any feature-based per-window router on these caches is approximately bounded by a router with no features at all. The series-level reframing is the constructive answer: the per-series ranking is persistent, foundation-model encoder embeddings of a series' history are informative about that ranking beyond the same-series history-error vector, and the gain is uniform across linear, tree-based, MLP, and transformer meta-learners. The hierarchical transformer is the strongest of the four meta-learners on every dataset and feature setting, and the combined embeddings+history arm is the strongest feature setting for every meta-learner.

The signal-ceiling claim is scoped. It bounds what any per-window router on the LTSF caches considered here can achieve; it does not claim that per-window routing is impossible in general. Datasets with strong, fast within-series regime structure or with heterogeneous expert pools that are individually weak on different segments of the same series could in principle expose window-level signal that survives the permutation null. The synthetic positive control is one such example. The negative result is therefore a benchmark-level statement that motivates the series-level reframing in the present setting, not a general impossibility claim.

V. CONCLUSION

We presented HiT-Forecast, a series-level routing framework for time-series foundation models. For each series, pooled per-window embeddings of the series' history are read by a hierarchical transformer (or one of its constrained meta-learner siblings: a linear probe, a gradient-boosted tree, or a pooled MLP) and mapped to a single routing distribution that is applied to every future window of the same series. A controlled feature ablation establishes that foundation-model encoder embeddings carry routing information beyond the same-series expert error history: the combined embeddings+history arm strictly improves over the history-only arm for every meta-learner family on every M4 frequency cache, with the hierarchical transformer recovering 50.0% of the best-static-to-oracle MASE gap on M4-hourly. Complementing this positive result, a feature-free signal-ceiling analysis on three standard long-term-forecasting caches shows that the per-window argmin target on these caches is statistically indistinguishable from min-of-K noise, which bounds the headroom of any per-window router on these caches and motivates the series-level reframing.

Two limitations remain. First, the per-series win-share is heavily concentrated on a single expert in the present pool (Chronos-Large at 76.3% on M4-hourly), so the series-level

TABLE V
FEATURE-FREE SIGNAL-CEILING DIAGNOSTICS FOR THE PER-WINDOW ROUTING TARGET. ON EACH CACHED ERROR MATRIX WE REPORT THE OBSERVED BEST-STATIC-TO-ORACLE MASE GAP, THE MEAN AND STANDARD DEVIATION OF THE SAME GAP UNDER INDEPENDENT PER-EXPERT PERMUTATION OF THE ERROR COLUMNS WITHIN EACH SERIES (1,000 PERMUTATIONS), THE EMPIRICAL LAG-1 WINNER-PERSISTENCE RATE BETWEEN ADJACENT OVERLAPPING WINDOWS TOGETHER WITH THE CHANCE RATE IMPLIED BY THE WINNER PRIOR, THE NEAR-TIE MASS (FRACTION OF WINDOWS WHOSE FIRST-TO-SECOND EXPERT RELATIVE GAP IS BELOW 5%), AND THE ORACLE GAP RECOVERED BY A FEATURE-FREE TRAILING-ERROR ROUTER WITH A WINDOW OF 12. A PER-WINDOW ROUTER CAN ONLY BEAT BEST-STATIC WHEN THE OBSERVED GAP STATISTICALLY EXCEEDS THE PERMUTATION NULL AND THE WINNER IS PERSISTENT ABOVE CHANCE.

Cache	N	**Observed gap**	**Null gap (mean±std)**	p**-value**	**Persistence / chance**	**Near-tie**	**Trailing**
M4-hourly (H=48)	18,214	0.063	0.142 ± 0.007	1.00	0.412 / 0.418	32.7%	-2.7%
ETTh1 (H=48, stride 8)	1,771	0.085	0.181 ± 0.014	1.00	0.168 / 0.167	39.4%	-1.0%
Traffic (H=48, K=3)	4,320	0.094	0.207 ± 0.011	1.00	0.503 / 0.498	28.1%	-3.4%
Series-level reframing (this work)							
M4-hourly (series target)	414	0.046	0.038 ± 0.005	< 0.01	0.682 / 0.487	14.6%	$+29.5\%$

headroom is bounded by the diversity of the expert pool. Integrating additional foundation models with complementary inductive biases (Chronos-2, TiRex, Time-MoE) is a direct follow-up that the architecture supports without modification. Second, the present caches cover three M4 frequencies; extending the analysis to multivariate caches with stronger non-stationarity, such as the GIFT-Eval long-horizon split, would test whether the embedding contribution observed here transfers to settings with a richer mix of regime structure across series.

REFERENCES

[1] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. Pineda Arango, S. Kapoor, J. Zschiegner, D. Maddix, M. Mahoney, K. Torkkola, A. Gordon Wilson, M. Bohlke-Schneider, and Y. Wang, "Chronos: Learning the language of time series," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024, pp. 1–36, https://arxiv.org/abs/2403.07815.

[2] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo, "Unified training of universal time series forecasting transformers," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024, pp. 1–26, https://arxiv.org/abs/2402.02592.

[3] A. Das, W. Kong, R. Sen, and Y. Zhou, "A decoder-only foundation model for time-series forecasting," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024, pp. 1–23, https://arxiv.org/abs/2310.10688.

[4] K. Rasul, A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hassen, M. Bilos, S. Garg, A. Schneider, N. Chapados, A. Drouin, V. Zantedeschi, Y. Nevmyvaka, and I. Rish, "Lag-llama: Towards foundation models for probabilistic time series forecasting," in *Proceedings of the NeurIPS Workshop on Time Series in the Age of Large Models (NeurIPS-W)*, 2024, pp. 1–26, https://arxiv.org/abs/2310.08278.

[5] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023, pp. 1–24, https://arxiv.org/abs/2211.14730.

[6] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024, pp. 1–25, https://arxiv.org/abs/2310.06625.

[7] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023, pp. 1–23, https://arxiv.org/abs/2210.02186.

[8] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023, pp. 11 121–11 128, https://arxiv.org/abs/2205.13504.

[9] T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, S. Savarese, C. Xiong, and D. Sahoo, "Gift-eval: A benchmark for general time series forecasting model evaluation," in *Proceedings of the NeurIPS Workshop on Time Series in the Age of Large Models (NeurIPS-W)*, 2024, pp. 1–28, https://arxiv.org/abs/2410.10393.

[10] R. Godahewa, C. Bergmeir, G. Webb, R. Hyndman, and P. Montero-Manso, "Monash time series forecasting archive," in *Proceedings of the NeurIPS Datasets and Benchmarks Track (NeurIPS-DB)*, 2021, pp. 1–12, https://arxiv.org/abs/2105.06643.

[11] P. Montero-Manso, G. Athanasopoulos, R. Hyndman, and T. Talagala, "Fforma: Feature-based forecast model averaging," *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, 2020, https://doi.org/10.1016/j.ijforecast.2019.02.011.

[12] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021, pp. 11 106–11 115, https://arxiv.org/abs/2012.07436.

[13] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021, pp. 22 419–22 430, https://arxiv.org/abs/2106.13008.

[14] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," in *International Journal of Forecasting (IJF)*, 2020, pp. 1181–1191, https://arxiv.org/abs/1704.04110.

[15] B. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020, pp. 1–31, https://arxiv.org/abs/1905.10437.

[16] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020, https://doi.org/10.1016/j.ijforecast.2019.04.014.

[17] T. Talagala, R. Hyndman, and G. Athanasopoulos, "Meta-learning how to forecast time series," *Monash Econometrics and Business Statistics Working Paper Series*, vol. 18, no. 6, pp. 1–29, 2018, https://ideas.repec.org/p/msh/ebswps/2018-6.html.

[18] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017, pp. 1–19, https://arxiv.org/abs/1701.06538.

[19] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformer: Scaling to trillion parameter models with simple and efficient sparsity," in *Journal of Machine Learning Research (JMLR)*, 2022, pp. 1–40, https://arxiv.org/abs/2101.03961.

[20] T. Shnitzer, A. Ou, M. Silva, K. Soule, Y. Sun, J. Solomon, N. Thompson, and M. Yurochkin, "Large language model routing with benchmark datasets," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024, pp. 1–19, https://arxiv.org/abs/2309.15789.

[21] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. Gonzalez, W. Kadous, and I. Stoica, "Routellm: Learning to route llms with preference data," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024, pp. 1–26, https://arxiv.org/abs/2406.18665.

[22] D. Hu, T. Zhang, Y. Lin, X. Chen, and X. Ren, "Llm-blender: Ensembling large language models with pairwise ranking and generative

fusion," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024, pp. 1–18, https://arxiv.org/abs/2306.02561.

[23] R. Hyndman and A. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006, https://doi.org/10.1016/j.ijforecast.2006.03.001.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008, https://arxiv.org/abs/1706.03762.

[25] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019, pp. 1–19, https://arxiv.org/abs/1711.05101.