TAFS: Transformer-Attentive Feature Selection for Sequential Regression

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

***Abstract*—We study sequential regression in dynamically changing environments under a mixture-of-experts setting. A set of experts produces predictions in parallel, while the best-performing expert and the relevance of the available features may change over time. Our objective is to determine the expert combination weights using the features available to the experts and additional side information. We introduce TAFS, a transformer-based ensemble method that conditions feature attention on recent target history. The method represents individual features as tokens and uses their context-dependent representations to compute combination weights over a fixed pool of experts. We consider unconstrained, affine, and convex combinations and train the proposed model by minimizing cumulative prediction error while keeping the experts fixed. We establish differentiability properties of the training objective, a stationary-point convergence result under standard smoothness assumptions, and boundedness of convex combinations for bounded expert predictions. On a synthetic regime-switching task, TAFS reduces mean squared prediction error by 41% relative to its context-free variant. In the real-data experiments, TAFS reduces mean absolute scaled error by up to 56% compared with the LightGBM ensemble and by up to 55% compared with the strongest static feature-transform baseline. We release the source code to support reproducibility and further research.**

***Index Terms*—Time-series forecasting, feature selection, transformers, attention, ensemble learning, forecast combination, concept drift, interpretability.**

I. Introduction

A. Background

We study sequential regression under a mixture-of-experts setting, where several experts produce predictions of a target sequence in parallel. At each prediction step, we receive the features used by the experts and additional side information. Since the relative accuracy of the experts is unknown and may change over time, our objective is to use this information to determine their combination weights and minimize the prediction error of the resulting ensemble.

The experts may use different subsets or transformations of a high-dimensional engineered feature vector. Their combined feature set can contain redundant or irrelevant variables, and feature relevance may change over time [1]. To account for this variability, we compute expert combination weights from feature interactions conditioned on recent target observations and lagged input features.

We introduce Transformer-Attentive Feature Selection (TAFS) to implement this combination rule. TAFS represents each engineered feature as a separate token and processes these tokens jointly with a context token encoding recent target observations and their first differences. Self-attention [2] models interactions among the feature tokens and the temporal context, while a CLS token aggregates the resulting representations. A trainable combination head then maps the CLS output to the expert weights. We train the feature representation and combination head directly against the error of the combined prediction while keeping the base experts fixed.

Within this architecture, we consider unconstrained, affine, and convex expert weights. The unconstrained head permits arbitrary real-valued weights, whereas the affine head requires the weights to sum to one. The convex head additionally requires nonnegative weights. Thus, unconstrained and affine combinations can cancel expert outputs through negative weights, while a convex combination remains within the range of the expert predictions. We compare the three heads using the same feature-attention architecture.

B. Related Work

Existing approaches to expert combination learn weights from descriptive features or recent observations, or estimate individual expert errors before combining forecasts. A complementary line of work learns feature representations, using transformations or attention to model variable relevance and interactions. These approaches differ in how they use temporal information and whether the learned representation produces forecasts directly or determines expert combination weights.

FFORMA learns a mapping from time-series summary features, such as trend strength, seasonal strength, and autocorrelation statistics, to combination weights over candidate forecasting models [3]. In its original formulation, it assigns a combination weight vector to each series, whereas TAFS computes the weights at each prediction step. The arbitrated ensemble of Cerqueira et al. instead predicts the losses of individual experts and uses these estimates to combine their outputs [4]. In TAFS, we optimize the feature representation and combination head directly against the error of the combined prediction.

Fazla et al. learn time- and context-dependent combination weights using neural networks and boosted decision trees under unconstrained, affine, and convex constraints [5]. Their framework supplies the engineered feature vector to a meta-

H. Karaca and S. S. Kozat are with Bilkent University, Ankara, Türkiye (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 Türkiye (TÜBİTAK) 1515 Frontier R&D Laboratories Support Program for Turk Telekom 6G R&D Lab under project number 5249902, and by the TÜBİTAK 2210-A program.

learner that outputs the expert weights. TAFS adds a context-conditioned transformer that learns interactions among individual features before generating the weights.

Patel and Wikner use attention to pool candidate forecasts and incorporate recent dynamics through time-delayed embeddings [6]. Their attention mechanism computes pooling weights from representations of the system state and candidate forecasts. TAFS instead applies self-attention to engineered features and computes the expert weights through a separate combination head.

A related line of work learns feature transformations to construct the inputs to a predictor or combiner. A fixed feature gate assigns coefficients to the input variables; a linear projection combines them; and a nonlinear transformation models their interactions. TabNet uses input-dependent sequential attention to select features at successive decision steps [7], while the FT-Transformer represents individual tabular features as tokens and processes their interactions through a transformer [8]. Whereas these models learn representations for tabular prediction, we condition the feature processing on recent target observations and train its output for expert combination.

Attention-based forecasting methods model temporal dependencies and variable relevance. The dual-stage attention-based recurrent network selects driving variables using input attention and attends to past encoder states when generating predictions [9]. The Temporal Fusion Transformer combines variable-selection networks with recurrent and attention-based temporal processing [10]. The iTransformer applies attention across variate representations constructed from their historical observations [11]. These methods use the learned representations to generate forecasts, whereas TAFS uses its representation to combine predictions from fixed experts.

To assess the role of feature attention, we compare the feature-attention stage with diagonal, linear, low-rank, and nonlinear feature transforms. We also include a context-free transformer ablation to isolate the additional contribution of recent target history.

C. Contributions

Our contributions are as follows:

- We introduce a context-conditioned feature-axis transformer for expert combination in sequential regression. The feature representations and combination weights are learned jointly against the final prediction objective.
- We formulate the method under unconstrained, affine, and convex weight spaces and compare their prediction performance.
- We establish differentiability properties of the objective, derive a stationarity bound for stochastic gradient descent under standard assumptions, and show boundedness of convex combinations for bounded expert predictions. We relate restricted feature-mixing maps to the static feature transforms used as baselines.
- We compare TAFS with individual experts, conventional ensembles, learned combiners, and static feature transforms on a synthetic regime-switching task and two financial datasets. Context conditioning reduces synthetic mean squared prediction error by 41% relative to the context-free variant. TAFS achieves a mean absolute scaled error of 0.252 on the Istanbul Stock Exchange dataset and improves on simple averaging and the static feature transforms on Exchange-Rate returns, where a competing learned ensemble achieves lower error.
- We release the source code with all experimental details and preprocessing procedures to support reproducibility and further research.

D. Organization

Section II formulates the sequential regression and expert combination problem. Section III presents TAFS, its training procedure, and its theoretical properties. Section IV reports the synthetic experiments, financial-data comparisons, and ablation studies. Section V concludes the paper.

II. Problem Formulation

We consider a univariate sequential regression problem with exogenous covariates. At each step $t = 1, \dots, T$ the target $y_t \in \mathbb{R}$ is observed together with a vector of engineered features $\mathbf{x}_t \in \mathbb{R}^p$, which can be constructed from lags of y, rolling statistics, calendar variables and exogenous series. We assume a fixed pool of K base forecasters

$$\mathcal{F} = \{f_1, f_2, \dots, f_K\},$$

each of which produces a prediction $\hat{y}_t^{(k)} = f_k(\mathbf{x}_t^{(k)})$ for the response associated with prediction instance t. Here, $\mathbf{x}_t^{(k)}$ denotes the subset or transformation of the global feature vector $\mathbf{x}_t$ used by the kth forecaster, or equivalently, $\mathbf{x}_t$ is the union of all engineered features available to the system.

Let $\hat{\boldsymbol{y}}_t = \big(\hat{y}_t^{(1)}, \dots, \hat{y}_t^{(K)}\big)^\top \in \mathbb{R}^K$. A combination policy is a function π_θ that takes $\mathbf{x}_t$ together with a context summary $\mathbf{s}_t \in \mathbb{R}^{d_s}$, where d_s denotes the dimension of the context summary, and returns a weight vector

$$\mathbf{w}_t = \pi_\theta(\mathbf{x}_t, \mathbf{s}_t) \in \mathcal{W} \subseteq \mathbb{R}^K,$$

where the admissible set $\mathcal{W}$ encodes one of three constraint regimes: unconstrained, affine-constrained, or simplex-constrained:

$$\mathcal{W}_{\mathrm{u}} = \mathbb{R}^K, \tag{1}$$

$$\mathcal{W}_{\mathrm{a}} = \{\mathbf{w} \in \mathbb{R}^K : \mathbf{1}^\top \mathbf{w} = 1\}, \tag{2}$$

$$\mathcal{W}_{\mathrm{c}} = \{\mathbf{w} \in \mathbb{R}^K_{\geq 0} : \mathbf{1}^\top \mathbf{w} = 1\}. \tag{3}$$

The combined prediction is $\tilde{y}_t = \mathbf{w}_t^\top \hat{\boldsymbol{y}}_t$. Given a per-step loss $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ (e.g., squared, Huber, or pinball), training minimizes the cumulative loss

$$\mathcal{L}(\theta) \;=\; \frac{1}{T} \sum_{t=1}^{T} \ell\big(y_t,\, \pi_\theta(\mathbf{x}_t, \mathbf{s}_t)^\top \hat{\boldsymbol{y}}_t\big). \tag{4}$$

In the regime of interest p is typically large (tens to several hundred), T is moderate (several hundred to several thousand), and the conditional importance of features in $\mathbf{x}_t$ for estimating y_t is non-stationary in t. Selection mechanisms that compress $\mathbf{x}_t$ into a single static map discard this temporal

structure. Such a static map applies the same learned feature transformation at every prediction instance and does not adapt to the local context $\mathbf{s}_t$. We next introduce an algorithm that uses a context-conditioned feature-axis transformer to adapt feature interactions and compute weights over base forecasters, producing the final combined prediction in an efficient and context-dependent manner.

Remark 1 (Extensions)**.** *The scalar one-step notation is used only for clarity. Since the index t denotes a prediction instance, the same formulation extends to h-step-ahead forecasting by associating instance t with the future response y_{t+h}. For multi-step forecasting, the response can be written as $\mathbf{y}_{t,H} = (y_{t+1}, \ldots, y_{t+H})^\top$, and for multi-target regression the response vector can collect multiple target variables. In these cases, the combination policy and admissible weight sets remain unchanged; only the expert outputs and the loss become vector-valued.*

III. TAFS: Transformer-Attentive Feature Selection for Sequential Regression

We implement the combination policy π_θ, which maps $(\mathbf{x}_t, \mathbf{s}_t)$ to the expert weights $\mathbf{w}_t$, in three stages. First, the engineered features are represented as feature-axis tokens, which allows the model to assign time-varying relevance to individual features. Second, a context-conditioned transformer models interactions among these feature tokens and produces a task-level summary. Third, a constraint-aware combination head maps this summary to $\mathbf{w}_t$. Figure 1 summarizes the architecture.

A. Feature-Axis Tokenization

We represent each scalar engineered feature $x_{t,j}$ as a separate token, giving p feature tokens at step t. For feature j at step t, the input token is

$$\mathbf{t}_{t,j} \;=\; x_{t,j}\mathbf{a}_j \;+\; \mathbf{e}_j \;\in\; \mathbb{R}^d, \tag{5}$$

where $\mathbf{a}_j \in \mathbb{R}^d$ is a learnable scaling vector that lifts the scalar feature value into the model dimension d, and $\mathbf{e}_j \in \mathbb{R}^d$ is a learnable feature-identity embedding. Features are organized into named families (lags, rolling statistics, calendar, exogenous, regime indicators); to encourage early structure during training, we initialize $\mathbf{e}_j$ to family-specific initial values plus small Gaussian noise.

Before applying the transformer, we prepend two extra tokens. The first is a CLS token $\mathbf{c}_{\text{cls}} \in \mathbb{R}^d$, used as the readout for the combination head; the second is a context token $\mathbf{c}_t = W_c\mathbf{s}_t + \mathbf{b}_c$, where $\mathbf{s}_t$ stacks (a) the most recent L_y observed target values $(y_{t-1}, \ldots, y_{t-L_y})$, (b) recent residuals against a simple seasonal-naïve predictor, and (c) optional regime indicators (e.g. binary flags for trading vs. non-trading days). The input sequence to the transformer is

$$\mathbf{Z}_t^{(0)} \;=\; (\mathbf{c}_{\text{cls}},\, \mathbf{c}_t,\, \mathbf{t}_{t,1},\, \mathbf{t}_{t,2},\, \ldots,\, \mathbf{t}_{t,p}) \;\in\; \mathbb{R}^{(p+2)\times d}. \tag{6}$$

B. Feature-Axis Transformer

We apply a stack of L pre-norm transformer encoder layers [2] with h attention heads, hidden width d, and feed-forward width d_{ffn}, where MHA denotes multi-head attention and FFN denotes a feed-forward network:

$$\mathbf{Z}_t^{(\ell+\frac{1}{2})} = \mathbf{Z}_t^{(\ell)} + \mathrm{MHA}\big(\mathrm{LN}(\mathbf{Z}_t^{(\ell)})\big), \tag{7}$$

$$\mathbf{Z}_t^{(\ell+1)} = \mathbf{Z}_t^{(\ell+\frac{1}{2})} + \mathrm{FFN}\big(\mathrm{LN}(\mathbf{Z}_t^{(\ell+\frac{1}{2})})\big). \tag{8}$$

Named feature tokens require no temporal positional encoding. Unlike the order of temporal tokens, the order in which feature tokens are listed is arbitrary; the feature identity embeddings $\mathbf{e}_j$ already specify which token corresponds to which feature.

The final layer outputs $\mathbf{Z}_t^{(L)} \in \mathbb{R}^{(p+2)\times d}$. We define

$$\mathbf{u}_t \;=\; \mathbf{Z}_t^{(L)}[0,:] \;\in\; \mathbb{R}^d \tag{9}$$

as the task-aligned summary read from the CLS position. For interpretation, we also retain the per-feature attention weights $A_{0,j+2}^{(\ell,m)}$ from the CLS token to each feature token j at layer ℓ and head m; averaged over layers and heads, they give a single feature-importance score per step,

$$\alpha_{t,j} \;=\; \frac{1}{Lh}\sum_{\ell=1}^{L}\sum_{m=1}^{h} A_{0,\,j+2}^{(\ell,m)}. \tag{10}$$

We use $\alpha_{t,j}$ to produce the heatmaps in Section IV-D.

Remark 2. *If we remove the context token, restrict the transformer to a single layer with a single head, and tie the queries and keys to be the identity, the CLS readout reduces to a weighted sum $\sum_j \alpha_j \mathbf{a}_j x_{t,j}$ with α_j shared across t. This recovers diagonal gating; further re-parameterizing $\mathbf{a}_j$ recovers linear and low-rank transforms, and replacing* MHA *with a multilayer perceptron (MLP) applied to the concatenated feature tokens recovers a static nonlinear feature transform. The static feature transforms used as baselines in Section IV are therefore strict special cases.*

C. Constraint-Aware Combination Head

A small head maps $\mathbf{u}_t$ to combination weights $\mathbf{w}_t$. Let $\mathbf{z}_t = W_2\,\mathrm{GELU}(W_1\mathbf{u}_t + \mathbf{b}_1) + \mathbf{b}_2 \in \mathbb{R}^K$, where W_1, W_2 are learnable weight matrices and $\mathbf{b}_1, \mathbf{b}_2$ are learnable bias vectors of compatible dimensions. We then project $\mathbf{z}_t$ onto $\mathcal{W}$:

- *Unconstrained:* $\mathbf{w}_t = \mathbf{z}_t$.
- *Affine:* $\mathbf{w}_t = \mathbf{z}_t - \frac{1}{K}(\mathbf{1}^\top\mathbf{z}_t - 1)\mathbf{1}$, where K is the number of base forecasters, i.e. the orthogonal projection onto the hyperplane $\mathbf{1}^\top\mathbf{w} = 1$.
- *Convex:* $\mathbf{w}_t = \mathrm{softmax}(\mathbf{z}_t/\tau)$ with temperature τ=1, where
$$[\mathrm{softmax}(\mathbf{v})]_k = \frac{\exp(v_k)}{\sum_{r=1}^{K}\exp(v_r)}.$$

All three mappings are smooth in θ (the affine mapping is linear and the softmax is smooth), so that the cumulative loss in (4) is differentiable everywhere and back-propagation through the head, the transformer and the tokenizer is standard. During TAFS training, the parameters of the base forecasters are kept fixed; only the tokenizer, transformer and combination head are updated.

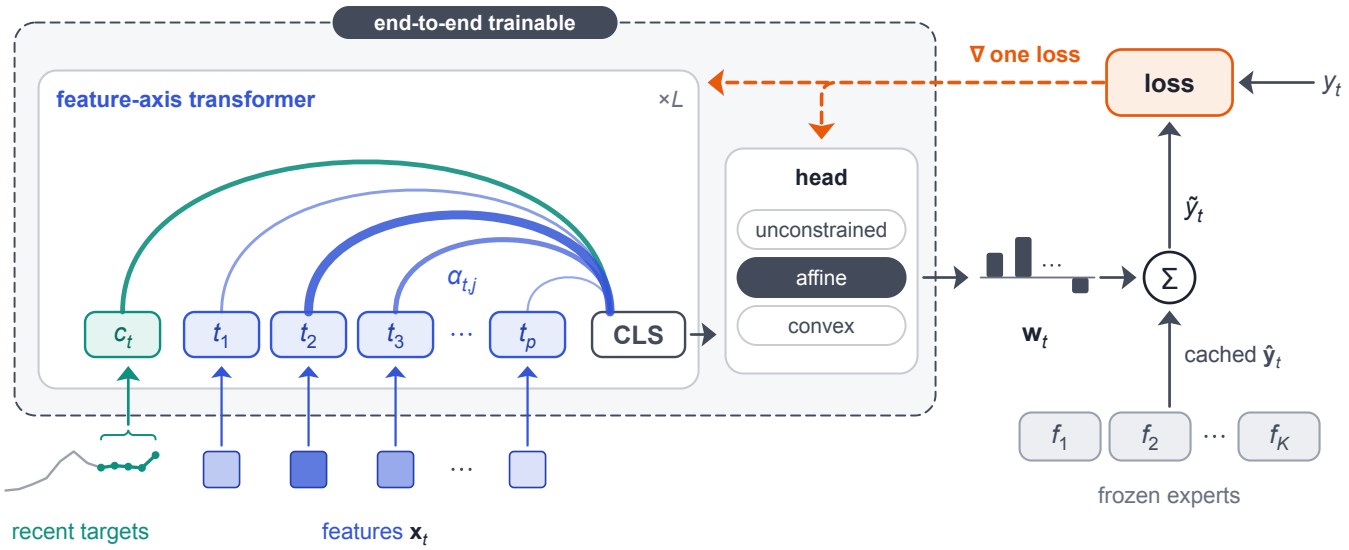


Fig. 1. Overview of the proposed TAFS framework. Each of the p engineered features is treated as a token with a learnable identity embedding, and a context token derived from lagged targets and regime indicators is prepended. A feature-axis transformer produces a task-aligned summary $\mathbf{u}_t$, which is mapped by a combination head to weights $\mathbf{w}_t$ over K pretrained base forecasters under one of three constraint regimes (unconstrained, affine, convex). The final prediction is a weighted sum of cached base outputs $\hat{\mathbf{y}}_t$, and the cumulative prediction error is back-propagated end-to-end through the transformer and the head; base predictors are kept frozen.

D. Training Procedure

Algorithm 1 summarizes training. We split the sequence chronologically into training, validation, and test segments of lengths T_{tr}, T_{val}, and T_{te}, respectively; validation is used for early stopping and hyperparameter selection, while the test segment is kept unseen until final evaluation. We use AdamW [12] with linear warmup followed by cosine annealing, mini-batches drawn over time-aligned windows, gradient clipping, and an early-stopping criterion on the rolling validation loss.

At inference time, TAFS adds a single forward pass through a small transformer and a linear head; computing the base predictions $\hat{\boldsymbol{y}}_t$ dominates the cost, and these predictions are already required by the simple-average baseline. The inference overhead is therefore comparable to that of a single small ensemble combiner.

Remark 3 (Online extension)**.** *Algorithm 1 describes offline training on a fixed chronological training segment. The same formulation can be adapted to an online setting by maintaining a sliding window of the most recent observations and periodically updating θ on this window. This extension changes only the training schedule; the base-forecaster interface, the combination policy, and the admissible weight sets remain unchanged.*

Algorithm 1 Offline Training of TAFS

Require: Training set $\{(\mathbf{x}_t, \mathbf{s}_t, y_t)\}_{t=1}^{T_{\text{tr}}}$; frozen base forecasters $\{f_1, \dots, f_K\}$; per-step loss ℓ; constraint regime $\mathcal{W}$; optimizer hyperparameters.
Ensure: Trained parameters $\theta = \{\mathbf{a}_j, \mathbf{e}_j, W_c, \mathbf{b}_c, \text{transformer}, W_1, \mathbf{b}_1, W_2, \mathbf{b}_2\}$.
Cache base predictions $\hat{\boldsymbol{y}}_t = (f_k(\mathbf{x}_t))_{k=1}^K$ for all t.
Initialize θ (Xavier; identity embeddings as in Section III-A).
for epoch $= 1, \dots, E$ **do**
  for mini-batch $\mathcal{B} \subset \{1, \dots, T_{\text{tr}}\}$ **do**
    Build $\mathbf{Z}_t^{(0)}$ for every $t \in \mathcal{B}$.
    Compute $\mathbf{u}_t$ through the transformer; head $\to \mathbf{z}_t$.
    Project to $\mathbf{w}_t \in \mathcal{W}$.
    Predict $\tilde{y}_t = \mathbf{w}_t^\top \hat{\boldsymbol{y}}_t$.
    Compute $\mathcal{L}_{\mathcal{B}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \ell(y_t, \tilde{y}_t)$.
    Update $\theta \leftarrow \text{AdamW}(\theta, \nabla_\theta \mathcal{L}_{\mathcal{B}})$.
  end for
  Evaluate $\mathcal{L}_{\text{val}}$; early-stop if no improvement for E_{pat} epochs.
end for

E. Theoretical Properties

We establish the following properties of TAFS. These results are not intended to establish global optimality, since the transformer parameterization and the combination head together define a nonconvex learning problem. Instead, they formalize four properties that are used implicitly by the proposed training procedure: the empirical objective is suitable for first-order optimization; standard nonconvex convergence guarantees apply under the usual smoothness assumptions [13], [14]; the static feature transforms considered in the experiments are restricted instances of the proposed feature-mixing mechanism; and the simplex-constrained head provides a simple boundedness guarantee.

Let

$$\tilde{y}_t(\theta) = \pi_\theta(\mathbf{x}_t, \mathbf{s}_t)^\top \hat{\mathbf{y}}_t$$

denote the combined prediction at time t, and let

$$L(\theta) = \frac{1}{T}\sum_{t=1}^{T} \ell\big(y_t, \tilde{y}_t(\theta)\big)$$

be the empirical objective in (4). Throughout this subsection, the base forecasters remain fixed. Therefore, $\hat{\mathbf{y}}_t = (\hat{y}_t^{(1)}, \ldots, \hat{y}_t^{(K)})^\top$ is treated as fixed with respect to the trainable TAFS parameters θ, while the tokenizer, transformer, and combination head remain trainable.

Proposition 1 (Differentiability of the TAFS objective)**.** *Assume that the per-step loss $\ell(y, \hat{y})$ is continuously differentiable with respect to its second argument, as in the squared or Huber losses used here, and that the transformer, tokenizer, and MLP head use differentiable or piecewise differentiable operations. Then, for the unconstrained, affine-constrained, and simplex-constrained heads, the empirical objective $L(\theta)$ is differentiable almost everywhere with respect to the trainable parameters θ. Moreover, the projection from the head logits $\mathbf{z}_t$ to the feasible weight set is linear for the affine-constrained head and smooth for the simplex-constrained head.*

Proof. For each time step t, the network first maps $(\mathbf{x}_t, \mathbf{s}_t)$ to the CLS representation $\mathbf{u}_t$, then maps $\mathbf{u}_t$ to the head logits $\mathbf{z}_t \in \mathbb{R}^K$, and finally maps $\mathbf{z}_t$ to the combination weights $\mathbf{w}_t \in \mathcal{W}$. We consider the three constraint regimes.

For the unconstrained head,

$$\mathbf{w}_t = \mathbf{z}_t,$$

and therefore introduces no additional non-differentiability. The combined prediction is

$$\tilde{y}_t = \mathbf{z}_t^\top \hat{\mathbf{y}}_t,$$

which is differentiable with respect to $\mathbf{z}_t$ because $\hat{\mathbf{y}}_t$ is fixed.

For the affine-constrained head, the projection is

$$\mathbf{w}_t = \mathbf{z}_t - \frac{1}{K}\big(\mathbf{1}^\top \mathbf{z}_t - 1\big)\mathbf{1}.$$

This affine function is differentiable in $\mathbf{z}_t$. It also satisfies the required constraint because

$$\mathbf{1}^\top \mathbf{w}_t = \mathbf{1}^\top \mathbf{z}_t - \frac{1}{K}\big(\mathbf{1}^\top \mathbf{z}_t - 1\big)\mathbf{1}^\top \mathbf{1} = \mathbf{1}^\top \mathbf{z}_t - \big(\mathbf{1}^\top \mathbf{z}_t - 1\big) = 1.$$

Thus $\mathbf{w}_t \in \mathcal{W}_{\text{aff}}$ and the mapping from $\mathbf{z}_t$ to $\mathbf{w}_t$ is differentiable.

For the simplex-constrained head, the weights are given by

$$w_{t,k} = \frac{\exp(z_{t,k}/\tau)}{\sum_{j=1}^{K} \exp(z_{t,j}/\tau)}, \qquad k = 1, \ldots, K,$$

with temperature $\tau > 0$. The softmax map is smooth for every finite $\mathbf{z}_t$, and it satisfies $w_{t,k} \geq 0$ and $\sum_{k=1}^{K} w_{t,k} = 1$, so that $\mathbf{w}_t \in \mathcal{W}_{\text{simp}}$.

In all three cases, $\tilde{y}_t = \mathbf{w}_t^\top \hat{\mathbf{y}}_t$ is a composition of the network map, the head projection, and a linear inner product with the fixed vector $\hat{\mathbf{y}}_t$. Since $\ell(y_t, \hat{y})$ is continuously differentiable in $\hat{y}$, each summand $\ell(y_t, \tilde{y}_t(\theta))$ is differentiable wherever the underlying network operations are differentiable. The transformer, tokenizer, normalization layers, residual connections, linear maps, and activation functions used in the architecture are differentiable or piecewise differentiable; hence the empirical average $L(\theta)$ is differentiable almost everywhere. This is sufficient for standard back-propagation-based optimization. □

The following proposition follows from the standard analysis of smooth nonconvex stochastic optimization [13], [14].

Proposition 2 (Convergence to a stationary point)**.** *Suppose that $L(\theta)$ is bounded below by $L^\star$ and has G-Lipschitz continuous gradients, i.e.,*

$$\|\nabla L(\theta) - \nabla L(\theta')\|_2 \leq G\|\theta - \theta'\|_2$$

for all θ, θ'. Assume further that the mini-batch gradient $\mathbf{g}_r$ at iteration r is an unbiased estimator of the full gradient,

$$\mathbb{E}[\mathbf{g}_r \mid \theta_r] = \nabla L(\theta_r),$$

and has a bounded second moment,

$$\mathbb{E}[\|\mathbf{g}_r\|_2^2 \mid \theta_r] \leq C$$

for some constant $C > 0$. Then stochastic gradient descent (SGD) updates of the form

$$\theta_{r+1} = \theta_r - \eta_r \mathbf{g}_r$$

with the standard horizon-dependent step size $\eta_r = \eta/\sqrt{N}$ satisfy

$$\min_{0 \leq r < N} \mathbb{E}\|\nabla L(\theta_r)\|_2^2 = \mathcal{O}(N^{-1/2}).$$

Consequently, under the stated assumptions, SGD applied to the TAFS objective converges to a first-order stationary point in the standard nonconvex optimization sense.

Proof. By the G-smoothness of L, for one stochastic-gradient update we have

$$L(\theta_{r+1}) \leq L(\theta_r) + \langle \nabla L(\theta_r), \theta_{r+1} - \theta_r \rangle + \frac{G}{2}\|\theta_{r+1} - \theta_r\|_2^2.$$

Substituting $\theta_{r+1} - \theta_r = -\eta_r \mathbf{g}_r$ gives

$$L(\theta_{r+1}) \leq L(\theta_r) - \eta_r \langle \nabla L(\theta_r), \mathbf{g}_r \rangle + \frac{G\eta_r^2}{2}\|\mathbf{g}_r\|_2^2.$$

Taking conditional expectation with respect to the mini-batch randomness and using $\mathbb{E}[\mathbf{g}_r \mid \theta_r] = \nabla L(\theta_r)$ yields

$$\mathbb{E}[L(\theta_{r+1}) \mid \theta_r] \leq L(\theta_r) - \eta_r \|\nabla L(\theta_r)\|_2^2 + \frac{G\eta_r^2}{2}\mathbb{E}[\|\mathbf{g}_r\|_2^2 \mid \theta_r].$$

Using the bounded second moment assumption,

$$\mathbb{E}[L(\theta_{r+1})] \leq \mathbb{E}[L(\theta_r)] - \eta_r \mathbb{E}\|\nabla L(\theta_r)\|_2^2 + \frac{GC\eta_r^2}{2}.$$

Rearranging and summing from $r = 0$ to $N - 1$ gives

$$\sum_{r=0}^{N-1} \eta_r \mathbb{E}\|\nabla L(\theta_r)\|_2^2 \leq L(\theta_0) - L^\star + \frac{GC}{2}\sum_{r=0}^{N-1} \eta_r^2.$$

Therefore,

$$\min_{0 \leq r < N} \mathbb{E}\|\nabla L(\theta_r)\|_2^2 \leq \frac{L(\theta_0) - L^\star + \frac{GC}{2}\sum_{r=0}^{N-1} \eta_r^2}{\sum_{r=0}^{N-1} \eta_r}.$$

For the usual nonconvex stochastic-optimization choice $\eta_r = \eta/\sqrt{N}$ over a horizon of N iterations, the denominator is of order $\sqrt{N}$ while the numerator remains bounded up to a constant independent of N. Hence

$$\min_{0\le r<N} \mathbb{E}\|\nabla L(\theta_r)\|_2^2 = \mathcal{O}(N^{-1/2}).$$

This proves convergence to a first-order stationary point in the standard sense. Although the implementation uses AdamW for empirical efficiency, the result shows that the differentiable TAFS objective is compatible with standard first-order convergence guarantees. □

Remark 4 (Practical optimizer)**.** *The training procedure in Algorithm 1 uses AdamW rather than plain SGD. Proposition 2 establishes that the TAFS objective satisfies the smoothness and boundedness conditions required by standard first-order nonconvex analyses. Under analogous Lipschitz-gradient and bounded stochastic-moment assumptions, adaptive variants of Adam/AdamW have also been shown to converge to first-order stationary points with appropriate hyperparameter choices [15], [16]. Therefore, Proposition 2 should be interpreted as a theoretical guarantee that the proposed objective is compatible with first-order optimization. The implementation uses AdamW because of its superior empirical stability and convergence behavior in transformer-based architectures.*

Proposition 3 (Static transforms as restricted or context-free TAFS variants)**.** *The diagonal, linear, and low-rank static feature transforms used as baselines can be represented as restricted instances of the TAFS feature-mixing module. The nonlinear MLP transform is a context-free static feature-mixing counterpart. Thus, the full TAFS architecture generalizes these static alternatives by allowing the feature-mixing rule to depend on the context summary $\mathbf{s}_t$ through the context token $\mathbf{c}_t$ and vary across time.*

Proof. Consider a restricted TAFS module in which the context token is removed, the transformer has a single attention layer and a single attention head, and the CLS-to-feature attention weights are independent of time. Under this restriction, the CLS readout can be written in the form

$$\mathbf{u}_t = \sum_{j=1}^{p} \alpha_j \mathbf{a}_j x_{t,j},$$

where α_j is a time-invariant attention coefficient and $\mathbf{a}_j \in \mathbb{R}^d$ is the learnable lifting vector associated with feature j. Thus, each scalar feature $x_{t,j}$ contributes through a fixed learned vector $\alpha_j \mathbf{a}_j$. This is a static linear feature map from $\mathbf{x}_t$ to $\mathbf{u}_t$.

A diagonal gate is obtained by restricting the output dimension to match the input dimension and forcing each feature to affect only its own coordinate. In that case, the transformation has the form

$$\tilde{\mathbf{x}}_t = D\mathbf{x}_t,$$

where D is a learned diagonal matrix.

A full linear transform is obtained by allowing the lifted feature vectors to contribute to arbitrary output coordinates. Then the restricted readout can be written as

$$\tilde{\mathbf{x}}_t = A\mathbf{x}_t$$

for a learned matrix A.

A low-rank transform is obtained by constraining the linear map to factor through an r-dimensional bottleneck,

$$\tilde{\mathbf{x}}_t = UV^\top \mathbf{x}_t, \qquad U \in \mathbb{R}^{q\times r}, \quad V \in \mathbb{R}^{p\times r}, \quad r < \min(p,q).$$

This is the same static linear feature map under a rank constraint.

A nonlinear MLP bottleneck is a context-free static special case of the TAFS feature-mixing module. Specifically, setting the context token to zero, replacing the multi-head attention block with a feed-forward block applied directly to the concatenated feature tokens, and tying the resulting map across time steps yields a static nonlinear transform of the form

$$\tilde{\mathbf{x}}_t = \phi(\mathbf{x}_t),$$

which is independent of the context summary $\mathbf{s}_t$. This is not a strict algebraic recovery from the attention parameterization of TAFS; rather, it is a degenerate instance of the broader class of context-free feature-mixing mechanisms included in the architecture.

Hence, the static baselines are obtained by removing the context token and restricting the attention or feature-mixing map so that the same transformation is applied at every time step. The full TAFS model removes these restrictions: the context token $\mathbf{c}_t = W_c\mathbf{s}_t + \mathbf{b}_c$ participates in the transformer, and the resulting CLS-to-feature attention can change with $\mathbf{s}_t$. Therefore, TAFS contains the static transforms as restricted cases while additionally permitting context-conditioned, time-varying feature selection. □

Proposition 4 (Boundedness of the simplex-constrained combiner)**.** *Assume that the base predictions are uniformly bounded, i.e., there exists $B > 0$ such that*

$$|\hat{y}_t^{(k)}| \le B$$

for every time step t and every base forecaster $k \in \{1,\dots,K\}$. Under the simplex-constrained head $\mathcal{W}_{\text{simp}} = \{\mathbf{w} \in \mathbb{R}^K_{\ge 0} : \mathbf{1}^\top\mathbf{w} = 1\}$, the combined prediction is also bounded:

$$|\tilde{y}_t| \le B$$

for every t.

Proof. Under the simplex-constrained head, the combination weights satisfy $w_{t,k} \ge 0$ and $\sum_{k=1}^K w_{t,k} = 1$. Therefore,

$$|\tilde{y}_t| = \left|\sum_{k=1}^K w_{t,k}\hat{y}_t^{(k)}\right| \le \sum_{k=1}^K w_{t,k}|\hat{y}_t^{(k)}| \le \sum_{k=1}^K w_{t,k}B = B.$$

Thus, the simplex-constrained combination remains within the interval spanned by the bounded base predictions. This gives the simplex-constrained head a simple stability property. In contrast, the affine-constrained and unconstrained heads may produce predictions outside the range of the base predictors, which can be beneficial when negative or extrapolative weights help cancel an inaccurate base forecaster, but they do not provide the same boundedness guarantee. □

The four propositions clarify the role of the architectural choices. The unconstrained and affine-constrained heads

TABLE I
DATASETS USED IN OUR EVALUATION. n IS THE NUMBER OF ONE-STEP PREDICTION WINDOWS AFTER THE FEATURE PIPELINE; p IS THE NUMBER OF ENGINEERED FEATURES; K IS THE SIZE OF THE FROZEN BASE-PREDICTOR POOL; H IS THE FORECASTING HORIZON.

Dataset	n	p	K	H	domain
Synthetic (regime-switch)	31 360	32	3	1	controlled
Istanbul Stock Exchange	526	94	7	1	finance
Exchange Rate	7 563	102	7	1	finance

preserve differentiability while allowing extrapolative combinations; the simplex-constrained head restricts the prediction to the convex hull of the base forecasts and therefore provides a boundedness guarantee. The static-transform result explains why the proposed model can be viewed as a context-conditioned generalization of diagonal gating, linear projections, low-rank bottlenecks, and nonlinear static transforms: these alternatives correspond to removing the context-conditioned attention mechanism and applying the same feature map at every time step.

Remark 5 (Extension to vector-valued outputs)**.** *We use scalar one-step prediction to simplify the notation. For multi-step forecasting or multi-target regression, the expert outputs and the combined prediction become vector-valued, and the loss can be applied componentwise or averaged over the output dimension. Under the same smoothness and boundedness assumptions on this vector-valued loss, the differentiability and first-order convergence arguments remain unchanged.*

IV. EXPERIMENTS

A. Datasets and Feature Extraction

We evaluate on a controlled synthetic regime-switch task and on two public real-world financial datasets: the Istanbul Stock Exchange (ISE) national-100 daily return series [17] and a daily Exchange-Rate panel of eight currencies [18]. The synthetic task is the only setting in which the ground-truth regime structure and the per-step optimal feature subset are known, so that it isolates whether TAFS can recover context-dependent feature relevance and regime-dependent expert suitability; the financial datasets test the method on non-stationary, low signal-to-noise series of the kind that adaptive forecasting systems are expected to handle in practice [19]. Table I gives the number of one-step prediction windows, the number of engineered features, the size of the base-predictor pool, and the forecasting horizon.

For the real datasets, we apply the same engineered-feature pipeline: lags of the target and of the other series in the panel (up to $L_{\max}$=10 for ISE and $L_{\max}$=24 for Exchange Rate); rolling means, medians, and standard deviations over short windows ($\{3, 5, 10\}$ and $\{3, 7, 14, 28\}$ respectively); and calendar variables. Because exchange-rate levels are non-stationary and can drift outside the training range, where tree-based predictors cannot extrapolate reliably, we forecast log-returns on the Exchange-Rate panel; the ISE series is already a returns series. All features and the context summary are standardized on the training window and the same statistics are applied to the validation and test windows. The synthetic generator and its p=32 features (2 regime-A signal features, 2 regime-B signal features, and 28 irrelevant noise features) are described in Section IV-D.

B. Models and Baselines

Our baselines are organized into four groups; the same frozen base-predictor pool is used throughout so that differences arise only from the selection/combination mechanism.

a) Individual base predictors.: For the real datasets, the pool consists of LightGBM [20], XGBoost [21], CatBoost [22], Random Forest [23], Lasso [24], an MLP, and a Naïve last-value predictor (K=7). For the synthetic task, the pool is the three generative predictors (a lag predictor, an exogenous predictor, and their average, K=3). Each predictor is fit once on the training window; base forecasters consume the standardized feature vector $\mathbf{x}_t$ and are not retrained when paired with a static transform or with TAFS, so that the transform and the combination head are the only learnable modules in those configurations. We report the individual predictors for reference; the best individual predictor for each dataset is identifiable only in hindsight on the test fold, whereas the best single predictor baseline below is its causal, validation-selected counterpart.

b) Static learned feature transforms.: Diagonal gating, full linear transform, low-rank (r=16) bottleneck, and a small nonlinear MLP transform. Each transform produces an alternative representation $\tilde{\mathbf{x}}_t$ that is consumed by the affine combination head; the frozen base forecasters continue to operate on the raw $\mathbf{x}_t$, so that no base predictor is retrained. The transforms share the same end-to-end training procedure, loss, and base pool as TAFS and differ from it only in that the transform is static while TAFS conditions on the context $\mathbf{s}_t$.

c) Meta-learners without feature selection.: An MLP-ensemble combiner under the affine constraint and a LightGBM-ensemble combiner under the convex constraint; both consume the same engineered features and the base predictions, in the manner of stacked and meta-learned forecast combiners [4], [25].

d) Conventional ensembles and oracle.: The simple average of all base predictors; the validation-selected best single predictor; and a per-step oracle that selects the base predictor with the lowest error at each t (a reference lower bound rather than a competing method).

C. Training and Evaluation Setup

For all learnable methods, we use AdamW [12] with learning rate 10^{-3}, weight decay 10^{-2}, batch size 128, gradient clipping at 1.0, a 200-step linear warmup followed by cosine decay, and early stopping on the validation combined-prediction error (patience 8–12 epochs, 30–80 epochs total). The per-step loss ℓ in (4) is the squared error of the combined prediction $\mathbf{w}_t^\top \hat{\mathbf{y}}_t$; this directly optimizes the cumulative objective and remains well-posed for the affine and unconstrained heads, for which the routing-style surrogate $\sum_k w_{t,k}\, e_{t,k}$ is unbounded. The Huber and pinball options described in Section II are available in the framework but are not evaluated

TABLE II
SYNTHETIC REGIME-SWITCH TASK. COMBINED-PREDICTION MSE (MEAN OVER THREE SEEDS; STANDARD DEVIATION IN PARENTHESES). MASE IS UNDEFINED UNDER THE STATIONARY-NOISE GENERATOR; WE THEREFORE REPORT MSE. TAFS APPROACHES THE ORACLE WHILE STATIC TRANSFORMS HAVE HIGHER ERROR.

Method	MSE (↓)
Best single base predictor	0.4162 (0.0043)
Equal-weight average	0.3566 (0.0042)
Diagonal gating	0.3248 (0.0025)
Linear transform	0.3246 (0.0026)
Low-rank transform	0.3395 (0.0041)
Nonlinear bottleneck	0.2392 (0.0020)
TAFS (no context)	0.2188 (0.0010)
TAFS (full)	**0.1292 (0.0024)**
Oracle	0.0581 (0.0003)

here. The default TAFS configuration uses d=64, h=4 heads, L=2 layers, d_{ffn}=128, dropout 0.1, a 16-dimensional context token, and the affine head; departures from these defaults are reported in the ablation tables. TAFS and the MLP- and LightGBM-ensemble combiners each have on the order of 10^5 trainable parameters. With cached base predictions, base-predictor evaluation dominates end-to-end inference cost and the TAFS forward pass adds only a small overhead; each configuration trains in minutes on a single multi-core CPU.

We report MASE [26] as the primary metric on the real datasets (it is loss-agnostic and scale-free) and sMAPE as a secondary check; on the synthetic task, whose stationary-noise generator makes the naive-difference MASE denominator ill-defined, we report mean squared error (MSE). We report means over three training seeds, with standard deviations in parentheses. All experiments are one-step (H=1). The real datasets use a chronological $60/20/20$ train/validation/test split with standardization fit on the training window only (no leakage), following standard practice for performance estimation on time-ordered data [27]; the synthetic task uses a $70/10/20$ split, which is unbiased because the series is stationary in distribution within each regime.

D. Results and Discussion

1) Synthetic Regime-Switch Demonstration: The synthetic study isolates the contribution of context-conditioned attention in a setting where the ground truth is known. We generate N=80 trajectories of length T=400 from a two-regime process. Under regime A ($t < 200$) the target y_t depends only on its first lag and a rolling-7 mean; under regime B ($t \geq 200$) it depends only on two exogenous series. The remaining 28 of the p=32 engineered features are nuisance features, and the three base predictors are a lag predictor (effective in regime A), an exogenous predictor (effective in regime B), and their average. The change point at t=200 is hidden from every model.

Table II reports the combined-prediction MSE and Table III the attention that the full model places on each feature group. Because the 28 nuisance tokens dominate the summed attention purely by count, the informative comparison is the mean attention per feature: on this basis, the two informative exogenous features receive $8.8\times$ (regime A) and $5.8\times$ (regime B) the attention of a nuisance feature, while the lag/rolling features, whose information is already carried by the context token, receive little token-level attention. The attention map is thus best interpreted as evidence of nuisance suppression rather than a clear shift from lag features to exogenous features.

The MSE in Table II provides the key result. The full TAFS model attains an MSE of 0.129, approaching the per-step oracle (0.058) and remaining well below the best static transform (the nonlinear bottleneck, 0.239) and the equal-weight average (0.357). Removing the context token raises the MSE to 0.219, a 69% increase. This is the single largest architectural effect in the study and provides direct evidence that time-varying, context-conditioned attention, rather than the feature-axis transformer alone, recovers the regime structure.

2) Real-Data Comparison: Table IV reports the combined one-step MASE on the two real datasets and Table V a representative sMAPE comparison. Highlighting is restricted to the combination methods, because the best individual predictor (a single Lasso on both datasets) is identifiable only in hindsight on the test fold; its validation-selected, causal counterpart is the best single predictor row.

The affine and unconstrained TAFS heads are the lowest-error learned combiners in the main comparison, both attaining a MASE of 0.252. They improve on the validation-selected best-single baseline, simple averaging, both meta-learners, and all static-transform baselines, although the best individual base predictor in hindsight has a lower MASE.

On the Exchange-Rate returns, the improvements are more limited. These series are close to a random walk, so that the scope for improvement by any combiner is small: the oracle is 0.198 and the strongest causal methods cluster near 0.45. TAFS (0.458 convex, 0.461 unconstrained) still improves on simple averaging (0.466), on the validation-selected best single predictor (0.485), and on every static transform (0.530–0.812), and its difference from the best baseline here, the LightGBM-ensemble (0.447), is within the observed variability. TAFS does not consistently outperform the baselines on this dataset: when the informative feature set is stable and the signal is weak, context conditioning cannot help and a well-regularized combiner is difficult to outperform. The constraint regime is a secondary factor, analyzed with the ablations below.

3) Ablation Studies: Table VI isolates the feature-mixing module. Replacing the feature-axis transformer with each static transform (holding the base pool, head, loss, and training procedure fixed) yields an aggregate MASE between 0.55 and 1.17 on the real data, whereas both TAFS variants achieve ≈ 0.36. The transformer is therefore responsible for the largest improvement, with a MASE roughly a third below that of the best static transform (the nonlinear bottleneck at 0.546). Within TAFS, the full and context-free variants are statistically indistinguishable on the real data (0.361 vs. 0.358).

Table VII explains why. Removing the context token leaves the real-data MASE nearly unchanged, but increases the synthetic MSE from 0.129 to 0.219. Context conditioning helps only when the informative feature subset is itself non-

TABLE III

CLS→FEATURE ATTENTION OF THE FULL TAFS MODEL ON THE SYNTHETIC TASK, BY FEATURE GROUP AND REGIME. *Mass* IS THE TOTAL ATTENTION RECEIVED BY THE GROUP; *per feature* DIVIDES THE MASS BY THE GROUP SIZE, BECAUSE THE 28 NUISANCE TOKENS OTHERWISE DOMINATE THE SUMMED MASS PURELY BY COUNT; *ratio* IS THE PER-FEATURE ATTENTION RELATIVE TO A NUISANCE FEATURE. THE INFORMATIVE EXOGENOUS FEATURES RECEIVE SEVERAL TIMES THE PER-FEATURE ATTENTION OF THE NUISANCE FEATURES IN BOTH REGIMES; LAG/ROLLING INFORMATION IS LARGELY CARRIED BY THE CONTEXT TOKEN, SO THAT ITS FEATURE TOKENS RECEIVE LITTLE TOKEN-LEVEL ATTENTION.

		Regime A ($t < 200$)			Regime B ($t \geq 200$)		
Feature group	Size	Mass	Per feature	Ratio	Mass	Per feature	Ratio
Exogenous signal (regime-B drivers)	2	0.364	**0.182**	8.8×	0.272	**0.136**	5.8×
Lag/rolling signal (regime-A drivers)	2	0.057	0.029	1.4×	0.069	0.035	1.5×
Nuisance	28	0.579	0.021	1.0×	0.659	0.024	1.0×

TABLE IV

MAIN RESULTS ON THE REAL-WORLD DATASETS: COMBINED ONE-STEP PREDICTION MASE (MEAN OVER THREE SEEDS; LOWER IS BETTER). THE BEST NON-ORACLE SCORE PER COLUMN IS IN **BOLD**, THE SECOND BEST IS UNDERLINED. THE ORACLE IS A PER-STEP BEST-BASE-PREDICTOR LOWER BOUND AND SERVES AS A REFERENCE.

Method	ISE	Exchange
Individual base predictors		
LightGBM	0.270	0.454
XGBoost	0.264	0.485
CatBoost	0.248	0.442
Random Forest	0.248	0.424
Lasso	0.216	0.422
MLP	9.557	1.106
Naïve	0.744	0.818
Conventional ensemble baselines		
Simple average	1.419	0.466
Best single predictor	0.270	0.485
Static feature transforms (affine head)		
Diagonal gating	1.531	0.787
Linear transform	1.537	0.812
Low-rank (r=16)	0.911	0.631
Nonlinear MLP	0.563	0.530
Meta-learners without feature selection		
MLP-ensemble (affine)	0.587	0.560
LGBM-ensemble (convex)	0.577	**0.447**
Proposed		
TAFS (unconstrained)	**0.252**	0.461
TAFS (affine)	0.252	0.470
TAFS (convex)	0.421	0.458
Oracle (per-step best)	0.150	0.198

TABLE V

SMAPE (%, ↓) OF THE COMBINED PREDICTION FOR REPRESENTATIVE COMPETITORS (MEAN OVER THREE SEEDS).

Method	ISE	Exchange
LightGBM	80.6	**131.4**
Nonlinear MLP	106.5	133.2
LGBM-ensemble (convex)	96.8	131.5
TAFS (affine)	**76.8**	132.8
Oracle	55.3	76.8

TABLE VI

CENTRAL ARCHITECTURAL ABLATION: AGGREGATE COMBINED-PREDICTION MASE OVER THE REAL DATASETS (LOWER IS BETTER). STATIC TRANSFORMS AND TAFS WITHOUT CONTEXT ARE COMPARED AGAINST THE FULL MODEL.

Variant	Aggregate MASE (real)
Diagonal gating	1.159
Linear transform	1.174
Low-rank transform	0.771
Nonlinear bottleneck	0.546
TAFS-noCtx	0.358
TAFS (full)	0.361

TABLE VII

CONTEXT-CONDITIONING ABLATION. REMOVING THE CONTEXT TOKEN REMOVES THE TIME-VARYING SELECTION SIGNAL; THE EFFECT IS LARGEST ON THE NON-STATIONARY SYNTHETIC TASK.

Variant	ISE	Exchange	Synth. (MSE)
TAFS (no context)	**0.249**	**0.468**	0.2188
TAFS (full context)	0.252	0.470	**0.1292**

stationary, and otherwise has no effect, so that TAFS reduces to a strong static combiner. Table VIII reports the variation across constraint regimes on the real data: the affine and unconstrained heads agree to within $\approx 1\%$, while the convex head is $\approx 20\%$ worse because its non-negative weights cannot cancel a divergent base predictor (here the MLP), whereas the affine and unconstrained heads can. We retain the affine head as the default for its accuracy and stability under $\mathbf{1}^\top \mathbf{w}=1$. The transformer is intentionally small (L=2, h=4); on such short series, deeper stacks overfit without lowering the validation error.

4) Discussion: Taken together, the three studies separate two questions that are often confused. The feature-axis transformer closes most of the gap to the oracle, and it outperforms the static transforms on every dataset (Table VI). The context token, which makes the attention time-varying, contributes only when the data are non-stationary in their feature relevance: substantially on the synthetic task, and very little on the near-stationary financial returns. By design, TAFS reduces to a strong static combiner when feature relevance does not vary over time and provides gains when it does.

Two limitations follow directly from the evidence. First, the attention map in (10) is an interpretation aid, not a guarantee. As the synthetic study shows, it reflects which feature tokens the model attends to given the context, so that information already summarized in the context token (here, the lagged target) is down-weighted at the token level; the map should therefore not be read as a formal feature-importance

TABLE VIII
CONSTRAINT-REGIME SPREAD (AGGREGATE COMBINED-PREDICTION MASE OVER THE REAL DATASETS).

Constraint regime	Aggregate MASE
Unconstrained	**0.357**
Affine (default, $\mathbf{1}^\top\mathbf{w}{=}1$)	0.361
Convex ($\mathbf{w}\succeq 0$, $\mathbf{1}^\top\mathbf{w}{=}1$)	0.440

score, in line with the broader caution about treating attention weights as explanations [28], [29]. Second, combination helps only when the base pool is both diverse and individually competitive; the Exchange-Rate results show that when the series is nearly unpredictable, every causal combiner clusters near the best single model and the gap to the oracle remains.

V. CONCLUSION

We introduced TAFS, a context-conditioned feature-axis transformer that jointly performs time-varying feature selection and forecaster combination for sequential regression. TAFS represents each engineered feature as a token and uses a context token derived from recent target history to adapt attention over time. A constrained combination head then maps the resulting summary to weights over a fixed pool of base predictors.

TAFS is most effective when feature relevance and expert suitability vary across regimes. On the controlled synthetic task, TAFS approaches the per-step oracle, and context conditioning reduces error by 41% relative to a context-free variant. Across two real financial datasets, TAFS is the most accurate learned combiner when regime structure can be exploited and remains competitive when that structure is weaker. It also improves on simple averaging, the validation-selected best single predictor, and static-transform baselines.

The theoretical analysis links these architectural choices to trainability, expressiveness, and stability. The TAFS objective is compatible with standard first-order optimization, and common static feature transforms can be viewed as restricted or context-free forms of its feature-mixing mechanism. The simplex-constrained head provides a basic boundedness guarantee by keeping the combined prediction within the convex hull of bounded base forecasts. Together, these findings establish TAFS as an effective framework with theoretical support for adaptive feature selection and forecaster weighting in non-stationary sequential regression. Future work may extend the framework to probabilistic, multi-step, and multi-target forecasting, with combination heads that account for forecast horizons or uncertainty.

REFERENCES

[1] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, "A survey on feature drift adaptation: Definition, benchmark, challenges and future directions," *Journal of Systems and Software*, vol. 127, pp. 278–294, 2017, https://doi.org/10/ggmr52.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008, https://doi.org/10/gpnmtv.

[3] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "Fforma: Feature-based forecast model averaging," *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, 2020, https://doi.org/10/ggjdcq.

[4] V. Cerqueira, L. Torgo, F. Pinto, and C. Soares, "Arbitrage of forecasting experts," *Machine Learning*, vol. 108, no. 6, pp. 913–944, 2019, https://doi.org/10/gjhjgz.

[5] A. Fazla, M. E. Aydin, and S. S. Kozat, "Time-aware and context-sensitive ensemble learning for sequential data," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 5, pp. 2264–2278, 2024, https://doi.org/10.1109/TAI.2023.3319308.

[6] D. Patel and A. Wikner, "Attention-based ensemble pooling for time series forecasting," *arXiv preprint arXiv:2310.16231*, 2023, https://arxiv.org/abs/2310.16231.

[7] S. Ö. Arık and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2021, pp. 6679–6687, https://doi.org/10/gtzvwk.

[8] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 18 932–18 943, https://doi.org/10/hcjp76.

[9] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 2627–2633, https://doi.org/10/gfsj7w.

[10] B. Lim, S. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021, https://doi.org/10/gnp3hv.

[11] 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 *International Conference on Learning Representations (ICLR)*, 2024, pp. 1–26, https://doi.org/10/g9tjx7.

[12] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019, pp. 1–19, https://doi.org/10/gqp96f.

[13] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013, https://doi.org/10/ggmm89.

[14] L. Bottou, F. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018, https://doi.org/10/gdhpjv.

[15] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of adam-type algorithms for non-convex optimization," in *International Conference on Learning Representations (ICLR)*, 2019, https://doi.org/10/hb9grg.

[16] A. Défossez, L. Bottou, F. Bach, and N. Usunier, "A simple convergence proof of adam and adagrad," *Transactions on Machine Learning Research*, 2022, https://doi.org/10/hb9grh.

[17] O. Akbilgic, H. Bozdogan, and M. E. Balaban, "A novel hybrid rbf neural networks model as a forecaster," *Statistics and Computing*, vol. 24, pp. 365–375, 2014, https://doi.org/10/rdq4.

[18] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2018, pp. 95–104, https://doi.org/10/ghfs5d.

[19] G. Ang and E.-P. Lim, "Temporal implicit multimodal networks for investment and risk management," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, pp. 38:1–38:25, 2024, https://doi.org/10/hcjp7t.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 3146–3154, https://papers.nips.cc/paper/6907-lightgbm.

[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794, https://doi.org/10/gdp84q.

[22] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 6638–6648, https://doi.org/10/gt677m.

[23] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, https://doi.org/10/d8zjwq.

[24] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 58, no. 1, pp. 267–288, 1996, https://doi.org/10/gfn45m.

[25] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, https://doi.org/10/cm453n.

[26] 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/dqmdbj.

[27] V. Cerqueira, L. Torgo, and I. Mozetič, "Evaluating time series forecasting models: An empirical study on performance estimation methods," *Machine Learning*, vol. 109, no. 11, pp. 1997–2028, 2020, https://doi.org/10/ghknw7.

[28] S. Jain and B. C. Wallace, "Attention is not explanation," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019, pp. 3543–3556, https://doi.org/10/gf8c7w.

[29] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 11–20, https://doi.org/10/ggcgr9.