MoHETS: Long-term Time Series Forecasting with Mixture-of-Heterogeneous-Experts

Anonymous authors
Paper under double-blind review

Abstract

Real-world multivariate time series exhibit heterogeneous temporal behavior, including recurring structure, local variation, and non-stationary regimes, making long-horizon forecasting challenging. Sparse Mixture-of-Experts (MoE) approaches improve scalability and specialization, but they typically rely on homogeneous MLP experts that poorly capture the diverse temporal dynamics of time series data. We introduce MoHETS, an encoder-only Transformer that integrates sparse Mixture-of-Heterogeneous-Experts (MoHE) layers. MoHE combines an always-active shared depthwise-convolution path for sequence-level continuity with routed Fourier-based experts applied independently to latent patch tokens. The resulting layer is heterogeneous along three axes: operator class, receptive field, and activation role. MoHETS further improves robustness to non-stationary dynamics by incorporating exogenous information through a cross-attention pathway. We also replace parameter-heavy linear projection heads with a lightweight convolutional decoder, thus improving parameter efficiency and training stability. Finally, a single trained model autoregressively produces forecasts over arbitrary horizons. We evaluated MoHETS on seven established multivariate benchmarks across multiple horizons and consistently achieved superior performance, indicating that structural heterogeneity is important for long-term forecasting.

1 Introduction

Time series forecasting is a critical task for decision-making in a wide range of domains, including energy management Lago et al. (2021), financial planning Nie et al. (2024), healthcare Lutsker et al. (2026), and climate analysis Zhang et al. (2023). However, accurately predicting future values from historical observations is challenging, as real-world time series data often present complex temporal dynamics, long-range dependencies, seasonality, trends, non-stationarity, and exogenous influences that lead to varied distributions even within short context windows Wang et al. (2024b); Liu et al. (2024a). Traditional statistical methods, including ARIMA, exponential smoothing, and vector autoregression Hyndman & Athanasopoulos (2018), often struggle to capture nonlinear patterns, multiple seasonalities, or high-dimensional multivariate interactions. Furthermore, these challenges intensify in long-term time series forecasting, where predicting intricate cross-variate dependencies over extended contexts demands models that scale efficiently while maintaining predictive accuracy Liu et al. (2023).

Deep learning approaches have demonstrated remarkable performance in time series forecasting by enabling robust modeling of nonlinear and multivariate patterns. Initially designed for natural language processing (NLP), the Transformer architecture Vaswani et al. (2017) has been successfully extended to computer vision (CV) Dosovitskiy et al. (2020), audio Gong et al. (2021), and time series Zhou et al. (2021); Wen et al. (2023). Transformer models introduce attention mechanisms that adaptively weight historical information to capture long-range dependencies Zhou et al. (2021). However, a fundamental misalignment persists in the context of time series: standard Transformer blocks, designed for the discrete semantics of NLP, apply the same point-wise feed-forward operator to every token. Time series data, conversely, comprise distinct structural components, persistent periodicities, and transient trends that require fundamentally different inductive biases. Applying a uniform architecture to disentangle these heterogeneous patterns often results in inefficient parameter usage and suboptimal fits Dong et al. (2024). Recent improvements have pushed Transformers to mitigate these issues. Patching techniques reduce complexity Nie et al. (2023), integration of

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

exogenous variables allows learning contextual correlations Liu et al. (2024b); Wang et al. (2024b), and sparse designs enable more efficient scaling Shi et al. (2024); Liu et al. (2024a).

Sparse Mixture-of-Experts (MoE) designs increase the conditional capacity, but most time-series MoE implementations retain the homogeneous expert design of large language models (LLMs), where every expert is an identical MLP Shi et al. (2024); Liu et al. (2024a). This design ignores the multi-scale nature of time series, where capturing high-frequency local variations requires different operators from those used to model long-term global dependencies. To address these challenges, we introduce MOHETS, a patch-based encoder-only Transformer with sparse Mixture-of-Heterogeneous-Experts (MoHE) layers designed to combine different operators, receptive fields, and activation roles, thereby enhancing specialization while preserving the benefits of standard MoEs. The complete forecaster also enables covariate integration, improving its robustness to diverse temporal dependencies and patterns. We demonstrate that MOHETS achieves the best overall results over well-known state-of-the-art models in forecasting benchmark experiments. In summary, the main contributions of this research are as follows:

- We introduce MOHETS, an encoder-only Transformer with Mixture-of-Heterogeneous-Experts (MoHE) layers that combine experts differing in operator class, receptive field over the patch sequence, and activation role.
- We propose the MoHE layer, which pairs an always-active depthwise-convolutional expert that mixes information across sequential patch positions with routed Fourier-based experts that apply token-wise periodic-basis transformations.
- We incorporate a multimodal cross-attention module that conditions endogenous patch representations on exogenous covariates. MOHETS enhances time series representations by capturing interactions between endogenous features and exogenous information.

2 Related Work

2.1 Deep Learning for Time Series Forecasting

Deep learning models have significantly advanced time series forecasting, transitioning from MLP-based networks Wang et al. (2024a), recurrent neural networks (RNNs) Salinas et al. (2020), and convolutional neural networks (CNNs) Sen et al. (2019) to Transformer-based architectures Wen et al. (2023). The attention mechanism of Transformers can adaptively weight historical information, making it a natural choice for handling long-term dependencies. Early Transformer models, such as Informer Zhou et al. (2021), introduced the ProbSparse self-attention to address the quadratic complexity of standard Transformers, while Autoformer Wu et al. (2021b) leveraged auto-correlation to discover period-based dependencies at the subseries level. Recently, PatchTST Nie et al. (2023) further improved efficiency by applying channel-independent processing and segmenting time series into subseries-level patches, reducing computational overhead while preserving local semantics and allowing the model to attend to longer context windows. iTransformer Liu et al. (2023) inverts the input dimensions to apply attention across variates rather than time, prioritizing multivariate correlations over temporal dependencies. However, these architectures retain homogeneous point-wise transformations.

2.2 Forecasting with Covariates

Transformer-based models have increasingly introduced covariates to improve contextual understanding. The Temporal Fusion Transformer (TFT) Lim et al. (2021) employed variable selection networks and entity embeddings to dynamically weigh covariates, while Timer-XL Liu et al. (2025) supports covariate-informed forecasting in a decoder-only patched architecture. Incorporating such data requires careful handling to address issues such as missing values or temporal misalignment. In this context, TimeXer Wang et al. (2024b) separates patch-wise endogenous modeling from a variate-wise cross-attention module for exogenous representations, thus mitigating issues related to partial observability and temporal misalignment. Exogenous covariates can provide valuable external information to enhance robustness to non-stationarity and improve forecasting accuracy. Most of these methods append covariates as auxiliary tokens or concatenate them as additional input

channels Lim et al. (2021); Das et al. (2023), which conflates variables that are forecast targets with variables that serve only as context.

2.3 Sparse Mixture-of-Experts (MoE)

Model sparsification has received considerable attention in the context of NLP and CV for efficient handling of diverse patterns Fedus et al. (2022); Jiang et al. (2024); Dai et al. (2024); Riquelme et al. (2021), but has received relatively less attention in time-series research, with few relevant works implementing sparse approaches Shi et al. (2024). While MoLE Ni et al. (2024) explores linear forecasting heads, combining several instances of the same linear operator, FreqMoE Liu (2025) explicitly decomposes the raw series into frequency bands processed by specialized experts. Recent works like Time-MoE Shi et al. (2024) and Moirai-MoE Liu et al. (2024a) adapt sparse MoE to decoder-only Transformers. However, these models adhere to the NLP standard and rely on MLP-based expert designs. Crucially, the direct adaptation of NLP-centric MoEs overlooks the heterogeneous-signal nature of time series. Standard MLP experts lack the inductive bias to efficiently separate global trends from local periodicities, a task for which specialized operators such as convolutions and Fourier operations have been shown to outperform MLPs Dong et al. (2024). We bridge this gap with MoHETS, replacing homogeneous MLPs with MoHE layers that assign architecturally distinct operators to the specific temporal components they are best suited to model.

3 Methodology

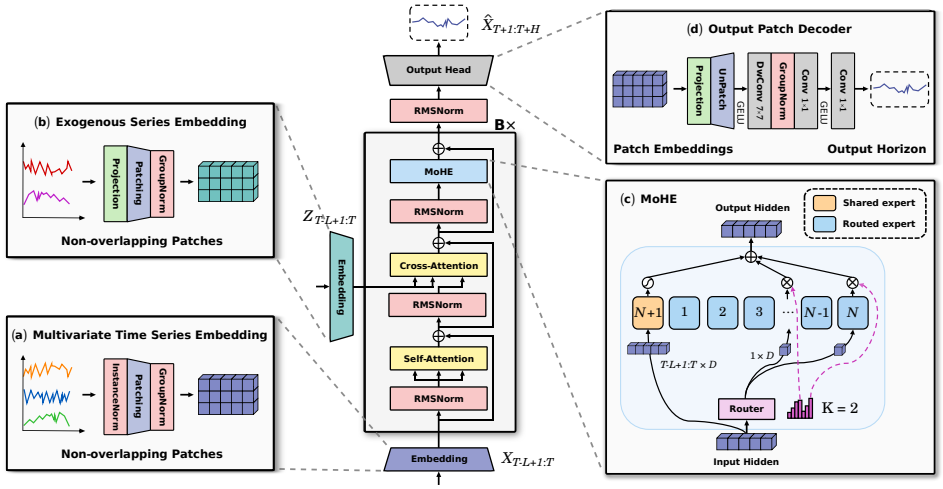

Figure 1: Architecture of MoHETS, an encoder-only transformer for multivariate time series forecasting. (a) The input embedding module splits time channels into sequences of channel-independent patch embeddings. (b) The exogenous embedding module projects, fuses, and patches covariates with the input series to produce aligned exogenous patch embeddings. (c) Mixture-of-Heterogeneous-Experts (MoHE), where a shared depthwise-convolution expert maintains sequence continuity and routed Fourier experts apply token-wise periodic and non-periodic transformations. (d) The patch decoder head projects final embeddings to forecasting horizons.

Problem Statement. Let $\mathbf{X} \in \mathbb{R}^{D \times T}$ denote a set of multivariate time series with D variates (or channels) and T time steps, where each $\mathbf{x}_t = [x_t^1, x_t^2, \ldots, x_t^D]^\top \in \mathbb{R}^D$ represents the observations across all variates at time t. Given a look-back window of length L, the objective is to estimate the next H time steps (the forecast horizon), which yields the forecast of $\hat{\mathbf{X}}_{T+1:T+H} \in \mathbb{R}^{D \times H}$, conditioned on the historical sequence $\mathbf{X}_{T-L+1:T} \in \mathbb{R}^{D \times L}$. The input sequence is processed by a learnable embedding module and projected to the latent dimension d_{model} (detailed in Section 3.2). Next, the input embeddings are forwarded to the Transformer backbone. A Transformer model is constructed by stacking B Transformer blocks, where each block can be represented as follows:

$$\mathbf{u}_t^b = \text{Attn}(\text{Norm}(\mathbf{h}_t^{b-1})) + \mathbf{h}_t^{b-1}, \tag{1}$$

$$\mathbf{h}_t^b = \text{FFN}(\text{Norm}(\mathbf{u}_t^b)) + \mathbf{u}_t^b, \tag{2}$$

where $\mathrm{Attn}(\cdot)$ denotes the self-attention module, $\mathrm{Norm}(\cdot)$ are normalization modules, $\mathrm{FFN}(\cdot)$ denotes the Feed-Forward Network, and $b \in \{0, \ldots, B-1\}$ denotes the b-th Transformer block Vaswani et al. (2017). Standard Transformers rely on dense computations, where a single shared FFN processes every token, effectively forcing a "one-size-fits-all" transformation. In contrast, sparse MoE layers enable conditional computation. To introduce sparsity and enable parameter scaling while keeping computational costs low, an emerging practice is to replace FFN modules in a Transformer with MoE layers. An MoE layer consists of several sparsely activated expert networks, where each expert is structurally identical to a standard FFN Shi et al. (2024); Dai et al. (2024). Then, each individual token can be routed through a gating mechanism to one or more selected experts Fedus et al. (2022); Lepikhin et al. (2021) as follows:

$$\mathbf{h}_t^b = \mathrm{MoE}^b(\mathrm{Norm}(\mathbf{u}_t^b)) + \mathbf{u}_t^b, \quad \text{with} \tag{3}$$

$$\mathrm{MoE}^b(\mathrm{Norm}(\mathbf{u}_t^b)) = \sum_{i=1}^{N} g_{i,t}^b \, \mathrm{FFN}_i^b(\mathrm{Norm}(\mathbf{u}_t^b)), \tag{4}$$

$$s_{i,t}^b = \mathrm{Softmax}_i(\mathbf{W}_g^b(\mathrm{Norm}(\mathbf{u}_t^b))), \tag{5}$$

$$g_{i,t}^b = \begin{cases} s_{i,t}^b, & s_{i,t}^b \in \mathrm{Topk}(\{s_{j,t}^b | 1 \le j \le N\}, K), \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where N represents the number of FFN experts, $g_{i,t}$ is the gate value for the i-th expert, $\mathrm{Topk}(\cdot, K)$ is the set of K highest affinity scores between the t-th token and all experts, $s_{i,t}$ is the point-to-expert affinity score computed by applying a $\mathrm{Softmax}$ to the gate logits $\mathbf{W}_g^b \in \mathbb{R}^{N \times d_{\text{model}}}$, a learnable linear projection Shazeer et al. (2017). Therefore, each token will be forwarded only to K of the N experts, allowing the activated experts to specialize in different time patterns and ensuring computational efficiency Liu et al. (2024a); Dai et al. (2024).

3.1 MoHETS Architecture

Our proposed MoHETS, illustrated in Figure 1, adopts an encoder-only Transformer backbone that leverages recent advances in stability optimizations from large-scale language models Grattafiori et al. (2024); Dai et al. (2024). In forecasting, the relative distance between time steps (e.g., "24 hours ago") carries more predictive signal than absolute indices Erturk et al. (2025). Consequently, we replace standard additive positional encodings with Rotary Position Embeddings (RoPE) Su et al. (2024). By injecting relative position information directly into the query-key interaction, RoPE supports rolling context windows without a learned absolute-position table. Finally, to better separate the modeling of sequence-level continuity from token-wise oscillatory structure, we replace dense FFNs with our MoHE layers. This design enables conditional computation by dynamically routing patches to the operators best suited to their heterogeneous characteristics.

3.2 Input Embedding

To mitigate distribution shift issues in non-stationary time series, we apply instance normalization to the input sequence Liu et al. (2022b), which normalizes each input variate by its instance mean and variance before patching and denormalizes the output predictions to restore the original scale. Standard Transformers process time-series sequences as time-point tokens, incurring quadratic complexity in the look-back length L. To reduce the sequence length processed by attention, we divide each variate into non-overlapping patches Nie et al. (2023), treating each patch as a semantic token Liu et al. (2024a). Specifically, for a patch length P, the look-back window $\mathbf{X}_{T-L+1:T}$ is embedded into $S = L/P$ patches $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_S\}$, with $\mathbf{p}_i \in \mathbb{R}^{D \times P}$, decreasing the quadratic attention complexity to $O(S^2)$ while aggregating local noise into robust feature vectors. Our patch embedding process is illustrated in Figure 1a and defined as:

$$\begin{aligned} \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_S\} &= \mathrm{Patchify}(\mathbf{x}), \\ \mathbf{h}_p^0 &= \mathrm{PatchEmbed}(\mathrm{GNorm}(\mathbf{p})), \end{aligned} \tag{7}$$

where $\mathbf{h}_p^0 \in \mathbb{R}^{d_{\text{model}}}$ is the embedded representation of patch p and GNorm denotes a GroupNorm with one group applied over the embedding dimension Wu & He (2018). We use GNorm to compute

normalization statistics only at the patching and projection boundaries, while $\mathrm{RMSNorm}$ is used at every Transformer sub-layer. Furthermore, the channel independence approach Nie et al. (2023) is used to process each variate of a multivariate input as a univariate series, allowing MOHETS to operate in any-variate forecasting tasks.

3.3 SELF-ATTENTION

To capture temporal dependencies across patch embeddings, we implement multi-head self-attention (see Equation 1) optimized for efficiency and extrapolation. We further reduce the inference memory footprint by implementing grouped-query attention (GQA) Ainslie et al. (2023). By sharing a single key-value head across multiple query heads, GQA maintains the expressivity of multi-head attention while significantly reducing the memory overhead needed for autoregressive rollout. Following Shi et al. (2024), we adopt a bias-free architecture for all layers except the QKV projections. Retaining QKV biases has been shown to preserve length extrapolation capabilities by maintaining shift-invariance in the attention scores Chowdhery et al. (2023). In addition, we apply channel-independent attention per variate Nie et al. (2023). This mechanism forces the model to learn universal temporal dynamics (e.g., seasonality) that generalize across different variates, preventing overfitting to spurious cross-variate correlations. Details about our implementation of multi-head attention are in Appendix A.1.

3.4 MULTIMODAL CROSS-ATTENTION

Let $\mathbf{Z}_{T-L+1:T+H} \in \mathbb{R}^{C\times(L+H)}$ denote an additional sequence with C covariates, assumed to be known over the forecast horizon Wang et al. (2024b). We design an embedding module that incorporates exogenous covariate information into endogenous time series. The module operates in two stages. First, we use linear layers to project both endogenous and covariate sequences to the model dimension d_{model}:

$$\begin{aligned} \mathbf{W}_t^X &= \mathrm{Linear}_X(\mathbf{x}_t), \quad \mathbf{x}_t \in \mathbb{R}^D, \quad \mathbf{W}_t^X \in \mathbb{R}^{d_{\text{model}}}, \\ \mathbf{W}_t^Z &= \mathrm{Linear}_Z(\mathbf{z}_t), \quad \mathbf{z}_t \in \mathbb{R}^C, \quad \mathbf{W}_t^Z \in \mathbb{R}^{d_{\text{model}}}, \end{aligned} \tag{8}$$

These projections are fused via concatenation and a subsequent projection layer: $\mathbf{w}_t^M = \mathrm{Linear}_{\text{fuse}}([\mathbf{W}_t^X; \mathbf{W}_t^Z])$, creating a covariate-enriched latent representation aligned with the endogenous dimension D. Concatenation followed by linear projection outperforms alternatives such as element-wise addition or concatenation without a fusion projection Bao et al. (2023). As illustrated in Figure 1b, the fused sequence $\mathbf{W}_t^M \in \mathbb{R}^{D\times L}$ is patched with the same patch length $P$by a dedicated $\mathrm{PatchEmbed}$ layer, producing $S = L/P$ aligned non-overlapping token embeddings that serve as keys and values for the cross-attention mechanism. By adopting this approach, we ensure that exogenous data is aggregated into endogenous data regardless of the difference between D and C.

In the cross-attention module, the self-attention output (Section 3.3) serves as the query, while the exogenous patch embeddings feed the key and value projections Wang et al. (2024b). This design allows the model to dynamically retrieve external context, such as holiday effects, conditioned on the current state. Endogenous variables and covariates play asymmetric roles: the former are the signals being represented and forecast, the latter provide context and are usually not prediction targets. Treating both as equivalent input channels obscures this distinction. Cross-attention instead preserves a direct endogenous residual pathway and allows the relevance of the conditioning stream to vary across patches and layers. The conditioning mechanism, therefore, injects forecasting-specific inductive biases rather than a generic feature-mixing scheme. Because the module is applied residually (Equation 9), the endogenous representation remains directly available when covariates are absent or partially observed. The above process can be formalized as

$$\mathbf{v}_p^b = \mathrm{CrossAttn}(\mathrm{RMSNorm}(\mathbf{u}_p^b), \mathbf{h}_m) + \mathbf{u}_p^b. \tag{9}$$

The conditioning embedding is not strictly channel-independent because its projection receives the complete D-dimensional endogenous vector before returning to D conditioned channels. Since covariates $\mathbf{Z}$ are known for the future horizon $t = T+1, \ldots, T+H$, the fusion process is repeated during the autoregressive rollout, allowing the model to anticipate future external temporal information before it appears in the series.

3.5 MIXTURE-OF-HETEROGENEOUS-EXPERTS (MOHE)

Conventional sparse MoEs usually replicate the same point-wise MLP architecture across all experts, assuming that all tokens require the same processing logic. We argue that this approach is suboptimal for time series, which are a superposition of global trends and local periodicities. We therefore propose the Mixture-of-Heterogeneous-Experts (MoHE) design, in which the experts within a layer differ structurally rather than only in their learned parameters. Each MoHE layer is heterogeneous along three axes simultaneously: in operator class, pairing a sequence-mixing depthwise convolution with Fourier-analysis transformations; in receptive field, since the convolutional path spans neighboring patch positions while the routed paths are point-wise with respect to the patch sequence; and in activation role, since one path is always active while the others are sparsely selected and separately gated. The always-active, shared expert operates at the sequence level and serves as an inductive prior to capture and consolidate common knowledge, thereby supporting the specialization of the routed experts Dai et al. (2024); Shi et al. (2024).

Specifically, each expert, shared or routed, comprises two layers that project time patches from the model dimension d_{model} to an intermediate dimension d_{ff} and back to d_{model}, with a factor of $d_{\text{ff}} = 2\times d_{\text{model}}$, GELU activation Hendrycks & Gimpel (2016), and dropout Srivastava et al. (2014). The shared expert is a depthwise separable convolution (DwConvFFN) that acts as a sequence-level expert: it slides along the sequence dimension to capture continuous trends and maintain temporal coherence across patches. Conversely, the routed experts are token-wise Fourier-based networks (FA-FFN) that introduce a learned periodic-basis inductive bias in the latent representation space; representing such oscillatory structure is a task at which standard MLPs often struggle owing to their spectral bias Rahaman et al. (2019). Specifically, an FA-FFN replaces general-purpose linear layers with FAN modules Dong et al. (2024), which are designed to leverage the strengths of Fourier series transformations to model periodic patterns from time series. Therefore, a MoHE layer combines one DwConvFFN shared expert and N FA-FFN routed experts:

$$\begin{aligned} \bar{\mathbf{v}}_p^b &= \mathrm{RMSNorm}(\mathbf{v}_p^b), \\ \mathrm{MoHE}^b(\bar{\mathbf{v}}_p^b) &= g_{N+1,p}^b\,\mathrm{DwConvFFN}_{N+1}^b(\bar{\mathbf{v}}_p^b) + \sum_{i=1}^{N}(g_{i,p}^b\,\mathrm{FA\text{-}FFN}_i^b(\bar{\mathbf{v}}_p^b)), \end{aligned} \tag{10}$$

where $g_{N+1,p}$ denotes an independently learned Sigmoid gate, modulating the shared expert contribution Shi et al. (2024), and $g_{i,p}$ represents the router gate value defined in Equation 6 (dotted lines in Figure 1c). The heterogeneity introduced by the MoHE design is inspired by the trend-seasonality decomposition commonly used in statistical forecasting Wu et al. (2021b). However, the architecture does not supervise the router with trend, frequency, or periodicity labels. Fourier-based experts also contain a non-periodic nonlinear branch and are not restricted to purely periodic functions. Sparse architectures dynamically activate different experts to handle heterogeneous data patterns, with each expert specializing in learning different patterns Shi et al. (2024). This MoHE design improves specialization relative to homogeneous MoEs, thereby enhancing generalization and forecasting accuracy across heterogeneous temporal patterns. The detailed FA-FFN transformation is provided in Appendix A.1.

3.6 OUTPUT PATCH DECODER

To map the Transformer's output from patches to forecast time points, most of the state-of-the-art time series models have implemented linear-based projection heads Nie et al. (2023); Liu et al. (2023); Wang et al. (2024b); Shi et al. (2024). A common forecasting head flattens the S patch embeddings and applies a dense projection, requiring $Sd_{\text{model}}L$ weights. An individual per-channel variant would scale to $O(Sd_{\text{model}}LD)$, leading to parameter explosion and overfitting when the number of channels D increases. Instead, MOHETS decodes the patch representations using a lightweight convolutional decoder (see Figure 1d). After the final Transformer block, a transposed convolution first expands each patch token to P time positions, after which a depthwise temporal convolution followed by two point-wise convolutions projects the features from the latent dimension d_{model} to a single scalar per channel and time point. For fixed P and d_{model}, the decoder parameter count is independent of S, L, and D. This design imposes a locality-based inductive bias, ensuring that output generation relies on the semantic vectors of each patch rather than on a dense global matrix. So, we provide a lightweight channel-independent solution that handles any-variate forecasting and mitigates the instability of heavy linear heads, helping MOHETS achieve superior accuracy. Refer to Appendix A.1 for details.

3.7 Loss Functions

Time series data often contain transient outliers and extreme spikes that can destabilize training Wen et al. (2019), especially for sparse models where gradients are routed to specific experts Fedus et al. (2022). Standard MSE amplifies these outliers quadratically. Therefore, we adopt the Huber loss Huber (1992) as our main prediction loss ($\mathcal{L}_{\text{pred}}$), which behaves quadratically for small errors and linearly for large errors, providing a robust gradient signal that prevents expert collapse due to outlier-driven gradients.

However, focusing only on prediction-error optimization often leads to stability and convergence challenges due to load imbalance when using MoE architectures. Specifically, the sparse routing mechanism introduces the risk of routing collapse Shazeer et al. (2017), in which the model converges to a trivial state by selecting a single expert, limiting the opportunities for other experts to receive sufficient training Dai et al. (2024). To prevent routing collapse, we use an auxiliary load-balancing loss ($\mathcal{L}_{\text{aux}}$) that combines the hard fraction of selected routing slots assigned to each expert with that expert's mean soft routing probability Lepikhin et al. (2021); Fedus et al. (2022); Shi et al. (2024). The objective penalizes simultaneous concentration of hard assignments and soft probability mass on the same experts, thereby discouraging overloaded experts. We detail the prediction and balance losses in Appendix A.2, and expert usage diagnostics in Appendix D.

3.8 Training Objective and Forecasting

Given an input window $X_{T-L+1:T}$, MOHETS produces a length-L sequence corresponding to the target window shifted forward by the next H_o time steps, where H_o is the output resolution. The training objective combines a dense prediction loss over the shifted window with the auxiliary balance loss, ensuring both accuracy and expert balance:

$$\mathcal{L} = \mathcal{L}_{\text{pred}}\Big(X_{T-L+1+H_o:T+H_o},\ \hat{X}_{T-L+1+H_o:T+H_o}\Big) + \alpha\mathcal{L}_{\text{aux}}, \tag{11}$$

where α is the expert balance factor. This formulation has the same form as the token-level objective used to train decoder-only forecasters Liu et al. (2024b); Shi et al. (2024), but it supervises every patch position of a bidirectional encoder. The patch length is uniformly set as P from the input embedding to the output patch decoder. At inference, only $\hat{X}_{T+1:T+H_o}$ is retained and appended to the input buffer, with MOHETS supporting flexible output resolutions H_o.

MOHETS is trained as a one-for-all forecaster Liu et al. (2025). For a requested horizon H, inference performs autoregressive rolling forecasting: the newly generated chunk (the next H_o time steps) is appended to the input buffer, and the final concatenated prediction is truncated to exactly H points when H is not divisible by H_o. This horizon flexibility allows a single trained model to generate arbitrary horizons (e.g., $\{96, 192, 336, 720\}$) by controlling the total generation length at inference time without training a separate output head for every H. Appendix C.4 provides further details.

3.9 Model Settings and Training Details

MOHETS and all experiments are implemented in PyTorch Paszke et al. (2019). Training is performed on a single NVIDIA A100 (80GB) GPU. To maximize throughput without sacrificing numerical stability, we utilized TensorFloat-32 (TF32) precision, which provides the dynamic range of FP32 with the matrix-multiplication speed of FP16. We experiment with the number of Transformer blocks searched over $B \in \{4, 6, 8\}$, the dimension of representations d_{model} from $\{64, 128, 256, 384\}$, the patch length as $P \in \{8, 12, 16\}$, and the output resolution as $H_o \in \{16, 24, 32\}$ time points. Following the results achieved by Dai et al. (2024); Liu et al. (2024a); Shi et al. (2024), we set 8 as the number of experts per MoHE layer with $K = 2$, which is an optimal choice to balance performance and computational efficiency Shi et al. (2024). Refer to Appendix B.1 for more optimization details.

Furthermore, we feed the cross-attention modules of MOHETS only with calendar information derived from the timestamp of each instance, splitting the calendar components and scaling them to continuous linear frequency values Alexandrov et al. (2020). The same construction is applied to all benchmarks.

4 MAIN RESULTS

We conduct in-domain experiments to evaluate the performance and efficiency of MOHETS in long-term multivariate point forecasting. Our experiments include 7 established benchmark datasets spanning a wide range of real-world domains, with different dimensionalities and sampling resolutions. We follow the chronological train-validation-test split protocol defined by Wu et al. (2023), which is commonly used in the long-term forecasting literature to prevent data leakage. Each benchmark is evaluated over four long-term prediction horizons, which are $H \in \{96, 192, 336, 720\}$. As baselines, we select 15 models that represent the state of the art in long-term forecasting. Comparisons align datasets, forecast lengths, metrics, and standard splits. A detailed summary of each benchmark dataset is provided in Appendix B.2, while the baseline models are presented in Appendix B.3. As evaluation metrics, we adopt the mean squared error (MSE) and the mean absolute error (MAE), detailed in Appendix B.4.

4.1 MULTIVARIATE TIME SERIES FORECASTING

Table 1: Results of long-term multivariate forecasting. Full-shot results are obtained from Kraus et al. (2026); Shi et al. (2024); Wang et al. (2024b); Han et al. (2024). **Bold red**: the best, underlined blue: the second best. Results are averaged from all prediction horizons $H = \{96, 192, 336, 720\}$. 1st Count represents the number of wins achieved by a model.

Models	MOHETS		xLSTM-M		S-Mamba		SOFTS		TimeXer		iTransformer		TimeMixer		TimesNet		PatchTST		TiDE		DLinear	
Metrics ↓	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.377**	**0.411**	0.397	0.420	0.455	0.450	0.449	0.442	0.437	0.437	0.454	0.447	0.447	0.440	0.458	0.450	0.469	0.454	0.540	0.507	0.455	0.451
ETTh2	0.343	**0.379**	**0.340**	0.382	0.381	0.405	0.373	0.400	0.367	0.396	0.383	0.406	0.364	0.395	0.414	0.427	0.387	0.407	0.611	0.549	0.558	0.515
ETTm1	**0.329**	**0.365**	0.339	0.366	0.398	0.405	0.393	0.403	0.382	0.397	0.407	0.410	0.381	0.395	0.400	0.405	0.387	0.400	0.419	0.419	0.403	0.406
ETTm2	0.255	0.308	**0.248**	**0.307**	0.288	0.332	0.287	0.330	0.274	0.322	0.288	0.332	0.275	0.323	0.291	0.332	0.281	0.326	0.358	0.403	0.350	0.400
Weather	**0.216**	**0.248**	0.219	0.250	0.251	0.276	0.255	0.278	0.241	0.271	0.258	0.278	0.240	0.271	0.259	0.286	0.259	0.281	0.270	0.320	0.265	0.316
ECL	0.157	0.250	**0.153**	**0.245**	0.170	0.265	0.174	0.264	0.171	0.270	0.178	0.270	0.182	0.272	0.192	0.295	0.205	0.290	0.251	0.344	0.212	0.300
Traffic	**0.385**	**0.252**	0.392	0.253	0.414	0.276	0.409	0.267	0.466	0.287	0.428	0.282	0.484	0.297	0.620	0.336	0.481	0.304	0.760	0.473	0.625	0.383
Average	**0.295**	**0.316**	0.298	0.318	0.337	0.344	0.334	0.341	0.334	0.340	0.342	0.346	0.339	0.342	0.376	0.362	0.353	0.352	0.458	0.431	0.410	0.396
1st Count	11		5		0		0		0		0		0		0		0		0		0	

Table 1 summarizes the long-term multivariate forecasting results. Within this literature-reported comparison, MOHETS achieves the best overall results, with the lowest overall average MSE and MAE and the most first places (11 of 16), ahead of the strongest baseline, xLSTM-M, which takes the other five (MSE on ETTh2 and both metrics on ETTm2 and ECL). In particular, MOHETS shows its largest margin on ETTh1, a dataset with strong seasonality, demonstrating the MoHE benefit of combining sequence-mixing convolutional and routed periodic-basis experts. When analyzing performance averaged over the horizons $\{96, 192, 336, 720\}$, MOHETS reduces the average MSE relative to xLSTM-M by 5.0% on ETTh1, 2.9% on ETTm1, and 1.8% on Traffic, and by 11.7% relative to TimeXer and SOFTS on the overall average, underscoring the effectiveness of the MoHE architecture and of the covariate injection support. The complete results and comparisons with other models are in Appendix C.

4.2 ABLATION STUDY

We validate the effectiveness of our designs in MOHETS by conducting detailed ablation studies on key architectural components using the experimental benchmarks, including various compositions of our MoHE approach, the incorporation of exogenous covariates, normalization, and the final projection head. We fix $d_{\text{model}} = 128$ (see Section 3.9) and limit the maximum number of training epochs to 20 in each ablation experiment to save computational time. All results are averaged over the horizons $H \in \{96, 192, 336, 720\}$, with lower MSE or MAE values indicating better performance.

Expert Composition. We ablate six expert compositions under the same encoder and training protocol. Table 2 shows that our proposed mixture (DwConvFFN + FA-FFNs) achieves improvements in average MSE of 4.0% on ETTh1 and 18.5% on Traffic over the all-MLP version (the standard MoE), and reduces the seven-dataset average MSE by 9.7% and MAE by 5.9%. MOHETS improves MSE on five out of seven datasets and MAE on all seven datasets relative to the all-MLP variant. Moreover, a pattern in the aggregate values supports our design – every composition using

Fourier-routed experts outperforms every composition using MLP-routed experts, irrespective of the shared expert, so the routed operator is the dominant factor. These results support the utility of the MoHE design.

Table 2: Ablation study with different expert compositions. Conv denotes a two-layers point-wise convolutional FFN. The best results are in **bold** and the second best are underlined.

Dataset	ETTh1		ETTh2		ETTm1		ETTm2		Weather		ECL		Traffic		Average ↓	
Metrics (Avg.) ↓	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MoHETS	**0.383**	**0.415**	0.362	**0.389**	**0.338**	**0.366**	0.270	0.313	0.244	0.265	**0.164**	**0.254**	**0.406**	**0.276**	**0.310**	**0.325**
MLP	0.399	0.420	0.361	0.392	0.348	0.373	0.264	0.314	0.262	0.272	0.267	0.318	0.498	0.331	0.343	0.346
FA	0.391	0.421	0.364	0.390	0.352	0.377	0.268	0.314	**0.230**	**0.262**	0.174	0.269	0.428	0.290	0.315	0.332
Conv + MLP	0.398	0.420	**0.357**	**0.389**	0.353	0.378	**0.260**	**0.312**	0.234	0.265	0.197	0.293	0.435	0.284	0.319	0.334
Conv + FA	0.402	0.425	0.367	0.395	0.356	0.376	0.269	**0.312**	0.237	0.266	**0.164**	0.255	0.413	**0.276**	0.315	0.329
DwConv + MLP	0.401	0.424	0.372	0.404	0.368	0.375	0.279	0.325	0.368	0.283	0.239	0.325	0.420	0.284	0.350	0.346

Normalization Strategies. We evaluate homogeneous alternatives using LayerNorm Ba et al. (2016) or RMSNorm throughout the model, together with the mixed strategy used by MoHETS: one-group GroupNorm in the input patch embedding and output decoder, and RMSNorm in the Transformer blocks Wu et al. (2021a). Table 3 shows that the mixed approach achieves significantly lower MSE and MAE values than homogeneous normalizations on most benchmarks, with 9.9% and 12.1% improvements in MSE over all-LayerNorm and all-RMSNorm, respectively, on ETTh1. However, the all-RMSNorm version achieves a strong MSE on Weather.

Table 3: Ablation study using different normalization layers, projection head, and with no covariate injection. The best results are in **bold** and the second best are underlined.

Dataset	ETTh1		ETTh2		ETTm1		ETTm2		Weather		ECL		Traffic		Average ↓	
Metrics (Avg.) ↓	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MoHETS	**0.383**	**0.415**	0.362	**0.389**	**0.338**	**0.366**	0.270	**0.313**	0.244	0.265	**0.164**	**0.254**	**0.406**	0.276	**0.310**	**0.325**
LayerNorm	0.425	0.437	0.458	0.442	0.362	0.379	0.280	0.326	0.281	0.284	**0.164**	0.256	0.413	**0.270**	0.340	0.342
RMSNorm	0.436	0.444	0.439	0.436	0.358	0.377	0.278	0.322	**0.229**	**0.259**	0.188	0.274	0.430	0.288	0.337	0.343
MLP-based head	0.457	0.451	**0.354**	0.392	0.351	0.380	**0.259**	0.314	0.237	0.272	0.177	0.276	0.418	0.301	0.322	0.341
w/o exogenous covs.	0.418	0.429	0.381	0.398	0.357	0.375	0.283	0.323	0.247	0.269	0.174	0.264	0.409	0.273	0.324	0.333

Projection Heads. We compare the convolutional decoder with traditional MLP-based heads. The convolutional design outperforms the MLP-based head by reasonable margins in most tests (Table 3), supporting that the locality inductive bias of convolutions is superior to dense linear projections for decoding. By respecting the patch boundary structure during upsampling, the convolutional head prevents overfitting on the output layer. Figure 4 compares the optimization trajectories of MoHETS when using an MLP-based head and our convolutional approach.

Covariate Conditioning. Incorporating only calendar covariates through our cross-attention approach improves performance, particularly on datasets with lower dimensionality such as ETT. Averaged over the seven benchmarks, covariate conditioning reduces MSE by 4.5% and MAE by 2.3% (Table 3). The effect is consistent, improving MSE on all seven datasets, but smaller in magnitude than that of the expert composition reported above. We observe a slight reduction in the advantage of injecting calendar information as data dimensionality increases, as in Traffic. Note that we significantly improved robustness to non-stationarity in coarse datasets (e.g., ETTh1 and ETTh2), even scoping this experiment only to timestamp-derived calendar conditioning.

5 Conclusion

By combining an always-active sequence-mixing expert with sparsely selected token-wise Fourier experts, our MoHE design introduces structural heterogeneity in operator class, receptive field, and activation role — a form of heterogeneity unavailable in homogeneous MLP-based MoE layers. Thus, MoHETS establishes a multi-scale receptive field: global dependencies are resolved by attention, sequence-level continuity is maintained by the shared convolution, and token-wise oscillatory structure is captured by the routed experts. MoHETS achieved remarkable performance on multiple in-domain forecasting tasks, and extensive ablation experiments support its design choices. Therefore, this work introduces advancements that position MoHETS as a state-of-the-art solution for real-world time series forecasting, providing a scalable and robust framework for diverse temporal applications.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing of this manuscript. We have not used generative AI tools to generate research ideas, formulate claims, or write code, and synthetic data generation is not applicable to this work. We have reviewed all AI-assisted work. Specifically, AI assistance was limited to copy-editing, grammar correction, and structural refinement of human-authored drafts; the authors thoroughly reviewed all suggested changes to ensure the original meaning and scientific accuracy were fully preserved. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several steps to support the reproducibility of our results. The MOHETS architecture and training objective are specified in Section 3, and Appendix A details the attention mechanism, the MoHE layer, the FA-FFN routed experts, the convolutional output decoder, and the prediction and expert-balance losses. Four model versions, their optimization, regularization, and initialization settings, and the hardware and numerical precision used are described in Section 3.9 and Appendix B.1. The dataset-specific configuration behind every reported result, including model size, patch length, output chunk size, learning rate, batch size, and epoch budget, is listed in Table 8. All seven benchmarks are publicly available; their statistics, chronological splits, and the construction of timestamp-derived covariates are described in Appendix B.2. Appendix B further describes the evaluation metrics and the baseline models, whose results are the published values from the cited sources, obtained on the same datasets, splits, horizons, and metrics. Sections 3.8 and 3.9 describe the model-selection protocol, in which configurations are chosen on the validation split, and the test split is reserved for final reporting, and the autoregressive rollout that uses a single checkpoint per dataset produces all four forecast horizons. FLOP and memory measurements are reported in Appendix C.6, and variability across five random seeds is characterized in Appendix C.1. The expert-utilization diagnostics are provided in Appendix D.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Turkmen, and Yuyang Wang. GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A ViT backbone for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22669–22679, 2023.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning (ICML)*, pp. 1691–1703. PMLR, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

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

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning (ICML)*, 2024.

Yihong Dong, Ge Li, Yongding Tao, Xue Jiang, Kechi Zhang, Jia Li, Jinliang Deng, Jing Su, Jun Zhang, and Jingjing Xu. FAN: Fourier analysis networks. *arXiv preprint arXiv:2410.02675*, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eray Erturk, Fahad Kamran, Salar Abbaspourazad, Sean Jewell, Harsh Sharma, Yujie Li, Sinead Williamson, Nicholas J Foti, and Joseph Futoma. Beyond sensor data: Foundation models of behavioral data from wearables improve health predictions. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. PMLR, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Georg Goerg. Forecastable component analysis. In *International Conference on Machine Learning (ICML)*, 2013.

Yuan Gong, Yu-An Chung, and James Glass. AST: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning (ICML)*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pp. 646–661. Springer, 2016.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Maurice Kraus, Felix Divo, Devendra Singh Dhami, and Kristian Kersting. xLSTM-Mixer: Multivariate time series forecasting by mixing via scalar memories. *Advances in Neural Information Processing Systems*, 38:6547–6581, 2026.

Jesus Lago, Grzegorz Marcjasz, Bart De Schutter, and Rafał Weron. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, 293:116983, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 2021.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022a.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024a.

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

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Transformers for time series analysis at scale. In *International Conference on Machine Learning (ICML)*, 2024b.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-context transformers for unified time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022c.

Ziqi Liu. Freqmoe: Enhancing time series forecasting through frequency decomposition mixture of experts. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Guy Lutsker, Gal Sapir, Smadar Shilo, Jordi Merino, Anastasia Godneva, Jerry R. Greenfield, Dorit Samocha-Bonet, Raja Dhir, Francisco Gude, Shie Mannor, Eli Meirom, Eric P. Xing, Gal Chechik, Hagai Rossman, and Eran Segal. A foundation model for continuous glucose monitoring data. *Nature*, 2026.

Ronghao Ni, Zinan Lin, Shuaiqi Wang, and Giulia Fanti. Mixture-of-linear-experts for long-term time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, pp. 4672–4680. PMLR, 2024.

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

Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M Mulvey, H Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges. *arXiv preprint arXiv:2406.11903*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. TFB: Towards comprehensive and fair benchmarking of time series forecasting methods. *arXiv preprint arXiv:2403.20150*, 2024.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Facebook AI Research. fvcore: Lightweight core library for computer vision. `https://github.com/facebookresearch/fvcore`, 2020.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2020.

Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. *arXiv preprint arXiv:2409.16040*, 2024.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024a.

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

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

Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, and Jian Tan. RobustTrend: a huber loss with a combined first and second order difference regularization for time series trend filtering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3856–3862, 2019.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 6778–6786, 2023.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021a.

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

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

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In *International Conference on Machine Learning (ICML)*, pp. 60179–60196, 2024.

Yuchen Zhang, Mingsheng Long, Kaiyuan Chen, Lanxiang Xing, Ronghua Jin, Michael I Jordan, and Jianmin Wang. Skilful nowcasting of extreme precipitation with NowcastNet. *Nature*, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2022.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

A TECHNICAL DETAILS

A.1 ARCHITECTURAL COMPONENTS

Attention. In MOHETS, we implement grouped-query attention (GQA) Ainslie et al. (2023) to optimize self-attention and cross-attention mechanisms, balancing computational efficiency and modeling capacity. Standard multi-head attention (MHA) Vaswani et al. (2017) allocates independent query (Q), key (K), and value (V) projections to each attention head, generating rich contextual information. The downside of MHA is its high memory bandwidth and computational cost, particularly for long sequences and autoregressive inference. Multi-query attention (MQA) Shazeer (2019) was proposed to mitigate attention costs by using a single K and V projection on multiple query heads, drastically reducing the memory footprint. However, this abrupt reduction in the K and V projections can degrade the model's capacity and training stability due to restricted representational expressivity, leading to a reduced performance compared to MHA Ainslie et al. (2023).

GQA addresses the efficiency-capacity trade-off with a more general and flexible formulation of MHA that groups query heads into tunable clusters, each cluster sharing a single K and V projection, thereby reducing memory while preserving performance close to that of MHA Ainslie et al. (2023). GQA has been successfully adopted in large-scale models, demonstrating improved efficiency with minimal loss of quality Grattafiori et al. (2024). For time series forecasting, long horizons require efficient attention. Thus, we adopt a light query grouping factor of 2 in MOHETS, that is, Q-heads $= 2\times$KV-heads (see Figure 2 and Table 4). To enhance the handling of extended look-back windows (L), we also employ FlashAttention-2 Dao (2023). This implementation reformulates the scaled dot-product to reduce memory access costs from quadratic to linear with respect to sequence length, enabling rapid training on long sequences. GQA combined with FlashAttention Dao et al. (2022) reduces memory overhead while maintaining robust modeling capacity.

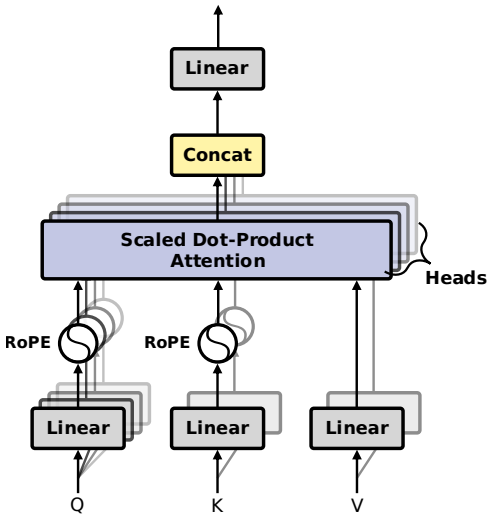

Figure 2: Grouped-query attention (GQA) with Rotary Position Embeddings (RoPE). Single key and value heads are shared for each group of query heads. In MOHETS, we adopt a grouping factor of two query heads for each key/value head, i.e., Q-heads $= 2\times$KV-heads.

MoHE Design. Our Mixture-of-Heterogeneous-Experts approach consists of one always-active (shared) expert designed to capture sequence-level temporal patterns and a routed pathway with multiple experts designed to apply token-wise periodic-basis transformations. During training, the router learns to send similar patches to the same experts. Only the selected routed experts receive task-loss gradients and specialize in processing that type of patch. The routing mechanism operates independently in each MoHE layer along the model, creating a sophisticated network of specialization. For an input patch embedding p, the shared expert is evaluated for the complete ordered patch sequence, and its output is modulated by a learned $\mathrm{Sigmoid}$ gate for every token (Equation 10). In contrast, the routed experts use the Topk $\mathrm{Softmax}$ routing weights (Equations 5 and 6), retaining only the K highest scores to ensure sparsity.

The final output combines the contributions of both shared and Topk experts to capture long-term patterns and local periodicity (see Figure 1c). The MoHE architecture activates only the relevant parts of the model for each time patch, enabling the total routed-expert capacity to grow faster than the expert computation applied to a single token. It should be noted that we did not experiment with convolutional routed experts. A routed convolutional operator would require dispatching contiguous patch groups or constructing per-expert masked sequences.

Fourier-based Networks. Multi-layer perceptron (MLP) networks are widely used in deep learning because of their general-purpose capacity to approximate diverse functions. However, deep neural networks may exhibit a spectral bias toward low-frequency functions of their inputs, which can make oscillatory mappings harder to learn efficiently and to generalize beyond the observed input range Rahaman et al. (2019); Dong et al. (2024). Motivated by the recurring structure commonly encountered in long-horizon forecasting, we replace the standard MLP-FFNs in the routed expert pool of MoHE with a two-layer adaptation of the Fourier Analysis Network (FAN) Dong et al. (2024), which we call an FA-FFN.

Our FA-FFN, illustrated in Figure 3, retains the inverted-bottleneck dimensional structure of a standard Transformer FFN. The first FAN layer expands the token from d_{model} to d_{ff}, dropout is applied to the expanded representation, and the second FAN layer contracts it from d_{ff} back to d_{model}. Unlike the network-level FAN architecture, which uses a final linear layer Dong et al. (2024), our FA-FFN uses a FAN layer for both expansion and contraction transformations.

Let $\mathbf{x}_{n,s} \in \mathbb{R}^{d_{\text{model}}}$ denote the normalized embedding of patch s for batch–variate instance n when it is dispatched to a routed expert. For notational simplicity, we omit the indices n and s below. Each FA-FFN is applied independently to the selected token along its latent feature dimension:

$$\text{FA-FFN}(\mathbf{x}) = \phi_2(\text{Dropout}\,(\phi_1(\mathbf{x}))), \qquad \phi_\ell : \mathbb{R}^{m_{\ell-1}} \to \mathbb{R}^{m_\ell}, \tag{12}$$

where $m_0 = d_{\text{model}}$, $m_1 = d_{\text{ff}}$, and $m_2 = d_{\text{model}}$. For $\ell \in \{1, 2\}$, the corresponding FAN layer is defined as

$$\phi_\ell(\mathbf{z}) = \left[\cos\left(W_{\text{per}}^{(\ell)}\mathbf{z}\right) \,\middle\|\, \sin\left(W_{\text{per}}^{(\ell)}\mathbf{z}\right) \,\middle\|\, \psi\left(W_{\text{np}}^{(\ell)}\mathbf{z}\right)\right], \tag{13}$$

where $W_{\text{per}}^{(\ell)} \in \mathbb{R}^{(m_\ell/4)\times m_{\ell-1}}$ and $W_{\text{np}}^{(\ell)} \in \mathbb{R}^{(m_\ell/2)\times m_{\ell-1}}$ are learned periodic and non-periodic projections, respectively, ψ is a GELU activation function, and $\|$ denotes concatenation along the feature dimension. No bias terms are used in these projections, consistent with the bias-free design described in Section 3.3. The cosine and sine branches each contribute $m_\ell/4$ coordinates, while the non-periodic ψ branch contributes $m_\ell/2$ coordinates. Therefore, the concatenated output width is $m_\ell/4+m_\ell/4+m_\ell/2 = m_\ell$. Consequently, both FAN-layer output dimensions, $m_1 = d_{\text{ff}}$ and $m_2 = d_{\text{model}}$, must be divisible by four.

For a row vector $\mathbf{w}^\top$ of $W_{\text{per}}^{(\ell)}$, the corresponding coordinates $\cos(\mathbf{w}^\top\mathbf{z})$ and $\sin(\mathbf{w}^\top\mathbf{z})$ are periodic with respect to the learned scalar coordinate $\mathbf{w}^\top\mathbf{z}$. The rows of $W_{\text{per}}^{(\ell)}$ therefore determine learned projection directions and oscillation rates in latent representation space. The sine and cosine branches share the same projection, yielding paired features with a common learned argument. A subsequent learned transformation can represent different phases without requiring separate periodic projections. The activation ψ is applied only to the non-periodic branch, providing a complementary MLP-like nonlinear pathway; consequently, the expert is not restricted to representing purely periodic functions.

The periodic feature map is an inductive prior rather than a hard constraint. If $W_{\text{per}}^{(\ell)} = 0$, then the periodic branches become $\cos(W_{\text{per}}^{(\ell)}\mathbf{z}) = 1$ and $\sin(W_{\text{per}}^{(\ell)}\mathbf{z}) = 0$, and all the input-dependent variation produced by that layer is carried by its non-periodic $\psi(W_{\text{np}}\mathbf{z})$ branch (Equation 13).

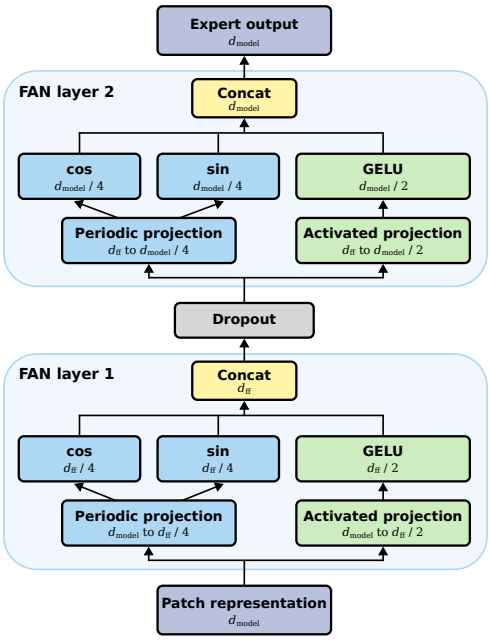

Figure 3: Structure of an FA-FFN routed expert. Each of the two FAN layers splits its output into three concatenated branches: two periodic branches of widths $m_\ell/4$ and $m_\ell/4$, and one activated branch of width $m_\ell/2$. No additional projection is needed to restore the layer width. Because the contraction layer is also a FAN layer, the expert output is a periodic-basis representation.

The FA-FFN design can thus preserve a conventional nonlinear pathway while the sine and cosine branches augment it with learned periodic features.

Output Head Architecture. As illustrated in Figure 1d, we apply an RMSNorm after the final Transformer block to improve stability and then forward the resulting patch embedding sequence to the decoder module, which is designed as follows. A single-layer MLP with dimensions $d_{\text{model}} \times d_{\text{model}}$ receives the normalized embeddings, and a ConvTranspose layer with kernel size and stride P expands each patch token to P temporal positions, producing a length-L feature sequence. The sequence is forwarded to a GELU activation and finally projected by a convolutional block inspired by large kernel inverted bottlenecks Liu et al. (2022c) with three stages: (*i*) a depthwise temporal convolution with default kernel size of 7 to focus on neighboring decoded positions within each channel, followed by a single-group GroupNorm, which normalizes jointly over the feature and temporal dimensions; (*ii*) a point-wise convolution reducing the dimension by a factor of four, followed by a GELU activation; and (*iii*) a point-wise convolution projecting the embedding to one scalar. Finally, channel-independent outputs are reshaped to the original data dimension, $\mathbb{R}^{D \times H}$.

Figure 4 compares training and validation loss trajectories of MOHETS on the Traffic dataset (MOHETS$_{\text{small}}$, $P = 12$, $H_o = 24$, and 20 epochs of training), using a conventional MLP projection head (left) and our proposed convolutional decoder head (right). The convolutional head version achieves smoother training optimization compared to the oscillatory validation loss decrease observed in the MLP head version, leading MOHETS to an average forecasting MSE of 0.406, compared to 0.418 for the MLP head (see Table 3). Furthermore, the convolutional variant is more parameter-efficient, reducing the total number of parameters by 42% (from 8.1M parameters with the MLP head to 4.7M with the convolutional decoder), indicating that the convolutional decoder improves generalization and reduces model size.

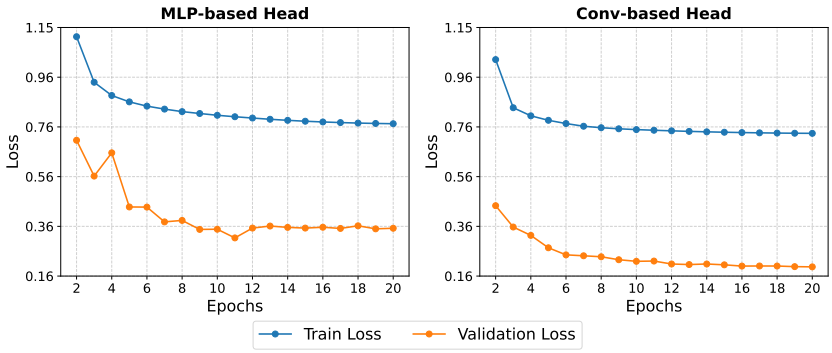

Figure 4: Training and validation trajectories of MOHETS on Traffic data, comparing a conventional MLP-based projection head (left) with our convolutional head (right). Both variants use the same architecture, optimizer, training budget, and checkpoint criterion; only the output head differs.

A.2 Training Objectives

Prediction Loss. Time series forecasting models are often trained using the MSE loss. We deviate and use the Huber loss Huber (1992); Wen et al. (2019), which combines the advantages of the L1 and MSE losses to provide robustness to outliers, improving training stability Shi et al. (2024). For predicted time points $\hat{\mathbf{x}}_t$ and ground truth $\mathbf{x}_t$, the Huber loss is defined as:

$$\mathcal{L}_{\text{pred}}(\mathbf{x}_t, \hat{\mathbf{x}}_t) = \begin{cases} 0.5\,(\mathbf{x}_t - \hat{\mathbf{x}}_t)^2\,, & \text{if } |\mathbf{x}_t - \hat{\mathbf{x}}_t| \le \delta, \\ \delta \times (|\mathbf{x}_t - \hat{\mathbf{x}}_t| - 0.5 \times \delta)\,, & \text{otherwise,} \end{cases} \tag{14}$$

with δ as a hyperparameter that balances the scaled L1 and MSE losses. As demonstrated by Shi et al. (2024), time series models trained with Huber loss outperform those trained using only MSE loss due to the superior robustness of Huber loss in handling outliers. This point-wise loss is averaged over all $D \times L$ entries of the shifted target. Thus, it supervises both the $L - H_o$ historical positions and the H_o future positions.

Expert Balance Loss. Sparse Mixture-of-Experts architectures, such as our MoHE, rely on automatically learned routing strategies that may suffer from load imbalance where a few experts dominate patch assignments. As a consequence, the routing can collapse, leading to under-utilization of experts and reduced specialization Shi et al. (2024); Dai et al. (2024). To mitigate the risk of routing collapse, we incorporate the auxiliary expert balance loss proposed by Fedus et al. (2022). This loss penalizes experts whose hard load and soft routing probability are simultaneously high, preventing stronger experts from monopolizing patches during training.

Let M be the total number of routed tokens included in the loss computation. In the implementation, these instances are aggregated across minibatch elements, variates, patch positions, and MoHE layers. Let $e_{m,k}$ be the expert selected in routing slot k for token instance m, and let $s_{i,m}$ be the routing probability the router assigns to expert i (from Softmax score, see Equation 5). Define

$$f_i = \frac{1}{KM} \sum_{m=1}^{M} \sum_{k=1}^{K} \mathbb{I}\left(e_{m,k} = i\right), \quad r_i = \frac{1}{M} \sum_{m=1}^{M} s_{i,m}. \tag{15}$$

Then

$$\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i r_i, \tag{16}$$

where f_i is the fraction of all selected routing slots assigned to expert i (hard load), while r_i is the average routing probability assigned to that expert (soft importance). Their product becomes large when both the hard assignments and the soft probability mass are concentrated on the same experts. Minimizing the summed product, therefore, encourages more balanced expert utilization by assigning higher loss values to experts whose hard load and routing probability are simultaneously large, thereby discouraging pathological concentration while still allowing nonuniform specialization.

B Experimental Details

B.1 Hyperparameter Settings

With the settings in Section 3.9, we defined four model sizes: MoHETS$_{\text{tiny}}$, with 0.3 million activated parameters; MoHETS$_{\text{small}}$, with 1.3 million activated parameters; MoHETS$_{\text{base}}$, with 7.4 million activated parameters; and MoHETS$_{\text{large}}$, with 21.4 million activated parameters; all versions are designed for inference on CPU architectures and are significantly lighter than current state-of-the-art models in long-term time series forecasting Shi et al. (2024); Liu et al. (2024a); Das et al. (2024). The detailed model configurations are in Table 4. Total Parameters includes all trainable weights, including every routed expert, and is the appropriate measure for describing model capacity and parameter storage. Activated Parameters approximates the parameters traversed by a single token, including the always-active components and K selected routed experts in each MoHE layer.

Table 4: Summary of MoHETS model configurations. Total counts may vary slightly according to data dimensionality.

	Blocks	Q-Heads	KV-Heads	Experts	K	d_{model}	d_{ff}	Activated Params	Total Params
MoHETS$_{\text{tiny}}$	4	4	2	8	2	64	128	0.3 M	0.6 M
MoHETS$_{\text{small}}$	4	4	2	8	2	128	256	1.3 M	2.5 M
MoHETS$_{\text{base}}$	6	8	4	8	2	256	512	7.4 M	14.5 M
MoHETS$_{\text{large}}$	8	12	6	8	2	384	768	21.4 M	42.7 M

For all experiments, we define a standard base frequency of $10,000$ for Rotary Position Embeddings (RoPE). We apply DropPath Huang et al. (2016) with stochastic decay to the output of the attention and MoHE modules according to the depth of their Transformer blocks, with a maximum probability of 0.3. We also apply Dropout Srivastava et al. (2014) with probability 0.2 to other encoder components. Furthermore, we use xavier_uniform Glorot & Bengio (2010) to initialize all the learnable weights of our model, except the Fourier Feed Forward networks in MoHE modules, which we initialize using normal distributions. The reason for this choice lies in the trigonometric nature of the Fourier layers Dong et al. (2024), which use explicit cos and sin components to model a periodic basis.

Initializations such as $\mathrm{xavier_}$ and $\mathrm{kaiming_}$ may not be ideal for periodic projections because they are designed to preserve variance for ReLU-style flows, but not for $\cos$ and $\sin$ operations. In practice, these initializations tend to produce distributions that are either too small or too structured, placing the pre-activations at a standard deviation of roughly one radian, confining $\cos$ and $\sin$ coordinates to a near-linear region around zero. Therefore, initialization must place the weights of the Fourier layers in a moderate range, with $\mathcal{N}(0, 1)$ providing non-trivial variations in the latent periodic features at the start of training, preventing them from collapsing.

For optimization, we apply the fused implementation of AdamW Loshchilov & Hutter (2017) with a maximum learning rate of 3.2×10^{-3}, weight_decay $= 1 \times 10^{-4}$, $\beta_1 = 0.9$, and $\beta_2 = 0.95$ Chen et al. (2020). Furthermore, we use a Cosine Annealing scheduler with a linear warmup Loshchilov & Hutter (2016) for the first 10% training steps, and a decay to a minimum learning rate of 1.2×10^{-4} and early stopping with patience $= 5$. Such a warmup is essential to stabilize the router's initial random assignment before the experts begin specializing. Following Shi et al. (2024), we set the Huber loss δ to 2.0 and the auxiliary loss factor α to 0.02. Training epochs are searched over the range 10 to 30. We do not use data augmentations. Model settings and optimization hyperparameters are selected separately for each dataset using the validation split. Once selected, the configuration is held fixed across all forecast horizons. The test split is used only for final reporting.

The code will be publicly available on GitHub upon acceptance.

B.2 Dataset Descriptions

We conduct long-term multivariate forecasting experiments on 7 well-established real-world datasets to evaluate the performance of MoHETS, including: the ETT Zhou et al. (2021) series that contains four subsets with seven features related to power load of electricity transformers recorded during two years, where ETTh1 and ETTh2 are recorded hourly, and ETTm1 and ETTm2 are recorded every 15 minutes; Weather Wu et al. (2021b) that includes 21 meteorological features collected every 10 minutes from the Max Planck Institute for Biogeochemistry; ECL Wu et al. (2021b) that contains hourly electricity consumption records from 321 clients; and Traffic Wu et al. (2021b) records hourly road occupancy rates from 862 sensors on San Francisco Bay freeways. These datasets vary substantially in dimension and temporal granularity, are publicly available, and have been extensively used to benchmark time-series forecasting models. The statistics of each dataset are provided in Table 5.

Table 5: Dataset descriptions. D denotes the number of variables. Dataset size refers to the number of time points and is organized into (Train, Validation, Test) splits. C denotes the number of covariates derived from timestamps.

Task	Dataset	D	Dataset Size	Frequency	C	Forecastability	Information
Long-term Forecasting	ETTh1	7	(8545, 2881, 2881)	1 Hour	4	0.38	Power Load
	ETTh2	7	(8545, 2881, 2881)	1 Hour	4	0.45	Power Load
	ETTm1	7	(34465, 11521, 11521)	15 Min	5	0.46	Power Load
	ETTm2	7	(34465, 11521, 11521)	15 Min	5	0.55	Power Load
	Weather	21	(36792, 5271, 10540)	10 Min	5	0.75	Weather
	ECL	321	(18317, 2633, 5261)	1 Hour	4	0.77	Electricity
	Traffic	862	(12185, 1757, 3509)	1 Hour	4	0.68	Transportation

Forecastability Goerg (2013) is a measure of future uncertainty computed by one minus the entropy of the Fourier decomposition of time series. Higher values indicate better levels of predictability. Configuration selection is based exclusively on validation-set performance, and the test set is used only for final reporting. In addition, we ensure that the "$\mathrm{drop\ last}$" option is set to False to avoid any unfair impact during testing Qiu et al. (2024). GIFT-Eval Aksu et al. (2024) is a recent benchmark designed to assess broad cross-domain generalization, making it the right venue for a pretrained forecaster. MoHETS is trained in-domain and that pretraining is left for future work; evaluating a non-pretrained model on that data would test a capability this paper does not claim to have.

We extract the calendar components from each data observation's timestamp to use as covariates, where each component is encoded as a continuous value Alexandrov et al. (2020). This is a deliberate design decision, since calendar variables are universally and deterministically available for virtually any timestamped series at zero cost, without requiring additional sensors, forecasts, subscriptions, or domain-specific data sources. Furthermore, they are exactly aligned with the endogenous samples, avoiding missing values. Hourly-resolution data has four components: hour-of-day, day-of-week, day-of-month, and day-of-year, while minute-resolution data has minute-of-hour as a fifth component. The same construction is used for all datasets.

B.3 Baseline Models

We select 15 advanced, well-known models as baselines, representing the state of the art in time series forecasting. We prioritize methods from peer-reviewed publications with publicly reported results, an official or maintained implementation, and an evaluation protocol sufficiently close to the standard chronological setting Wu et al. (2023). These baselines include Transformer-based models such as TimeXer Wang et al. (2024b), iTransformer Liu et al. (2023), PatchTST Nie et al. (2023), Timer-XL Liu et al. (2025), Time-MoE Shi et al. (2024), Moirai Woo et al. (2024), MOMENT Goswami et al. (2024), and Chronos Ansari et al. (2024); as well as SOFTS Han et al. (2024), TimeMixer Wang et al. (2024a), TiDE Das et al. (2023), and DLinear Zeng et al. (2023), which are based on MLP layers; xLSTM-Mixer Kraus et al. (2026), which is based on Long Short-Term Memory (LSTM) modules; S-Mamba Wang et al. (2025), which is a state space model; and TimesNet Wu et al. (2023), which is based on 2D convolutions. In particular, Timer-XL, TiDE, and TimeXer are recently published forecasters specifically designed to encode both historical time series and exogenous information.

SOFTS, TimeXer, iTransformer, TimeMixer, TimesNet, PatchTST, Crossformer, FEDformer, TiDE, and DLinear are in-domain (full-shot) time series models. On the other hand, Timer-XL, Time-MoE, Moirai, MOMENT, and Chronos are large time-series foundation models pretrained on multiple time-series datasets, and are reported as contextual comparisons. In particular, Time-MoE is a mixture-of-experts Transformer with three model versions that scale to more than two billion parameters and is pretrained on a vast database comprising over 300 billion time points Shi et al. (2024). We report the published baseline values from the cited sources.

B.4 Metrics

We use the mean squared error (MSE) and mean absolute error (MAE) as evaluation metrics for the forecasting experiments:

$$\text{MSE} = \frac{1}{H}\sum_{t=1}^{H}(\mathbf{x}_t - \widehat{\mathbf{x}}_t)^2, \qquad \text{MAE} = \frac{1}{H}\sum_{t=1}^{H}|\mathbf{x}_t - \widehat{\mathbf{x}}_t|,$$

where $\mathbf{x}_t, \widehat{\mathbf{x}}_t \in \mathbb{R}$ are the ground truth and the corresponding prediction of the t-th future time point. We further calculate the mean metric in the variable dimension for multivariate time series. They are standard metrics in time series forecasting literature, providing complementary point-forecast information: MSE penalizes large residuals quadratically and is therefore sensitive to occasional large errors, whereas MAE applies a linear penalty and is less dominated by extreme residuals. We note that CRPS is a metric that scores probabilistic forecasts, which requires a different model interface. For a deterministic point forecast, it reduces exactly to MAE.

C Full Experimental Results

In Table 6, we provide the full results for each forecasting task in Table 1. MoHETS uses a single dataset-specific configuration, held fixed across all four forecast lengths. We highlight the performance of MoHETS in ultra-long-horizon multivariate forecasting tasks, that is, predicting 720 future time points, where our model attains the lowest error in 9 of the 14 dataset–metric pairs, with xLSTM-M taking the remaining five. In MSE, MoHETS is best on ETTh1, ETTm1, Traffic, and Weather, reducing the error of the runner-up, xLSTM-M, by 2.4%, 2.2%, 1.8%, and 0.6%, respectively; compared with TimeXer, the reductions reach 12.8% on ETTh1 and 11.5% on ETTm1. xLSTM-M obtains the lowest MSE on ETTh2, ETTm2, and ECL. These results underscore MoHETS's robustness for long-term forecasting.

Table 6: Results of long-term multivariate forecasting experiments. Full-shot results are reported from Kraus et al. (2026); Shi et al. (2024); Wang et al. (2024b); Han et al. (2024). **Bold red**: the best, underlined blue: the second best. 1st Count represents the number of wins achieved by a model across all prediction lengths and datasets.

Models		MoHETS		xLSTM-M		S-Mamba		SOFTS		TimeXer		iTransformer		TimeMixer		TimesNet		PatchTST		TiDE		DLinear	
Metrics ↓		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.342**	**0.381**	0.359	0.386	0.386	0.405	0.381	0.399	0.382	0.403	0.386	0.405	0.375	0.400	0.384	0.402	0.414	0.419	0.479	0.464	0.386	0.400
	192	**0.369**	**0.404**	0.402	0.417	0.443	0.437	0.435	0.431	0.429	0.435	0.441	0.436	0.429	0.421	0.436	0.429	0.460	0.445	0.525	0.492	0.437	0.432
	336	**0.387**	**0.419**	0.408	0.429	0.489	0.468	0.480	0.452	0.468	0.448	0.487	0.458	0.484	0.458	0.491	0.469	0.501	0.466	0.565	0.515	0.481	0.459
	720	**0.409**	**0.441**	0.419	0.448	0.502	0.489	0.499	0.488	0.469	0.461	0.503	0.491	0.498	0.482	0.521	0.500	0.500	0.488	0.594	0.558	0.519	0.516
	Avg.	**0.377**	**0.411**	0.397	0.420	0.455	0.450	0.449	0.442	0.437	0.437	0.454	0.447	0.447	0.440	0.458	0.450	0.469	0.454	0.540	0.507	0.455	0.451
ETTh2	96	0.271	**0.328**	**0.267**	0.329	0.296	0.348	0.297	0.347	0.286	0.338	0.297	0.349	0.289	0.341	0.340	0.374	0.302	0.348	0.400	0.440	0.333	0.387
	192	0.340	**0.371**	**0.338**	0.375	0.376	0.396	0.373	0.394	0.363	0.389	0.380	0.400	0.372	0.392	0.402	0.414	0.388	0.400	0.528	0.509	0.477	0.476
	336	0.371	**0.397**	**0.367**	0.401	0.424	0.431	0.410	0.426	0.414	0.423	0.428	0.432	0.386	0.414	0.452	0.452	0.426	0.433	0.643	0.571	0.594	0.541
	720	0.391	**0.421**	**0.388**	0.424	0.426	0.444	0.411	0.433	0.408	0.432	0.427	0.445	0.412	0.434	0.462	0.468	0.431	0.446	0.874	0.679	0.831	0.657
	Avg.	0.343	**0.379**	**0.340**	0.382	0.381	0.405	0.373	0.400	0.367	0.396	0.383	0.406	0.364	0.395	0.414	0.427	0.387	0.407	0.611	0.549	0.558	0.515
ETTm1	96	**0.269**	**0.325**	0.275	0.328	0.333	0.368	0.325	0.361	0.318	0.356	0.334	0.368	0.320	0.357	0.338	0.375	0.329	0.367	0.364	0.387	0.345	0.372
	192	**0.307**	**0.352**	0.319	0.354	0.376	0.390	0.375	0.389	0.362	0.383	0.377	0.391	0.361	0.381	0.374	0.387	0.367	0.385	0.398	0.404	0.380	0.389
	336	**0.340**	**0.374**	0.353	**0.374**	0.408	0.413	0.405	0.412	0.395	0.407	0.426	0.420	0.390	0.404	0.410	0.411	0.399	0.410	0.428	0.425	0.413	0.413
	720	**0.400**	0.408	0.409	**0.407**	0.475	0.448	0.466	0.447	0.452	0.441	0.491	0.459	0.454	0.441	0.478	0.450	0.454	0.439	0.487	0.461	0.474	0.453
	Avg.	**0.329**	**0.365**	0.339	0.366	0.398	0.405	0.393	0.403	0.382	0.397	0.407	0.410	0.381	0.395	0.400	0.405	0.387	0.400	0.419	0.419	0.403	0.406
ETTm2	96	0.165	0.248	**0.157**	**0.244**	0.179	0.263	0.180	0.261	0.171	0.256	0.180	0.264	0.175	0.258	0.187	0.267	0.175	0.259	0.207	0.305	0.193	0.292
	192	0.222	0.287	**0.213**	**0.285**	0.250	0.309	0.246	0.306	0.237	0.299	0.250	0.309	0.237	0.299	0.249	0.309	0.241	0.302	0.290	0.364	0.284	0.362
	336	0.275	**0.321**	**0.269**	0.322	0.312	0.349	0.319	0.352	0.296	0.338	0.311	0.348	0.298	0.340	0.321	0.351	0.305	0.343	0.377	0.422	0.369	0.427
	720	0.358	**0.374**	**0.351**	0.377	0.411	0.406	0.405	0.401	0.392	0.394	0.412	0.407	0.391	0.396	0.408	0.403	0.402	0.400	0.558	0.524	0.554	0.522
	Avg.	0.255	0.308	**0.248**	**0.307**	0.288	0.332	0.287	0.330	0.274	0.322	0.288	0.332	0.275	0.323	0.291	0.332	0.281	0.326	0.358	0.403	0.350	0.400
Weather	96	**0.140**	**0.182**	0.143	0.184	0.165	0.210	0.166	0.208	0.157	0.205	0.174	0.214	0.163	0.209	0.172	0.220	0.177	0.218	0.202	0.261	0.196	0.255
	192	**0.183**	**0.225**	0.186	0.226	0.214	0.252	0.217	0.253	0.204	0.247	0.221	0.254	0.208	0.250	0.219	0.261	0.225	0.259	0.242	0.298	0.237	0.296
	336	**0.234**	**0.266**	0.236	**0.266**	0.274	0.297	0.282	0.300	0.261	0.290	0.278	0.296	0.251	0.287	0.280	0.306	0.278	0.297	0.287	0.335	0.283	0.335
	720	**0.308**	**0.319**	0.310	0.323	0.350	0.345	0.356	0.351	0.340	0.341	0.358	0.349	0.339	0.341	0.365	0.359	0.354	0.348	0.351	0.386	0.345	0.381
	Avg.	**0.216**	**0.248**	0.219	0.250	0.251	0.276	0.255	0.278	0.241	0.271	0.258	0.278	0.240	0.271	0.259	0.286	0.259	0.281	0.270	0.320	0.265	0.316
ECL	96	**0.125**	**0.217**	0.126	0.218	0.139	0.235	0.143	0.233	0.140	0.242	0.148	0.240	0.153	0.247	0.168	0.272	0.181	0.270	0.237	0.329	0.197	0.282
	192	**0.143**	0.236	0.144	**0.235**	0.159	0.255	0.158	0.248	0.157	0.256	0.162	0.253	0.166	0.256	0.184	0.289	0.188	0.274	0.236	0.330	0.196	0.285
	336	0.160	0.255	**0.157**	**0.250**	0.176	0.272	0.178	0.269	0.176	0.275	0.178	0.269	0.185	0.277	0.198	0.300	0.204	0.293	0.249	0.344	0.209	0.301
	720	0.200	0.292	**0.183**	**0.276**	0.204	0.298	0.218	0.305	0.211	0.306	0.225	0.317	0.225	0.310	0.220	0.320	0.246	0.324	0.284	0.373	0.245	0.333
	Avg.	0.157	0.250	**0.153**	**0.245**	0.170	0.265	0.174	0.264	0.171	0.270	0.178	0.270	0.182	0.272	0.192	0.295	0.205	0.290	0.251	0.344	0.212	0.300
Traffic	96	**0.350**	**0.235**	0.357	0.236	0.382	0.261	0.376	0.251	0.428	0.271	0.395	0.268	0.462	0.285	0.593	0.321	0.462	0.295	0.805	0.493	0.650	0.396
	192	**0.370**	0.244	0.377	**0.241**	0.396	0.267	0.398	0.261	0.448	0.282	0.417	0.276	0.473	0.296	0.617	0.336	0.466	0.296	0.756	0.474	0.598	0.370
	336	**0.388**	0.253	0.394	**0.250**	0.417	0.276	0.415	0.269	0.473	0.289	0.433	0.283	0.498	0.296	0.629	0.336	0.482	0.304	0.762	0.477	0.605	0.373
	720	**0.431**	**0.277**	0.439	0.283	0.460	0.300	0.447	0.287	0.516	0.307	0.467	0.302	0.506	0.313	0.640	0.350	0.514	0.322	0.719	0.449	0.645	0.394
	Avg.	**0.385**	**0.252**	0.392	0.253	0.414	0.276	0.409	0.267	0.466	0.287	0.428	0.282	0.484	0.297	0.620	0.336	0.481	0.304	0.760	0.473	0.625	0.383
Average		**0.295**	**0.316**	0.298	0.318	0.337	0.344	0.334	0.341	0.334	0.340	0.342	0.346	0.339	0.342	0.376	0.362	0.353	0.352	0.458	0.431	0.410	0.396
1st Count		49		25		0		0		0		0		0		0		0		0		0	

In Table 7, we present contextual comparisons with time-series foundation models to provide a more comprehensive evaluation. These models differ in pretraining data, model scale, and evaluation protocol. Specifically, we include Timer-XL, and all versions of Time-MoE, Moirai, MOMENT, and Chronos (see Section B.3). It should be noted that Moirai and Time-MoE achieve strong results on the ETTh1 and ETTh2 datasets. These datasets are small and coarse-grained, making them prone to overfitting even with models of moderate size Nie et al. (2023), suggesting that pretraining on large-scale time series is a valuable approach to improve performance in such contexts.

In particular, Time-MoE is a foundation model pretrained on a vast time-series dataset and outperforms MoHETS across a few horizons. However, MoHETS outperforms Time-MoE in most benchmarks, including a 15.6% reduction in average MSE on the Weather data relative to Time-MoE$_{\text{ultra}}$. We highlight that MoHETS is considerably lighter, demonstrating that combining carefully designed architectures to capture heterogeneous patterns from time series enables MoHETS to remain competitive with substantially larger pretrained models. The general results in Table 7 indicate that our model performs consistently against such established approaches, including the 2.4-billion-parameter Time-MoE$_{\text{ultra}}$. Even the largest MoHETS configuration contains on the order of tens of millions of total parameters.

During our experiments, we did not use pretrained versions of MoHETS, which is left for future work. For each dataset, we select one model and an optimization configuration based on the validation performance. Each configuration is held fixed across $H \in \{96, 192, 336, 720\}$, and different horizons are obtained by varying only the number of autoregressive rollout iterations. For readability and to save space for comparisons with a large set of baselines, we collect dataset-specific results of experiments using different configurations of MoHETS in Tables 1, 6, and 7, presenting them as a single column. Table 8 reports the exact model size, patch length, output chunk size, learning rate, batch size, and epoch budget used for every dataset.

Table 9 reports the forecasting performance of all four variants of MoHETS, with activated parameters ranging from approximately 300 K to 21 M. We did not evaluate the base and large variants

Table 7: Contextual comparison with time-series foundation model baselines. Results are obtained from Liu et al. (2024b). **Bold red**: the best, underlined blue: the second best. 1st Count represents the number of wins achieved by a model across all prediction lengths and datasets.

Models		MoHETS		Timer-XL$_{\text{Base}}$		Time-MoE$_{\text{Base}}$		Time-MoE$_{\text{Large}}$		Time-MoE$_{\text{Ultra}}$		Moirai$_{\text{Small}}$		Moirai$_{\text{Base}}$		Moirai$_{\text{Large}}$		MOMENT		Chronos$_{\text{Base}}$		Chronos$_{\text{Large}}$	
Metrics ↓		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.342**	0.381	0.369	0.391	0.357	0.381	0.350	0.382	0.349	**0.379**	0.401	0.402	0.376	0.392	0.381	0.388	0.688	0.557	0.440	0.393	0.441	0.390
	192	**0.369**	**0.404**	0.405	0.413	0.384	**0.404**	0.388	0.412	0.395	0.413	0.435	0.421	0.412	0.413	0.434	0.415	0.688	0.560	0.492	0.426	0.502	0.524
	336	**0.387**	**0.419**	0.418	0.423	0.411	0.434	0.411	0.430	0.447	0.453	0.438	0.434	0.433	0.428	0.485	0.445	0.675	0.563	0.550	0.462	0.576	0.467
	720	**0.409**	**0.441**	0.423	**0.441**	0.449	0.477	0.427	0.455	0.457	0.462	0.439	0.454	0.447	0.444	0.611	0.510	0.683	0.585	0.882	0.591	0.835	0.583
	Avg.	**0.377**	**0.411**	0.404	0.417	0.400	0.424	0.394	0.419	0.412	0.426	0.428	0.427	0.417	0.419	0.480	0.439	0.683	0.566	0.591	0.468	0.588	0.466
ETTh2	96	**0.271**	**0.328**	0.283	0.342	0.305	0.359	0.302	0.354	0.292	0.352	0.297	0.336	0.294	0.330	0.296	0.330	0.342	0.396	0.308	0.343	0.320	0.345
	192	**0.340**	**0.371**	**0.340**	0.379	0.351	0.386	0.364	0.385	0.347	0.379	0.368	0.381	0.365	0.375	0.361	**0.371**	0.354	0.402	0.384	0.392	0.406	0.399
	336	0.371	0.397	0.366	0.400	0.391	0.418	0.417	0.425	0.406	0.419	0.370	0.393	0.376	**0.390**	0.390	**0.390**	**0.356**	0.407	0.429	0.430	0.492	0.453
	720	**0.391**	0.421	0.397	0.431	0.419	0.454	0.537	0.496	0.439	0.447	0.411	0.426	0.416	0.433	0.423	**0.418**	0.395	0.434	0.501	0.477	0.603	0.511
	Avg.	**0.343**	0.379	0.347	0.388	0.366	0.404	0.405	0.415	0.371	0.399	0.361	0.384	0.362	0.382	0.367	**0.377**	0.361	0.409	0.405	0.410	0.455	0.427
ETTm1	96	**0.269**	**0.325**	0.317	0.356	0.338	0.368	0.309	0.357	0.281	0.341	0.418	0.392	0.363	0.356	0.380	0.361	0.654	0.527	0.454	0.408	0.457	0.403
	192	0.307	**0.352**	0.358	0.381	0.353	0.388	0.346	0.381	**0.305**	0.358	0.431	0.405	0.388	0.375	0.412	0.383	0.662	0.532	0.567	0.477	0.530	0.450
	336	**0.340**	**0.374**	0.386	0.401	0.381	0.413	0.373	0.408	0.369	0.395	0.433	0.412	0.416	0.392	0.436	0.400	0.672	0.537	0.662	0.525	0.577	0.481
	720	**0.400**	**0.408**	0.430	0.431	0.504	0.493	0.475	0.477	0.469	0.472	0.462	0.432	0.460	0.418	0.462	0.420	0.692	0.551	0.900	0.591	0.660	0.526
	Avg.	**0.329**	**0.365**	0.373	0.392	0.394	0.415	0.376	0.405	0.356	0.391	0.436	0.410	0.406	0.385	0.422	0.391	0.670	0.536	0.645	0.500	0.555	0.465
ETTm2	96	**0.165**	**0.248**	0.189	0.277	0.201	0.291	0.197	0.286	0.198	0.288	0.214	0.288	0.205	0.273	0.211	0.274	0.260	0.335	0.199	0.274	0.197	0.271
	192	**0.222**	**0.287**	0.241	0.315	0.258	0.334	0.250	0.322	0.235	0.312	0.284	0.332	0.275	0.316	0.281	0.318	0.289	0.350	0.261	0.322	0.254	0.314
	336	**0.275**	**0.321**	0.286	0.348	0.324	0.373	0.337	0.375	0.293	0.348	0.331	0.362	0.329	0.350	0.341	0.355	0.324	0.369	0.326	0.366	0.313	0.353
	720	**0.358**	**0.374**	0.375	0.402	0.488	0.464	0.480	0.461	0.427	0.428	0.402	0.408	0.437	0.411	0.485	0.428	0.394	0.409	0.455	0.439	0.416	0.415
	Avg.	**0.255**	**0.308**	0.273	0.336	0.317	0.365	0.316	0.361	0.288	0.344	0.307	0.347	0.311	0.337	0.329	0.343	0.316	0.365	0.310	0.350	0.295	0.338
Weather	96	**0.140**	**0.182**	0.171	0.225	0.160	0.214	0.159	0.213	0.157	0.211	0.198	0.222	0.220	0.217	0.199	0.211	0.243	0.255	0.203	0.238	0.194	0.235
	192	**0.183**	**0.225**	0.221	0.271	0.210	0.260	0.215	0.266	0.208	0.256	0.247	0.265	0.271	0.259	0.246	0.251	0.278	0.329	0.256	0.290	0.249	0.285
	336	**0.234**	**0.266**	0.274	0.311	0.274	0.309	0.291	0.322	0.255	0.290	0.283	0.303	0.286	0.297	0.274	0.291	0.306	0.346	0.314	0.336	0.302	0.327
	720	**0.308**	**0.319**	0.356	0.370	0.418	0.405	0.415	0.400	0.405	0.397	0.373	0.354	0.373	0.354	0.337	0.340	0.350	0.374	0.397	0.396	0.372	0.378
	Avg.	**0.216**	**0.248**	0.256	0.294	0.265	0.297	0.270	0.300	0.256	0.288	0.275	0.286	0.287	0.281	0.264	0.273	0.294	0.326	0.292	0.315	0.279	0.306
ECL	96	**0.125**	**0.217**	0.141	0.237	–	–	–	–	–	–	0.189	0.280	0.160	0.250	0.153	0.241	0.745	0.680	0.154	0.231	0.152	0.229
	192	**0.143**	**0.236**	0.159	0.254	–	–	–	–	–	–	0.205	0.292	0.175	0.263	0.169	0.255	0.755	0.683	0.179	0.254	0.172	0.250
	336	**0.160**	**0.255**	0.177	0.272	–	–	–	–	–	–	0.221	0.307	0.187	0.277	0.187	0.273	0.766	0.687	0.214	0.284	0.203	0.276
	720	**0.200**	**0.292**	0.219	0.308	–	–	–	–	–	–	0.258	0.335	0.228	0.309	0.237	0.313	0.794	0.696	0.311	0.346	0.289	0.337
	Avg.	**0.157**	**0.250**	0.174	0.278	–	–	–	–	–	–	0.218	0.303	0.187	0.274	0.186	0.270	0.765	0.686	0.214	0.278	0.204	0.273
Average		**0.280**	**0.327**	0.305	0.351	0.348	0.381	0.352	0.380	0.337	0.370	0.338	0.360	0.328	0.346	0.341	0.349	0.515	0.481	0.410	0.387	0.396	0.379
1st Count		56		2		1		0		2		0		1		4		1		0		0	

∗ Traffic from PEMS Liu et al. (2022a) is typically used for pretraining large time-series models and is therefore not evaluated here.
∗∗ Datasets used for pretraining are not evaluated on the corresponding models; dashes (–) denote unavailable results.

Table 8: Dataset-specific MoHETS configurations used for the results in Tables 1, 6, and 7. For each dataset, the listed configuration is fixed across all evaluated forecast horizons. LR denotes the maximum learning rate.

Dataset	Model version	Training Process					
		P	H_o	LR	Min LR	Batch Size	Epochs
ETTh1	MoHETS$_{\text{tiny}}$	8	24	3.2×10^{-3}	1.2×10^{-4}	128	30
ETTh2	MoHETS$_{\text{tiny}}$	8	24	3.2×10^{-3}	1.2×10^{-4}	128	30
ETTm1	MoHETS$_{\text{base}}$	16	24	3.2×10^{-3}	1.2×10^{-4}	128	20
ETTm2	MoHETS$_{\text{base}}$	16	24	3.2×10^{-3}	1.2×10^{-4}	128	20
Weather	MoHETS$_{\text{large}}$	16	24	2.2×10^{-3}	1.2×10^{-4}	96	30
ECL	MoHETS$_{\text{large}}$	12	24	1.8×10^{-3}	1.2×10^{-4}	5	10
Traffic	MoHETS$_{\text{base}}$	12	24	2.6×10^{-3}	1.2×10^{-4}	6	10

on the hourly ETT subsets because larger-capacity models are prone to overfitting on such small datasets Nie et al. (2023). Similarly, MoHETS$_{\text{large}}$ was not evaluated on Traffic due to hardware and time constraints in training a high-capacity model on such a large-scale, multivariate dataset. Overall, we observe improvements in MSE and MAE as both the data and the model sizes increase. We further analyze this scaling behavior in the following section.

C.1 Scalability Analysis

Increasing model size and the number of training tokens generally improves performance, a phenomenon known as scaling laws Kaplan et al. (2020). We evaluate the impacts of scalability on MoHETS's forecasting performance by varying the representation dimension $d_{\text{model}} \in \{64, 128, 256, 384\}$ (see Table 4). Deep learning training is stochastic, so results can vary across runs and initializations. Although training large models typically incurs high computational costs, hindering the execution of multiple trials per experiment, we report the mean and standard deviation of MoHETS performance across five runs with different random seeds. For these experiments, we use the ETTm1, ETTm2, Weather, and ECL datasets. ETTh1 and ETTh2 are excluded due to the high risk of overfitting Nie et al. (2023), and Traffic is excluded due to hardware and time constraints.

Table 9: Results of long-term multivariate forecasting experiments with all versions of MoHETS. **Bold red**: the best, underlined blue: the second best.

Models		MoHETS$_{tiny}$		MoHETS$_{small}$		MoHETS$_{base}$		MoHETS$_{large}$	
Metrics ↓		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.342**	**0.381**	0.355	0.389	–	–	–	–
	192	**0.369**	**0.404**	0.388	0.410	–	–	–	–
	336	**0.387**	**0.419**	0.406	0.422	–	–	–	–
	720	**0.409**	**0.441**	0.410	0.439	–	–	–	–
	Avg.	**0.377**	**0.411**	0.390	0.415	–	–	–	–
ETTh2	96	**0.271**	**0.328**	0.288	0.336	–	–	–	–
	192	**0.340**	**0.371**	0.362	0.383	–	–	–	–
	336	**0.371**	**0.397**	0.394	0.411	–	–	–	–
	720	**0.391**	**0.421**	0.408	0.433	–	–	–	–
	Avg.	**0.343**	**0.379**	0.363	0.391	–	–	–	–
ETTm1	96	0.294	0.337	0.281	0.326	**0.269**	**0.325**	0.279	0.329
	192	0.342	0.368	0.319	0.354	**0.307**	**0.352**	0.322	0.357
	336	0.386	0.396	0.349	0.375	**0.340**	**0.374**	0.357	0.380
	720	0.467	0.441	0.404	0.409	**0.400**	**0.408**	0.414	0.413
	Avg.	0.372	0.386	0.338	0.366	**0.329**	**0.365**	0.343	0.370
ETTm2	96	0.169	0.249	0.165	**0.247**	**0.165**	0.248	0.172	0.252
	192	0.235	0.292	0.226	**0.287**	**0.222**	**0.287**	0.237	0.293
	336	0.296	0.332	0.280	0.323	**0.275**	**0.321**	0.294	0.331
	720	0.388	0.390	0.369	0.377	**0.358**	**0.374**	0.384	0.385
	Avg.	0.272	0.316	0.260	0.309	**0.255**	**0.308**	0.272	0.315
Weather	96	0.155	0.202	0.147	0.189	0.143	0.186	**0.140**	**0.182**
	192	0.206	0.249	0.195	0.236	0.188	0.229	**0.183**	**0.225**
	336	0.261	0.291	0.248	0.277	0.240	0.270	**0.234**	**0.266**
	720	0.335	0.345	0.335	0.334	0.313	0.323	**0.308**	**0.319**
	Avg.	0.239	0.272	0.231	0.259	0.221	0.252	**0.216**	**0.248**
ECL	96	0.140	0.234	0.130	0.222	0.127	0.218	**0.125**	**0.217**
	192	0.158	0.251	0.148	0.239	0.145	**0.236**	**0.143**	**0.236**
	336	0.179	0.271	0.167	0.258	0.162	**0.254**	**0.160**	0.255
	720	0.226	0.311	0.211	0.297	0.204	**0.292**	**0.200**	**0.292**
	Avg.	0.176	0.267	0.164	0.254	0.160	**0.250**	**0.157**	**0.250**
Traffic	96	0.376	0.251	0.355	0.239	**0.350**	**0.235**	–	–
	192	0.393	0.259	0.375	0.248	**0.370**	**0.244**	–	–
	336	0.411	0.269	0.393	0.258	**0.388**	**0.253**	–	–
	720	0.452	0.292	0.434	0.283	**0.431**	**0.277**	–	–
	Avg.	0.408	0.268	0.389	0.257	**0.385**	**0.252**	–	–

∗ Dataset not evaluated on corresponding models; dashes denote results (–).

Figure 5 indicates that increasing model capacity generally improves forecast performance of MoHETS, particularly on larger datasets, where additional parameters can be utilized effectively, confirming the benefits of scalability in the time-series domain Shi et al. (2024). ETTm1 and ETTm2 improve up to $d_{\text{model}} = 256$ and then degrade slightly at $d_{\text{model}} = 384$, while Weather and ECL continue to improve over the tested range, indicating robust scaling benefits on larger datasets. Thus, across the evaluated datasets and model widths, larger representations often improve accuracy until the available data no longer support the additional capacity. These results not only align with scaling laws but also support MoHETS's potential for further scaling in resource-rich contexts.

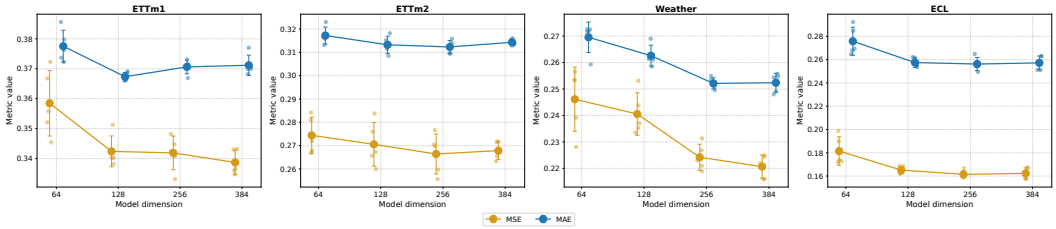

Figure 5: Scalability analysis with varying d_{model} sizes. Lower MSE or MAE indicates better performance. Small markers denote individual runs; large markers and error bars denote mean ± standard deviation.

C.2 Transformer Backbone Type

We compare encoder-only and decoder-only variants of MoHETS. Table 10 shows that the encoder-only design outperforms the decoder-only variant, achieving average MSE reductions of 8.1% on ETTh1 and 9.6% on Traffic. Additionally, the encoder-only approach is more flexible in supporting longer output resolutions, which can help mitigate the accumulation of autoregressive errors in long-horizon forecasting.

Table 10: Ablation study with different Transformer backbones. The best results are in **bold**.

Dataset	ETTh1		ETTh2		ETTm1		ETTm2		Weather		ECL		Traffic		Average ↓	
Metrics (Avg.) ↓	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Encoder	**0.383**	**0.415**	**0.362**	**0.389**	**0.338**	**0.366**	0.270	**0.313**	**0.244**	**0.265**	**0.164**	**0.254**	**0.406**	**0.276**	**0.310**	**0.325**
Decoder	0.417	0.430	0.371	0.395	0.354	0.375	**0.269**	0.315	0.255	0.270	0.167	0.262	0.449	0.297	0.326	0.335

C.3 Patch Lengths

Forecast performance can be sensitive to patch length P. Smaller patches retain finer local resolution but increase attention length, FLOPs, and activation memory, reducing computational efficiency and increasing GPU memory demands due to extended sequences. In contrast, larger patches compress the sequence more aggressively, but may discard useful local variation, degrading accuracy in coarse-grained data. Previous works Zhang & Yan (2022); Nie et al. (2023); Wang et al. (2024b); Liu et al. (2024a) demonstrate the effects of different patch sizes, suggesting that moderate lengths (e.g., 8–24) offer a good balance between efficiency and capture of temporal patterns.

We search $P \in \{8, 12, 16\}$ Nie et al. (2023), with Table 8 showing our best configurations according to each benchmark. As we can see in Tables 8 and 9, for coarse-grained hourly datasets, reduced patch lengths ($P \in \{8, 12\}$) yield superior performances, as they preserve local patterns critical for high-frequency signals. On the other hand, minute-level datasets benefit from longer patch lengths ($P = 16$), as larger patches capture broader temporal trends Zhang et al. (2024). Values of P outside this range degraded performance in either direction, confirming that the optimal patch lengths depend on the dataset frequency Zhang & Yan (2022).

C.4 Look-back Context

The look-back length L controls the amount of historical conditioning, but it does not change the evaluation targets. Under the chronological protocol Wu et al. (2023), the test interval is fixed, and each test window draws its context from preceding observations. Changing L therefore shifts only the left boundary of the historical window; it does not change the first predicted timestamp, the set of eligible target windows, or the number of eligible target windows. Under this protocol, models with different look-back lengths are evaluated on the same targets but receive different amounts of historical information.

A longer context is not guaranteed to improve forecasting accuracy Zeng et al. (2023). It may expose the model to more history and recurring structures, but can also introduce stale regimes, nuisance variation, and a harder optimization problem. The effect of L is coupled with the dataset, architecture, optimization, and temporal-resolution hyperparameters, particularly the patch length P. MoHETS processes $S = L/P$ patch tokens rather than L individual observations. With $L = 672$ and $P \in \{8, 12, 16\}$, the Transformer therefore attends to $S \in \{84, 56, 42\}$ tokens.

We selected $L = 672$ a priori and held it fixed across all datasets and forecast horizons rather than tuning it for individual results. Because the datasets have different sampling frequencies, this standardizes the number of observations rather than the elapsed duration. MoHETS is not architecturally tied to a fixed look-back length. The implementation exposes L as a hyperparameter, and the model can be instantiated and trained with any L that satisfies the patch-alignment condition $P \mid L$. It can also process shorter patch-aligned contexts up to its maximum context size. Therefore, we treat $L = 672$ as a design choice for the present evaluation, not as a universally optimal value or as a claim that forecasting accuracy increases monotonically with L.

C.5 Output Chunk Size

We evaluate the trade-off between forecasting efficiency and accuracy, ablating $\text{MoHETS}_{\text{tiny}}$ ($d_{\text{model}} = 64$ and $P = 8$) with different output chunk sizes $H_o \in \{8, 16, 24, 32\}$ on ETTh1 and ETTh2, with MSE and MAE results averaged over the full horizons $H \in \{96, 192, 336, 720\}$. Table 11 presents the MSE, MAE, and total forecasting time for both datasets. Increasing H_o reduces the number of autoregressive iterations from $\lceil H/8 \rceil$ toward $\lceil H/32 \rceil$, improving computational efficiency but also requiring each forward pass to generate a longer chunk. For ETTh1, MSE improves from the output chunk size $H_o = 8$ to $H_o = 24$ with a 8.5% reduction, before degrading at $H_o = 32$, indicating a

possible overgeneralization. Similarly, the ETTh2 columns show a 13.4% reduction in MSE from $H_o = 8$ to $H_o = 24$. The forecast time decreases 77% from the output chunk size $H_o = 8$ to $H_o = 32$; however, the forecast performance degrades when H_o increases to 32. We observed similar behaviors across the other benchmark datasets, which led us to set $H_o = 24$ as the optimal balance between inference efficiency and accuracy in the main configurations.

Table 11: Ablation study with different output chunk sizes. A lower MSE or MAE indicates a better prediction. The best results are in **bold**.

Dataset			ETTh1		ETTh2		
Model version	P	H_o	MSE ↓	MAE ↓	MSE ↓	MAE ↓	Time (s)
MoHETS$_{\text{tiny}}$	8	8	0.412	0.430	0.396	0.408	113
	8	16	0.390	0.421	0.372	0.391	57
	8	24	**0.377**	**0.411**	**0.343**	**0.379**	35
	8	32	0.402	0.419	0.366	0.392	26

C.6 Computational Efficiency and Memory Footprint

In addition to forecasting performance, we analyze the computational cost of MoHETS in terms of floating-point operations (FLOPs) and memory footprint during training and inference. These measurements are intended to characterize the practical scalability of our proposed architecture across the smallest to the largest model size. To standardize the measurements, we fixed the inputs to have a batch size of 1 and an input shape of $(1, D, L)$, where D denotes the number of input variables in the benchmark datasets (see Table 5), and $L = 672$ represents the look-back window. All models were evaluated under the same look-back window, forecasting configuration, numerical precision (TF32), and implementation backend.

We report FLOPs following the conventions established by fvcore Research (2020). Memory usage was measured separately for inference and training, as these regimes impose substantially different resource requirements. Inference memory was estimated during a forward pass in evaluation mode, while training memory was measured over a complete optimization step, including the forward pass, loss computation, backward pass, and optimizer update. In Table 12, we summarize FLOPs and peak allocated memory, providing estimates of computational and memory requirements for training or deploying MoHETS and allowing us to analyze the trade-off between predictive performance and resource consumption as model capacity increases.

Table 12: Computational efficiency and memory footprint of MoHETS. Training represents the peak memory usage during training, and Inference represents the peak memory usage during inference.

Models		MoHETS$_{\text{tiny}}$			MoHETS$_{\text{small}}$			MoHETS$_{\text{base}}$			MoHETS$_{\text{large}}$		
Dataset	P	FLOPs	Training	Inference	FLOPs	Training	Inference	FLOPs	Training	Inference	FLOPs	Training	Inference
ETT	8	0.24G	58.09MB	15.60MB	0.84G	114.08MB	26.32MB	4.46G	362.39MB	80.73MB	12.71G	864.12MB	200.36MB
	12	0.16G	49.90MB	15.44MB	0.57G	97.75MB	26.30MB	3.04G	316.90MB	81.79MB	8.64G	771.27MB	203.02MB
	16	0.12G	45.92MB	15.43MB	0.45G	90.26MB	26.56MB	2.36G	293.66MB	83.37MB	6.66G	728.41MB	206.97MB
Weather	8	0.71G	121.32MB	22.16MB	2.51G	238.58MB	39.21MB	13.37G	716.25MB	106.89MB	38.12G	1.55GB	238.49MB
	12	0.47G	96.36MB	21.45MB	1.71G	188.05MB	38.14MB	9.12G	573.62MB	105.41MB	25.91G	1.25GB	239.47MB
	16	0.36G	84.02MB	21.17MB	1.34G	163.22MB	37.88MB	7.07G	497.57MB	106.11MB	19.96G	1.11GB	240.98MB
ECL	8	11.05G	1.52GB	164.36MB	38.46G	3.01GB	321.47MB	0.20T	8.40GB	655.90MB	0.58T	15.79GB	1.06GB
	12	7.37G	1.15GB	151.96MB	26.30G	2.24GB	293.27MB	0.14T	6.24GB	612.94MB	0.40T	11.28GB	0.99GB
	16	5.73G	0.93GB	144.97MB	20.64G	1.82GB	282.59MB	0.11T	5.10GB	592.16MB	0.31T	9.10GB	0.96GB
Traffic	8	30.62G	4.01GB	427.67MB	0.10T	7.89GB	824.42MB	0.55T	22.19GB	1.65GB	1.57T	41.15GB	2.55GB
	12	20.72G	2.97GB	394.23MB	71.56G	5.83GB	759.56MB	0.38T	16.27GB	1.53GB	1.06T	29.27GB	2.36GB
	16	16.33G	2.45GB	374.54MB	56.36G	4.79GB	726.82MB	0.29T	13.25GB	1.47GB	0.82T	23.47GB	2.28GB

The computational cost of MoHETS can be controlled through the patch length and model width. So, increasing P reduces the number of attention tokens and changes the temporal granularity processed by the backbone model. In our experiments, larger patch sizes consistently reduced FLOPs and memory consumption, although this choice may also affect the temporal granularity available to the model. Training memory can also be reduced by adopting bfloat16 (BF16) mixed-precision, which lowers the storage cost of activations and gradients while preserving numerical stability on modern accelerators, with minimal degradation in predictive performance Shi et al. (2024); Kalamkar et al.

(2019). For example, MoHETS$_{\text{tiny}}$ with $P = 8$ requires 51.36MB of memory during training on ETT data using BF16, an 11.6% reduction compared to TF32, while MoHETS$_{\text{large}}$ with $P = 8$ requires 31.19GB for training on Traffic using BF16, a 24.2% reduction in memory usage.

Finally, MoHETS remains flexible under tight computational constraints because both FLOPs and memory usage can be further reduced by shortening the look-back length, enabling a controllable trade-off among historical context, forecast accuracy, and resource requirements.

D Expert-Utilization Diagnostics

The expert balance loss is designed to discourage routing collapse. However, it should not be interpreted as forcing perfectly uniform routing. Expert specialization is expected and often desirable in the context of MoEs. The key objective is to avoid pathological routing behavior, in which only a small subset of experts receives most tokens, while other experts remain inactive or nearly unused Shazeer et al. (2017); Fedus et al. (2022); Shi et al. (2024). The expert balance factor α controls the strength of experts' regularization (Equation 11). If α is too small, routing collapse may occur because the model prioritizes the main prediction loss and ignores the expert balance. If α is too large, the model may over-prioritize uniform routing, potentially limiting expert specialization and reducing predictive performance.

It is critical to monitor and ensure a healthy balance of experts when implementing MoE-based models. Therefore, we include a built-in expert utilization-tracking framework with MoHETS, providing quantitative and visual diagnostics to analyze the routing behavior of the MoHE layers during training. These diagnostics are useful for detecting routing collapse, dead experts, probability concentration, and layer-dependent assignment patterns, as well as for evaluating whether the balance loss effectively encourages diverse expert usage. The diagnostics in this section evaluate the Topk routed experts, as the shared path is evaluated for every token. Semantic analysis of routing assignments is set as future work.

D.1 Global Utilization

The global expert utilization monitors whether the MoHE routers fairly distribute tokens across experts or collapse toward a small subset. Hard utilization (Figure 6, left) reflects the fraction of routed selections, f_i, assigned to each expert i through training. For a model with N experts, the ideal uniform utilization reference is $1/N$. When using Topk routing, hard utilization reflects the fraction of selected routing slots assigned to each expert, not necessarily the fraction of unique tokens processed by each expert. For example, we use $k = 2$ in MoHETS, meaning each token receives two expert assignments. Healthy utilization is not characterized by every expert receiving exactly $1/N$ of the assignments, but rather by all experts receiving non-negligible fractions of the assignments.

The soft importance (Figure 6, right) complements the hard utilization, describing how routers distribute the probability mass r_i across experts before hard Topk selection. A small subset of experts may receive slightly higher soft probabilities if they become specialized in common temporal patterns. However, all experts should be given some probability mass, indicating that the router is not becoming overconfident. A particularly concerning pattern occurs when hard utilization and soft importance collapse simultaneously, meaning that the router not only selects a few experts, but also assigns almost all the probability mass to them.

One of the most direct indicators of routing collapse is the dead-experts metric, which counts the number of experts receiving zero token assignments. The number of dead experts should be zero (or very small) after the initial training steps. If one or more experts remain dead for multiple training epochs, they will not receive meaningful gradient updates, indicating that the expert balance loss is too weak, the routers are poorly initialized, or the data distribution requires only a subset of experts.

The router entropy measures how spread the router probability distributions are. For each token, the entropy is computed from the $\mathrm{Softmax}$ routing probabilities. For N experts, the maximum entropy is $\log N$, which corresponds to a perfectly uniform routing distribution. A low entropy early in training suggests premature router confidence. If entropy remains low, routers may be assigning most of the probability mass to a small subset of experts. In contrast, non-degenerate routing behavior exhibits high entropy at the beginning of training, followed by a gradual decrease, suggesting that routers start

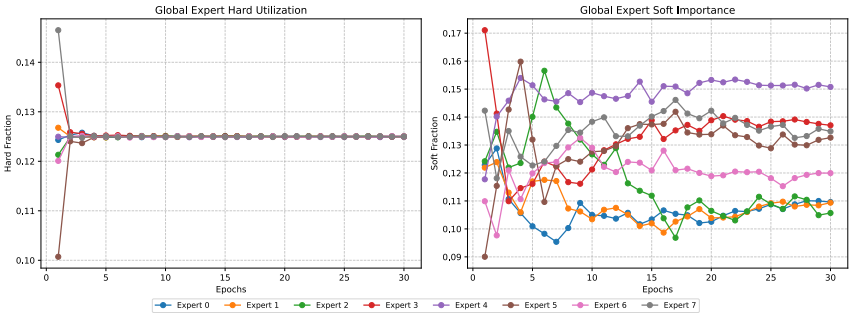

Figure 6: Global expert utilization (hard and soft) tracking over the training epochs of MoHETS$_{\text{small}}$ on the Weather dataset.

broadly exploratory and then become more selective as experts specialize. Figure 7 (left) shows the number of dead experts and the average router entropy of MoHETS$_{\text{small}}$ over the training epochs on the Weather data.

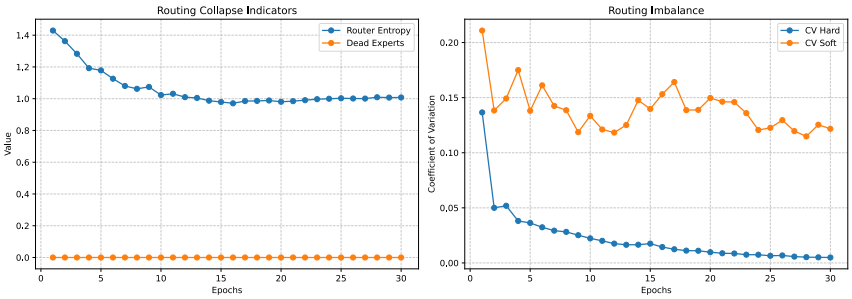

Figure 7: Router entropy, dead experts, and coefficient of variation (CV) during training of MoHETS$_{\text{small}}$ on the Weather dataset.

In Figure 7 (right), we present the coefficient of variation (CV), which measures the relative dispersion in expert usage. CV is calculated as σ/μ, where σ is the standard deviation of expert usage, and μ is the mean usage. An increasing CV suggests that usage is becoming more concentrated among a few experts, indicating a growing risk of collapse. In contrast, a decreasing CV indicates refinements in expert usage. A CV near zero means that all experts are used almost equally. However, a near-zero CV is not always the goal, because specialization may naturally introduce some imbalance.

Router entropy and CV provide complementary summaries. A gradual reduction in entropy may indicate specialization, but an abrupt collapse in entropy, combined with an increasing CV, indicates a pathological concentration of routing decisions.

D.2 Layerwise Utilization

A globally balanced routing profile can indicate non-degenerate routing, but it does not guarantee that every MoHE layer is balanced, since global aggregation can hide local routing problems. For example, global utilization may appear balanced even if one MoHE layer is collapsed while the others remain balanced. A single collapsed router reduces the model's effective capacity at that depth. Therefore, layerwise expert diagnostics are critical, and we also provide tools for fine-grained inspection of expert utilization in each MoHE layer. Figure 8 shows the routing profiles of MoHETS$_{\text{small}}$ across the training epochs on Weather.

Healthy behavior corresponds to the active usage of most experts in each layer. A layer has suspicious behavior if it shows persistent dominance by a single expert or if most experts remain inactive across multiple epochs. However, different layers may exhibit different routing patterns, as in the plots in Figure 8. Different routing profiles are expected across the model due to layer-dependent routing specialization: early layers process general low-level temporal representations, while deeper layers operate on more abstract or task-specific features.

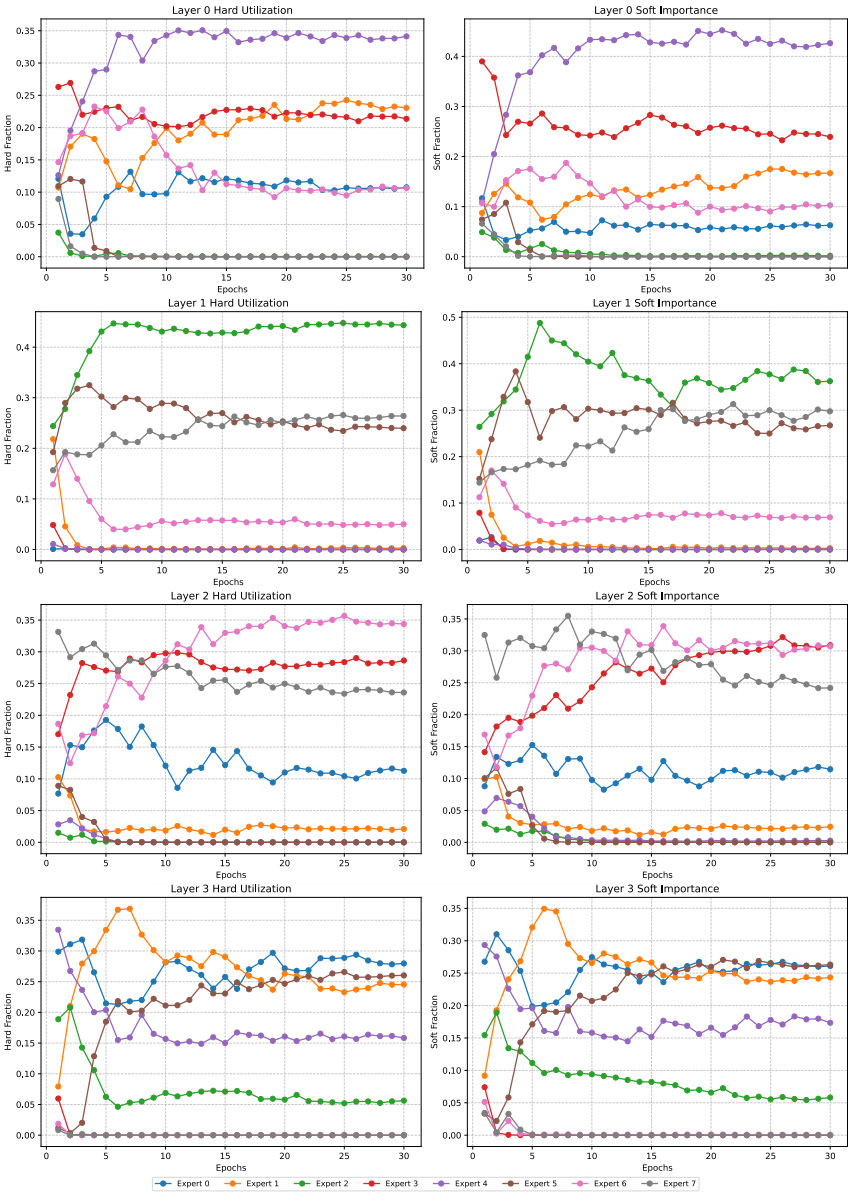

Figure 8: Layerwise expert utilization (hard and soft) during training of MoHETS$_{\text{small}}$ on the Weather dataset.

Overall, the joint interpretation of the expert routing metrics suggests that MoE-based models should be evaluated not only by their forecasting performance but also by their routing dynamics. Taken together, the hard-load, soft-importance, entropy, dead-expert, and coefficient-of-variation diagnostics indicate that the routed expert pool does not collapse during the reported training runs. Most experts receive non-negligible assignments, probability mass does not concentrate abruptly, and different layers develop different utilization profiles. Perfect uniformity is neither observed nor required. For simplicity, we demonstrate expert usage with MoHETS$_{\text{small}}$ here. Our expert tracking framework also provides heatmaps for custom versions with more experts.

E Discussion, Limitations, and Future Work

Although both language and time series are sequential data with long-range dependencies, they differ fundamentally: language relies on deterministic structures and semantic patterns, whereas time series

data often arise from stochastic processes with complex temporal dynamics. Time is not language. Consequently, effective temporal modeling requires specialized architectures beyond simply scaling language models. MOHETS introduces tailored components tailored to temporal structure, including continuity, seasonality, and non-stationarity. MoHE guarantees structural differences between its shared and routed paths, and the expert-composition ablations support the selected configuration. The utilization diagnostics further show that the routed pool remains non-degenerate.

Although MOHETS demonstrates significant capabilities, specific directions warrant further exploration. All routed expert parameters remain stored even though each token selects only K experts. Different tokens in a batch can collectively invoke every routed expert. In contexts with large model configurations (e.g., $d_{\text{model}} = 384$) and high-dimensional datasets, inference latency increases, particularly during long autoregressive rollouts (e.g., $H = 720$ time points). Increasing the output chunk size improves efficiency and, up to a point, accuracy; overly large chunks degrade accuracy on coarse-grained datasets. We find $H_o = 24$ to be a balance between inference speed and accuracy, generating $24\times$ more future points per iteration than standard "next-token" autoregression.

Caching mechanisms could be used to speed up forecasting. Widely used in language models to accelerate inference, KV Caching Pope et al. (2023) has not been thoroughly investigated in time series forecasting, particularly for patch-level attention encoders with multi-resolution outputs. Thus, adapting caching approaches for encoder-only models is a promising direction to reduce latency in long-horizon predictions. Another inspiring concept for future exploration is the multi-resolution projection head Shi et al. (2024), in which multiple projection heads, each corresponding to a distinct forecasting horizon, are optimized during training to provide more flexibility for handling time-series data with different frequencies.

The current study uses complete, timestamp-aligned calendar coordinates as covariates. Extending the conditioning interface to include general C-dimensional projections, availability masks, alignment mechanisms, and separate treatment of observed and forecast covariates is future work that can improve robustness in real-world scenarios Wang et al. (2024b); Zhang et al. (2024). Furthermore, extending MoHE to incorporate additional architectures (e.g., graph-based or attention-based experts) could capture more complex temporal dependencies, broadening MOHETS's applicability to heterogeneous time-series tasks. A structured routing formulation that dispatches contiguous patch groups would enable routing to be performed jointly by convolutional and point-wise experts.Finally, we reported models trained from scratch for each dataset. Scaling and pretraining MOHETS on large-scale, heterogeneous time-series corpora could enable zero-shot forecasting across diverse domains by leveraging the structural expert heterogeneity and sparsity of MoHE.

F IMPACT STATEMENT

This work advances long-term multivariate time series forecasting, a capability that can improve decision-making in energy systems, climate modeling, supply chain management, and public health by increasing predictive accuracy and computational efficiency. Although these improvements support positive social outcomes, such as reducing energy waste and improving disaster readiness, forecasting models also carry risks. In particular, they can reproduce or amplify biases present in historical data, which may be misapplied in automated high-stakes decision systems without adequate human oversight. To mitigate these risks, we emphasize the need for rigorous validation, transparency in data provenance, and human oversight in deployment. We are not aware of any malicious applications specific to this research beyond those common to forecasting systems, but we encourage practitioners to apply domain-appropriate safeguards before operational deployment.

G FORECAST SHOWCASES

To provide a qualitative assessment of MOHETS's performance, we visualize its forecasting results across different channels from the test sets of each benchmark dataset, namely ETTh1, ETTh2, ETTm1, ETTm2, Weather, ECL, and Traffic (Figures 9 – 15). In each figure, the forecast horizon H is set to 96 time steps. To enhance clarity and ensure intuitive visualizations, we display the full predicted horizon alongside a slice of the historical input data (look-back window) and the corresponding ground-truth future values.

These visualizations illustrate MoHETS's ability to generate accurate and coherent forecasts in highly heterogeneous multivariate time series, underscoring the effectiveness of the proposed architecture. As demonstrated in quantitative evaluations (see Section C), MoHETS's performance gains are particularly pronounced in long- and ultra-long-term prediction settings, where it robustly captures complex temporal dynamics. Overall, these qualitative results highlight the practical utility of MoHETS for long-term multivariate time series forecasting.

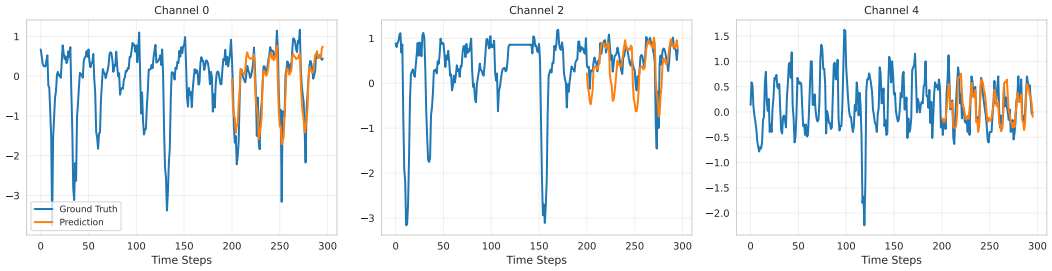

Figure 9: Forecast showcases of MoHETS across different time channels from **ETTh1**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.

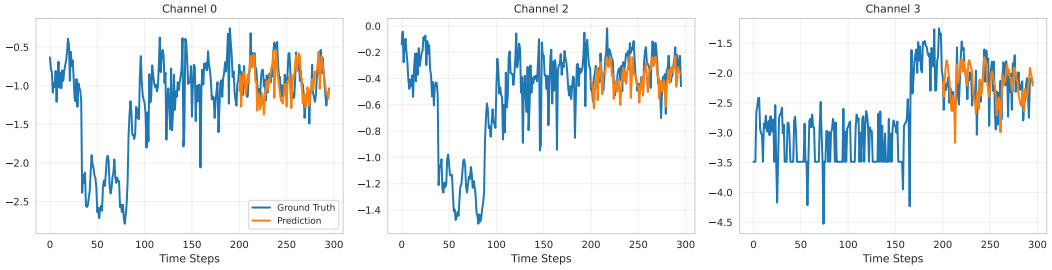

Figure 10: Forecast showcases of MoHETS across different time channels from **ETTh2**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.

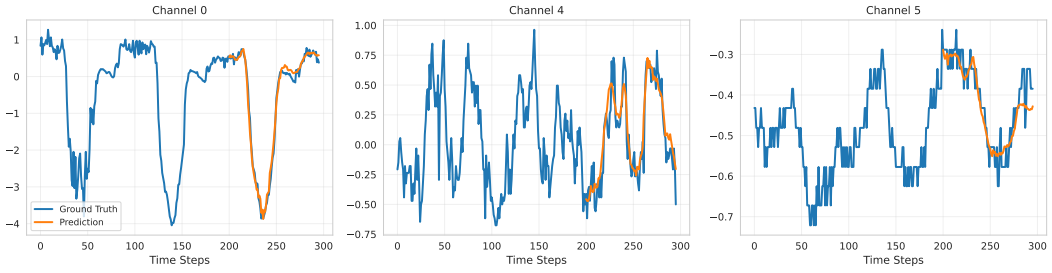

Figure 11: Forecast showcases of MoHETS across different time channels from **ETTm1**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.

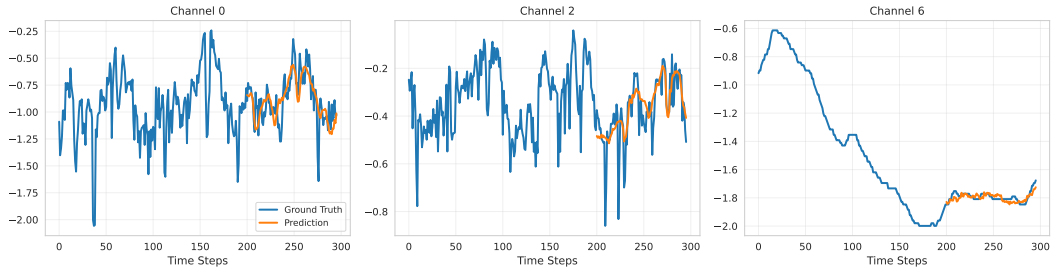

Figure 12: Forecast showcases of MOHETS across different time channels from **ETTm2**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.

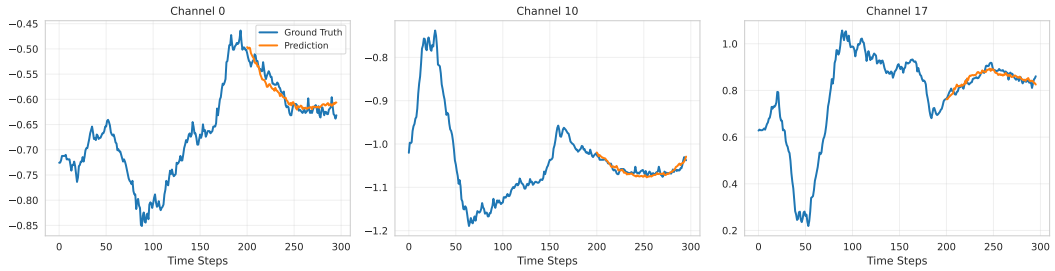

Figure 13: Forecast showcases of MOHETS across different time channels from **Weather**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.

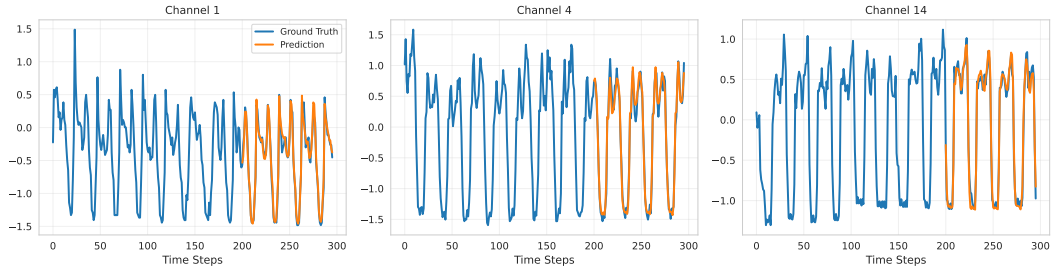

Figure 14: Forecast showcases of MOHETS across different time channels from **ECL**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.

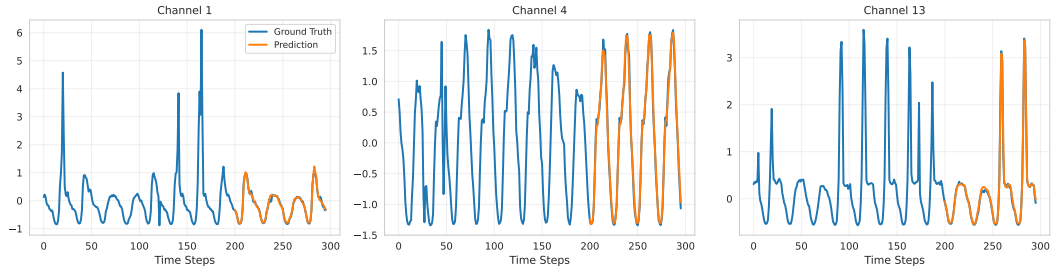

Figure 15: Forecast showcases of MOHETS across different time channels from **Traffic**, with a horizon of 96. Blue curves are the ground truths, and orange curves are the model predictions. Curves before the model predictions are the input data.