# Scaling to Billion Parameters for Time Series Foundation Models with Mixture of Experts

**Xiaoming Shi**[*♠], **Shiyu Wang**[*♠], **Yuqi Nie**[*1], **Dianqi Li, Zhou Ye, Qingsong Wen**[†2], **Ming Jin**[†♠3]
[1]Princeton University [2]Squirrel Ai Learning [3]Griffith University
sxm728@hotmail.com, kwuking@gmail.com, ynie@princeton.edu
{dianqili77, yezhou199032, qingsongedu, mingjinedu}@gmail.com

## Abstract

Deep learning has made significant strides in time series forecasting, yet the field lacks large-scale pre-trained models comparable to those in language and vision domains. In this paper, we introduce TIME-MOE, a scalable and unified architecture designed to pre-train larger, more capable forecasting foundation models while reducing inference costs. By leveraging a sparse mixture-of-experts (MoE) design, TIME-MOE enhances computational efficiency by activating only a subset of networks for each prediction while maintaining high model capacity. TIME-MOE comprises a family of decoder-only transformer models that operate in an auto-regressive manner and support arbitrary forecasting horizons with varying input context lengths. We pre-trained these models on large-scale data, spanning over 9 domains and encompassing over 300 billion time points. For the first time, we scaled a time series foundation model up to 2.4 billion parameters, achieving significantly improved forecasting precision. Our results validate the applicability of scaling laws for training tokens and model size in the context of time series forecasting. Compared to dense models with the same number of activated parameters or equivalent computation budgets, our models consistently outperform them by large margin. These advancements position TIME-MOE as a state-of-the-art solution for tackling real-world forecasting challenges with superior capability, efficiency, and flexibility.

## 1 Introduction

Time series data is a major modality in real-world dynamic systems and applications across various domains [52, 22]. Recent advancements have underscored the potential of foundation models (FMs) for universal forecasting [9, 45, 1]. However, the scaling laws—which posit that increasing model size and training tokens typically leads to performance improvements—have not been thoroughly investigated in the time series domain. More specifically, we aim to address a pivotal question: *How to scale time series foundation models to achieve universal forecasting while balancing model capability and computational overhead, mirroring the success of foundation models in other domains?*

Answering this question drives the design of TIME-MOE, a scalable and unified framework for pre-training larger, more efficient, and more capable forecasting FMs. Our key contributions include: 1) We explore sparse solutions for time series forecasting foundation models and their scaling properties; 2) We introduce TIME-MOE, a sparse, decoder-only time series foundation model architecture designed for scalable, universal forecasting, maximizing task-solving capability within a given computational budget; 3) We design a data-cleaning pipeline and curated the largest open time series dataset with over 300 billion time points across more than 9 domains, aimed at facilitating the pre-training of time series foundation models; 4) We present a family of open-source, pre-trained forecasting foundation models, scaling up to 2.4 billion parameters for the first time. TIME-MOE

[*] Equal contribution [♠] Project lead [†] Corresponding author

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

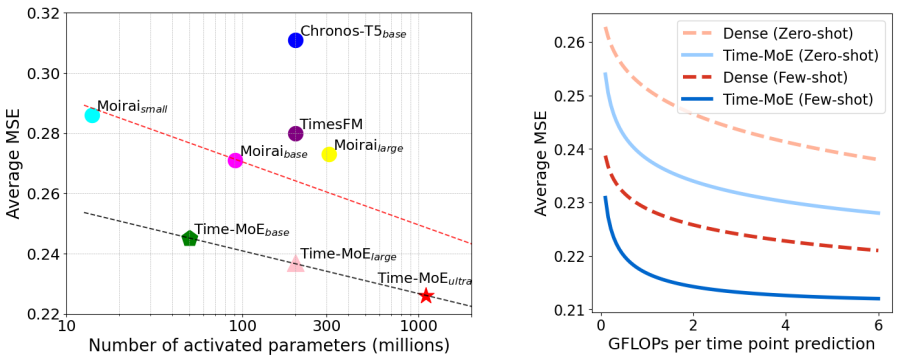


Figure 1: Performance overview. (**Left**) Comparison between TIME-MOE models and state-of-the-art time series foundation methods, reporting the average zero-shot performance across six benchmark datasets. (**Right**) Comparison of few- and zero-shot performance between TIME-MOE and dense variants, with similar effective FLOPs per time series token, across the same six benchmarks.

models outperform other time series foundation models by reducing squared error by 20% on average in zero-shot performance across six benchmarks, with a similar number of activated parameters.

## 2 Methodology

Our proposed TIME-MOE, illustrated in Figure 2, adopts a mixture-of-experts-based, decoder-only Transformer architecture, comprising three key components: (1) *input token embedding*, (2) *MoE Transformer block*, and (3) *multi-resolution output projection*.

**Input Token Embedding.** We utilize *point-wise tokenization* for time series embedding to ensure the completeness of temporal information. This enhances our model's flexibility and broad applicability in handling variable-length sequences. Then, we employ SwiGLU [36] to embed each time series point: $\mathbf{h}_t^0 = \text{SwiGLU}(x_t) = \text{Swish}\left(Wx_t\right) \otimes \left(Vx_t\right)$, where $W \in R^{D\times 1}$ and $V \in R^{D\times 1}$ are learnable parameters, and $D$ denotes the hidden dimension.

**MoE Transformer Block.** Our approach builds upon a decoder-only Transformer [40] and integrates recent advancements from large language models [3, 39]. We employ RMSNorm [51] to normalize the input of each Transformer sub-layer, thereby enhancing training stability. Instead of absolute positional encoding, we utilize rotary positional embedding [38], which offers greater flexibility in sequence length and better extrapolation capabilities. In line with [5], we remove biases from most layers but retain them in the QKV layer of self-attention to improve extrapolation. To introduce sparsity, we replace a standard feed-forward network (FFN) with a mixture-of-experts layer, incorporating a shared pool of experts that are sparsely activated.

$$\mathbf{u}_t^l = \text{SA}\left(\text{RMSNorm}\left(\mathbf{h}_t^{l-1}\right)\right) + \mathbf{h}_t^{l-1}, \tag{1}$$

$$\bar{\mathbf{u}}_t^l = \text{RMSNorm}\left(\mathbf{u}_t^l\right), \tag{2}$$

$$\mathbf{h}_t^l = \text{Mixture}\left(\bar{\mathbf{u}}_t^l\right) + \mathbf{u}_t^l. \tag{3}$$

Here, SA denotes self-attention with a causal mask, and Mixture refers to the mixture-of-experts layer. In practice, Mixture comprises several expert networks, each mirroring the architecture of a standard FFN. An individual time series point can be routed to either a single expert [12] or multiple experts [21]. Following [6], we designate one expert as a shared expert to capture and consolidate common knowledge across different contexts.

$$\text{Mixture}\left(\bar{\mathbf{u}}_t^l\right) = g_{N+1,t}\,\text{FFN}_{N+1}\left(\bar{\mathbf{u}}_t^l\right) + \sum_{i=1}^{N}\left(g_{i,t}\,\text{FFN}_i\left(\bar{\mathbf{u}}_t^l\right)\right), \tag{4}$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t}|1 \le j \le N\}, K), \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

$$g_{N+1,t} = \text{Sigmoid}\left(\mathbf{W}_N^l \bar{\mathbf{u}}_t^l\right), \tag{6}$$

$$s_{i,t} = \text{Softmax}_i\left(\mathbf{W}_i^l \mathbf{u}_t^l\right), \tag{7}$$

where $\mathbf{W}_i^l \in \mathbb{R}^{1\times D}$ denotes the trainable parameters, and $N$ and $K$ respectively denote the numbers of non-shared experts and activated non-shared experts per MoE layer.

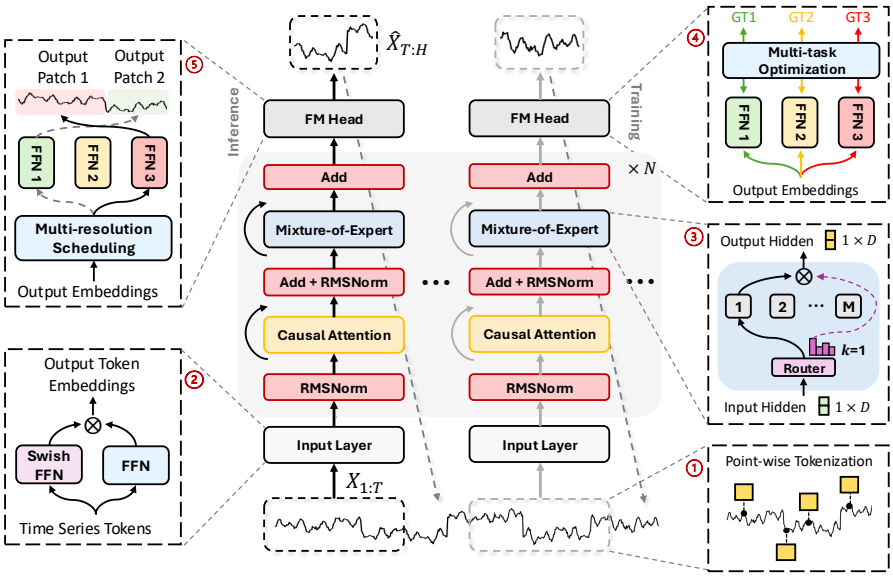


Figure 2: The architecture of TIME-MOE, which is a decoder-only model. Given an input time series of arbitrary length, ① we first tokenize it into a sequence of data points, ② which are then encoded. These tokens are processed through $N$-stacked backbone blocks, primarily consisting of causal multi-head self-attention and ③ sparse temporal mixture-of-expert layers. During training, ④ we optimize forecasting heads at multiple resolutions using multi-task learning. For model inference, TIME-MOE provides forecasts of arbitrary length by ⑤ dynamically scheduling these heads.

**Multi-resolution Forecasting.** We introduce a novel multi-resolution forecasting head, which allows for forecasting at multiple scales simultaneously, in contrast to existing foundation models that are limited to a single fixed scale. This capability enhances TIME-MOE 's flexibility by enabling it to generate forecasts across various horizons. The model uses multiple output projections from single-layer FFNs, each designed for different prediction horizons. By incorporating a simple greedy scheduling algorithm (see Appendix A.3), TIME-MOE efficiently handles predictions across arbitrary horizons. This design also boosts prediction robustness through multi-resolution ensemble learning. Additionally, TIME-MOE aggregates forecasting errors from different horizons to compute a composite loss (Appendix A.2), thereby improving the model generalization.

## 3 Main Results

TIME-MOE consistently outperforms state-of-the-art forecasting models by large margins across six well-established benchmarks and experimental settings, particularly in zero-shot scenarios. We compared TIME-MOE against a wide range of time series models, including several recently released foundation models for forecasting. Details of experiment settings and baselines are provided in Appendix A.4. To ensure a fair comparison, we adhered to the experimental configurations from [45] for out-of-distribution forecasting and [46] for in-distribution forecasting, applying a unified evaluation pipeline that we developed. More analysis about ablation, scalability and sparsification are provided in Appendix A.5, A.6, A.7.

**Zero-shot Forecasting.** We provide detailed results of zero-shot forecasting in Table 1. TIME-MOE achieves consistent state-of-the-art performances, improving a large margin as MSE reduction exceeding **20%** over the most competitive baselines. Importantly, as the model size scales (from `base` to `ultra`), it continuously exhibits enhanced performance across all datasets, affirming the efficacy of scaling laws within our time series foundation models. Furthermore, in comparisons with robust baselines possessing a similar number of activated parameters, TIME-MOE demonstrates significantly superior performance, underscoring the successful implementation of the MoE architecture in the large time series model.

**In-distribution Forecasting.** The full results are in Table 2. TIME-MOE exhibits remarkable capabilities, comprehensively surpassing advanced deep time series models from recent years, achieving an average MSE reduction of **24%**. Fine-tuning downstream data with only one epoch

Table 1: Full results of zero-shot forecasting experiments. A lower MSE or MAE indicates a better prediction. TimesFM, due to its use of Weather datasets in pretraining, is not evaluated on this dataset and is denoted by a dash (−). **Red**: the best, Blue: the 2nd best.

| Models | | Time-MoE (Ours) | | | | | | Zero-shot Time Series Models | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Time-MoE$_{base}$ | | Time-MoE$_{large}$ | | Time-MoE$_{ultra}$ | | Moirai$_{small}$ | | Moirai$_{base}$ | | Moirai$_{large}$ | | TimesFM | | Moment | | Chronos$_{small}$ | | Chronos$_{base}$ | | Chronos$_{large}$ | |
| 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 | 0.357 | 0.381 | 0.350 | 0.382 | **0.349** | **0.379** | 0.401 | 0.402 | 0.376 | 0.392 | 0.381 | 0.388 | 0.414 | 0.404 | 0.688 | 0.557 | 0.466 | 0.409 | 0.440 | 0.393 | 0.441 | 0.390 |
| | 192 | **0.384** | **0.404** | 0.388 | 0.412 | 0.395 | 0.413 | 0.435 | 0.421 | 0.412 | 0.413 | 0.434 | 0.415 | 0.465 | 0.434 | 0.688 | 0.560 | 0.530 | 0.450 | 0.492 | 0.426 | 0.502 | 0.424 |
| | 336 | **0.411** | 0.434 | **0.411** | 0.430 | 0.447 | 0.453 | 0.438 | 0.434 | 0.433 | **0.428** | 0.495 | 0.445 | 0.503 | 0.456 | 0.675 | 0.563 | 0.570 | 0.486 | 0.550 | 0.462 | 0.576 | 0.467 |
| | 720 | 0.449 | 0.477 | **0.427** | 0.455 | 0.457 | 0.462 | 0.439 | 0.454 | 0.447 | **0.444** | 0.611 | 0.510 | 0.511 | 0.481 | 0.683 | 0.585 | 0.615 | 0.543 | 0.882 | 0.591 | 0.835 | 0.583 |
| ETTh2 | 96 | 0.305 | 0.359 | 0.302 | 0.354 | **0.292** | 0.352 | 0.297 | 0.336 | 0.294 | **0.330** | 0.296 | **0.330** | 0.315 | 0.349 | 0.342 | 0.396 | 0.307 | 0.356 | 0.308 | 0.343 | 0.320 | 0.345 |
| | 192 | 0.351 | 0.386 | 0.364 | 0.385 | **0.347** | 0.379 | 0.368 | 0.381 | 0.365 | 0.375 | 0.361 | **0.371** | 0.388 | 0.395 | 0.354 | 0.402 | 0.376 | 0.401 | 0.384 | 0.392 | 0.406 | 0.399 |
| | 336 | 0.391 | 0.418 | 0.417 | 0.425 | 0.406 | 0.419 | 0.370 | 0.393 | 0.376 | **0.390** | 0.390 | **0.390** | 0.422 | 0.427 | **0.356** | 0.407 | 0.408 | 0.431 | 0.429 | 0.430 | 0.492 | 0.453 |
| | 720 | 0.419 | 0.454 | 0.537 | 0.496 | 0.439 | 0.447 | 0.411 | 0.426 | 0.416 | 0.433 | 0.423 | **0.418** | 0.443 | 0.454 | **0.395** | 0.434 | 0.604 | 0.533 | 0.501 | 0.477 | 0.603 | 0.511 |
| ETTm1 | 96 | 0.338 | 0.368 | 0.309 | 0.357 | **0.281** | **0.341** | 0.418 | 0.392 | 0.363 | 0.356 | 0.380 | 0.361 | 0.361 | 0.370 | 0.654 | 0.527 | 0.511 | 0.423 | 0.454 | 0.408 | 0.457 | 0.403 |
| | 192 | 0.353 | 0.388 | 0.346 | 0.381 | **0.305** | **0.358** | 0.431 | 0.405 | 0.388 | 0.375 | 0.412 | 0.383 | 0.414 | 0.405 | 0.662 | 0.532 | 0.618 | 0.485 | 0.567 | 0.477 | 0.530 | 0.450 |
| | 336 | 0.381 | 0.413 | 0.373 | 0.408 | **0.369** | 0.395 | 0.433 | 0.412 | 0.416 | **0.392** | 0.436 | 0.400 | 0.445 | 0.429 | 0.672 | 0.537 | 0.683 | 0.524 | 0.662 | 0.525 | 0.577 | 0.481 |
| | 720 | 0.504 | 0.493 | 0.475 | 0.477 | 0.469 | 0.472 | 0.462 | 0.432 | **0.460** | **0.418** | 0.462 | 0.420 | 0.512 | 0.471 | 0.692 | 0.551 | 0.748 | 0.566 | 0.900 | 0.591 | 0.660 | 0.526 |
| ETTm2 | 96 | 0.201 | 0.291 | **0.197** | 0.286 | 0.198 | 0.288 | 0.214 | 0.288 | 0.205 | 0.273 | 0.211 | 0.274 | 0.202 | **0.270** | 0.260 | 0.335 | 0.209 | 0.291 | 0.199 | 0.274 | **0.197** | 0.271 |
| | 192 | 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 | 0.321 | 0.289 | 0.350 | 0.280 | 0.341 | 0.261 | 0.322 | 0.254 | 0.314 |
| | 336 | 0.324 | 0.373 | 0.337 | 0.375 | **0.293** | **0.348** | 0.331 | 0.362 | 0.329 | 0.350 | 0.341 | 0.355 | 0.360 | 0.366 | 0.324 | 0.369 | 0.354 | 0.390 | 0.326 | 0.366 | 0.313 | 0.353 |
| | 720 | 0.488 | 0.464 | 0.480 | 0.461 | 0.427 | 0.428 | 0.402 | **0.408** | 0.437 | 0.411 | 0.485 | 0.428 | 0.462 | 0.430 | **0.394** | 0.409 | 0.553 | 0.499 | 0.455 | 0.439 | 0.416 | 0.415 |
| Weather | 96 | 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 | 0.211 | 0.243 | 0.203 | 0.238 | 0.194 | 0.235 |
| | 192 | 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 | 0.263 | 0.294 | 0.256 | 0.290 | 0.249 | 0.285 |
| | 336 | 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 | 0.321 | 0.339 | 0.314 | 0.336 | 0.302 | 0.327 |
| | 720 | 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 | 0.404 | 0.397 | 0.397 | 0.396 | 0.372 | 0.378 |
| Global Temp | 96 | 0.211 | 0.343 | **0.210** | **0.342** | 0.214 | 0.345 | 0.227 | 0.354 | 0.224 | 0.351 | 0.224 | 0.351 | 0.255 | 0.375 | 0.363 | 0.472 | 0.234 | 0.361 | 0.230 | 0.355 | 0.228 | 0.354 |
| | 192 | 0.257 | 0.386 | 0.254 | 0.385 | **0.246** | **0.379** | 0.269 | 0.396 | 0.266 | 0.394 | 0.267 | 0.395 | 0.313 | 0.423 | 0.387 | 0.489 | 0.276 | 0.400 | 0.273 | 0.395 | 0.276 | 0.398 |
| | 336 | 0.281 | 0.405 | 0.267 | **0.395** | **0.266** | 0.398 | 0.292 | 0.419 | 0.296 | 0.420 | 0.291 | 0.417 | 0.362 | 0.460 | 0.430 | 0.517 | 0.314 | 0.431 | 0.324 | 0.434 | 0.327 | 0.437 |
| | 720 | 0.354 | 0.465 | 0.289 | **0.420** | **0.288** | 0.421 | 0.351 | 0.437 | 0.403 | 0.498 | 0.387 | 0.488 | 0.486 | 0.545 | 0.582 | 0.617 | 0.418 | 0.504 | 0.505 | 0.542 | 0.472 | 0.535 |
| **Average** | | 0.336 | 0.384 | 0.336 | 0.380 | **0.322** | 0.372 | 0.349 | 0.377 | 0.347 | **0.370** | 0.359 | 0.373 | 0.396 | 0.413 | 0.461 | 0.454 | 0.428 | 0.420 | 0.429 | 0.412 | 0.416 | 0.405 |
| 1st Count | | 3 | | 7 | | **23** | | 1 | | 8 | | 8 | | 1 | | 3 | | 0 | | 0 | | 1 | |

significantly enhances predictive performance. This demonstrates the extraordinary potential of large time series models based on the MoE architecture. Similar to zero-shot forecasting results, as the model size increases, the scaling law continues to be effective, leading to continuous improvements in the performance of the Time-MoE.

Table 2: Full results of in-domain forecasting experiments. A lower MSE or MAE indicates a better prediction. Full-shot results besides Global Temp are from [24]. **Red**: the best, Blue: the 2nd best.

| Models | | Time-MoE (Fine-tuned) | | | | | | Full-shot Time Series Models | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Time-MoE$_{base}$ | | Time-MoE$_{large}$ | | Time-MoE$_{ultra}$ | | iTransformer | | TimeMixer | | TimesNet | | PatchTST | | Crossformer | | TiDE | | DLinear | | FEDformer | |
| 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 | 0.345 | 0.373 | 0.335 | 0.371 | **0.323** | **0.365** | 0.386 | 0.405 | 0.375 | 0.400 | 0.384 | 0.402 | 0.414 | 0.419 | 0.423 | 0.448 | 0.479 | 0.464 | 0.386 | 0.400 | 0.376 | 0.419 |
| | 192 | 0.372 | 0.396 | 0.374 | 0.400 | **0.359** | **0.391** | 0.441 | 0.436 | 0.436 | 0.429 | 0.421 | 0.429 | 0.460 | 0.445 | 0.471 | 0.474 | 0.525 | 0.492 | 0.437 | 0.432 | 0.420 | 0.448 |
| | 336 | 0.389 | **0.412** | 0.390 | **0.412** | **0.388** | 0.418 | 0.487 | 0.458 | 0.484 | 0.458 | 0.491 | 0.469 | 0.501 | 0.466 | 0.570 | 0.546 | 0.565 | 0.515 | 0.481 | 0.459 | 0.459 | 0.465 |
| | 720 | 0.410 | 0.443 | **0.402** | **0.433** | 0.425 | 0.450 | 0.503 | 0.491 | 0.498 | 0.482 | 0.521 | 0.500 | 0.500 | 0.488 | 0.653 | 0.621 | 0.594 | 0.558 | 0.519 | 0.516 | 0.506 | 0.507 |
| ETTh2 | 96 | 0.276 | 0.340 | 0.278 | **0.335** | **0.274** | 0.338 | 0.297 | 0.349 | 0.289 | 0.341 | 0.340 | 0.374 | 0.302 | 0.348 | 0.745 | 0.584 | 0.400 | 0.440 | 0.333 | 0.387 | 0.358 | 0.397 |
| | 192 | 0.331 | 0.371 | 0.345 | 0.373 | **0.330** | **0.370** | 0.380 | 0.400 | 0.372 | 0.392 | 0.402 | 0.414 | 0.388 | 0.400 | 0.877 | 0.656 | 0.528 | 0.509 | 0.477 | 0.476 | 0.429 | 0.439 |
| | 336 | 0.373 | 0.402 | 0.384 | 0.402 | **0.362** | **0.396** | 0.428 | 0.432 | 0.386 | 0.414 | 0.452 | 0.541 | 0.426 | 0.433 | 1.043 | 0.731 | 0.643 | 0.571 | 0.594 | 0.541 | 0.496 | 0.487 |
| | 720 | 0.404 | 0.431 | 0.437 | 0.437 | **0.370** | **0.417** | 0.427 | 0.445 | 0.412 | 0.434 | 0.462 | 0.657 | 0.431 | 0.446 | 1.104 | 0.763 | 0.874 | 0.679 | 0.831 | 0.657 | 0.463 | 0.474 |
| ETTm1 | 96 | 0.286 | 0.334 | 0.264 | 0.325 | **0.256** | **0.323** | 0.334 | 0.368 | 0.320 | 0.357 | 0.338 | 0.375 | 0.329 | 0.367 | 0.404 | 0.426 | 0.364 | 0.387 | 0.345 | 0.372 | 0.379 | 0.419 |
| | 192 | 0.307 | 0.358 | 0.295 | 0.350 | **0.281** | **0.343** | 0.377 | 0.391 | 0.361 | 0.381 | 0.374 | 0.387 | 0.367 | 0.385 | 0.450 | 0.451 | 0.398 | 0.404 | 0.380 | 0.389 | 0.426 | 0.441 |
| | 336 | 0.354 | 0.390 | **0.323** | 0.376 | 0.326 | **0.374** | 0.426 | 0.420 | 0.390 | 0.404 | 0.410 | 0.411 | 0.399 | 0.410 | 0.532 | 0.515 | 0.428 | 0.425 | 0.413 | 0.413 | 0.445 | 0.459 |
| | 720 | 0.433 | 0.445 | **0.409** | **0.435** | 0.454 | 0.452 | 0.491 | 0.459 | 0.454 | 0.441 | 0.478 | 0.450 | 0.454 | 0.439 | 0.666 | 0.589 | 0.487 | 0.461 | 0.474 | 0.453 | 0.543 | 0.490 |
| ETTm2 | 96 | 0.172 | 0.265 | **0.169** | 0.259 | 0.183 | 0.273 | 0.180 | 0.264 | 0.175 | **0.258** | 0.187 | 0.267 | 0.175 | 0.259 | 0.287 | 0.366 | 0.207 | 0.305 | 0.193 | 0.292 | 0.203 | 0.287 |
| | 192 | 0.228 | 0.306 | **0.223** | **0.295** | **0.223** | 0.301 | 0.250 | 0.309 | 0.237 | 0.299 | 0.249 | 0.309 | 0.241 | 0.302 | 0.414 | 0.492 | 0.290 | 0.364 | 0.284 | 0.362 | 0.269 | 0.328 |
| | 336 | 0.281 | 0.345 | 0.293 | 0.341 | **0.278** | **0.339** | 0.311 | 0.348 | 0.298 | 0.340 | 0.321 | 0.351 | 0.305 | 0.343 | 0.597 | 0.542 | 0.377 | 0.422 | 0.369 | 0.427 | 0.325 | 0.366 |
| | 720 | 0.403 | 0.424 | 0.451 | 0.433 | 0.425 | 0.424 | 0.412 | 0.407 | **0.391** | **0.396** | 0.408 | 0.403 | 0.402 | 0.400 | 1.730 | 1.042 | 0.558 | 0.524 | 0.554 | 0.522 | 0.421 | 0.415 |
| Weather | 96 | 0.151 | 0.203 | **0.149** | **0.201** | 0.154 | 0.208 | 0.174 | 0.214 | 0.163 | 0.209 | 0.172 | 0.220 | 0.177 | 0.218 | 0.158 | 0.230 | 0.202 | 0.261 | 0.196 | 0.255 | 0.217 | 0.296 |
| | 192 | 0.195 | 0.246 | **0.192** | **0.244** | 0.202 | 0.251 | 0.221 | 0.254 | 0.208 | 0.250 | 0.219 | 0.261 | 0.225 | 0.259 | 0.206 | 0.277 | 0.242 | 0.298 | 0.237 | 0.296 | 0.276 | 0.336 |
| | 336 | 0.247 | 0.288 | **0.245** | **0.285** | 0.252 | 0.287 | 0.278 | 0.296 | 0.251 | 0.287 | 0.280 | 0.306 | 0.278 | 0.297 | 0.272 | 0.335 | 0.287 | 0.335 | 0.283 | 0.335 | 0.339 | 0.380 |
| | 720 | 0.352 | 0.366 | 0.352 | 0.365 | 0.392 | 0.376 | 0.358 | 0.349 | **0.339** | **0.341** | 0.365 | 0.359 | 0.354 | 0.348 | 0.398 | 0.418 | 0.351 | 0.386 | 0.345 | 0.381 | 0.403 | 0.428 |
| Global Temp | 96 | 0.192 | 0.328 | 0.192 | 0.329 | **0.189** | **0.322** | 0.223 | 0.351 | 0.215 | 0.346 | 0.250 | 0.381 | 0.219 | 0.349 | 0.272 | 0.406 | 0.223 | 0.352 | 0.221 | 0.354 | 0.261 | 0.392 |
| | 192 | 0.238 | **0.375** | 0.236 | **0.375** | **0.234** | 0.376 | 0.282 | 0.404 | 0.266 | 0.393 | 0.298 | 0.418 | 0.269 | 0.395 | 0.305 | 0.435 | 0.278 | 0.401 | 0.257 | 0.388 | 0.299 | 0.423 |
| | 336 | 0.259 | **0.397** | 0.256 | **0.397** | **0.253** | 0.399 | 0.313 | 0.431 | 0.313 | 0.430 | 0.315 | 0.434 | 0.319 | 0.435 | 0.352 | 0.468 | 0.330 | 0.440 | 0.294 | 0.418 | 0.341 | 0.454 |
| | 720 | 0.345 | 0.465 | 0.322 | 0.451 | **0.292** | **0.426** | 0.393 | 0.488 | 0.468 | 0.536 | 0.407 | 0.497 | 0.452 | 0.526 | 0.508 | 0.562 | 0.485 | 0.544 | 0.380 | 0.479 | 0.359 | 0.469 |
| **Average** | | 0.306 | 0.362 | 0.304 | 0.359 | **0.301** | **0.358** | 0.349 | 0.382 | 0.337 | 0.375 | 0.356 | 0.400 | 0.349 | 0.382 | 0.560 | 0.516 | 0.421 | 0.439 | 0.387 | 0.416 | 0.375 | 0.417 |
| 1st Count | | 3 | | 18 | | **28** | | 0 | | 5 | | 0 | | 0 | | 0 | | 0 | | 0 | | 0 | |

## 4 Conclusion

In this paper, we present Time-MoE, a scalable architecture for time series foundation models that uses sparse MoE to improve computational efficiency without sacrificing model capacity. By pre-training on a vast dataset, we scaled our model to 2.4 billion parameters, activating 1.1 billion of them, demonstrating significant improvements in forecasting precision. Our results validate the applicability of scaling laws to time series forecasting and show that Time-MoE consistently outperforms dense models with equivalent computational budgets across multiple benchmarks. The multi-resolution forecasting capability and superior performance in both zero-shot and fine-tuned scenarios position Time-MoE as a state-of-the-art solution for tackling real-world forecasting challenges. This work paves the way for future advancements in scaling and constructing more efficient time series FMs.

## References


[1] Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., et al. (2024). Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*.

[2] Assimakopoulos, V. and Nikolopoulos, K. (2000). The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4):521–530.

[3] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. (2023). Qwen technical report. *arXiv preprint arXiv:2309.16609*.

[4] Chen, P., Zhang, Y., Cheng, Y., Shu, Y., Wang, Y., Wen, Q., Yang, B., and Guo, C. (2024). Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *International Conference on Learning Representations*.

[5] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

[6] Dai, D., Deng, C., Zhao, C., Xu, R., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., et al. (2024). Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

[7] Dao, T. (2024). FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

[8] Das, A., Kong, W., Leach, A., Mathur, S. K., Sen, R., and Yu, R. (2023). Long-term forecasting with tide: Time-series dense encoder. *Transactions on Machine Learning Research*.

[9] Das, A., Kong, W., Sen, R., and Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

[10] Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. (2019). Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32.

[11] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[12] Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

[13] Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., and Montero-Manso, P. (2021). Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*.

[14] Goerg, G. (2013). Forecastable component analysis. *ICML*.

[15] Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., and Dubrawski, A. (2024). Moment: A family of open time-series foundation models. In *Forty-first International Conference on Machine Learning*.

[16] Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

[17] Hu, J., Hu, Y., Chen, W., Jin, M., Pan, S., Wen, Q., and Liang, Y. (2024). Attractor memory for long-term time series forecasting: A chaos perspective. *arXiv preprint arXiv:2402.11463*.

[18] Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer.

[19] Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. (2008). *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

[20] Jin, M., Zheng, Y., Li, Y.-F., Chen, S., Yang, B., and Pan, S. (2022). Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9168–9180.

[21] Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2020). Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

[22] Liang, Y., Wen, H., Nie, Y., Jiang, Y., Jin, M., Song, D., Pan, S., and Wen, Q. (2024). Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6555–6565.

[23] Lin, S., Lin, W., Wu, W., Chen, H., and Yang, J. (2024). Sparsetsf: Modeling long-term time series forecasting with* 1k* parameters. In *Forty-first International Conference on Machine Learning*.

[24] Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. (2024a). itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*.

[25] Liu, Y., Zhang, H., Li, C., Huang, X., Wang, J., and Long, M. (2024b). Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning*.

[26] Ni, R., Lin, Z., Wang, S., and Fanti, G. (2024). Mixture-of-linear-experts for long-term time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, pages 4672–4680. PMLR.

[27] Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*.

[28] Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2020). N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.

[29] Qi, S., Xu, Z., Li, Y., Wen, L., Wen, Q., Wang, Q., and Qi, Y. (2024). Pdetime: Rethinking long-term multivariate time series forecasting from the perspective of partial differential equations. *arXiv preprint arXiv:2402.16913*.

[30] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[31] Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31.

[32] Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. (2021). Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595.

[33] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191.

[34] Selva, J., Johansen, A. S., Escalera, S., Nasrollahi, K., Moeslund, T. B., and Clapés, A. (2023). Video transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12922–12943.

[35] Sen, R., Yu, H.-F., and Dhillon, I. S. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32.

[36] Shazeer, N. (2020). Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

[37] Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). The sparsely-gated mixture-of-experts layer. *Outrageously large neural networks*.

[38] Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

[39] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[40] Vaswani, A. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[41] Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J. Y., and ZHOU, J. (2024a). Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*.

[42] Wang, X., Zhou, T., Wen, Q., Gao, J., Ding, B., and Jin, R. (2024b). Card: Channel aligned robust blend transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations (ICLR)*.

[43] Wen, Q., Gao, J., Song, X., Sun, L., and Tan, J. (2019). 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*, pages 3856–3862.

[44] Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. (2023). Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6778–6786.

[45] Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. (2024). Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*.

[46] Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. (2023). Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*.

[47] Yang, Y., Zhang, C., Zhou, T., Wen, Q., and Sun, L. (2023). Dcdetector: Dual attention contrastive representation learning for time series anomaly detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3033–3045.

[48] Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. (2022). Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987.

[49] Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128.

[50] Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. (2021). A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2114–2124.

[51] Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

[52] Zhang, K., Wen, Q., Zhang, C., Cai, R., Jin, M., Liu, Y., Zhang, J. Y., Liang, Y., Pang, G., Song, D., et al. (2024). Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[53] Zhang, X., Zhao, Z., Tsiligkaridis, T., and Zitnik, M. (2022). Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35:3988–4003.

[54] Zhang, Y. and Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*.

[55] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.

[56] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022). FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*.

# A Appendix

## A.1 Related Work

**Time Series Forecasting.** Deep learning models have become powerful tools for time series forecasting over the past decade. These models outperform traditional, local statistical methods, such as Theta [2] and ARIMA [19], which fit separate models to each time series independently. In contrast, deep learning models learn from data across multiple time series within a dataset, which can be broadly categorized into two types: (1) *univariate models*, such as DeepState [31], DeepAR [33], and N-BEATS [28], which focus on modeling individual time series, and (2) *multivariate models*, which include both transformer-based approaches [44, 55, 27, 24, 42, 4] and non-transformer models [35, 20, 41, 17, 29], designed to handle multiple time series simultaneously. While these models achieve competitive in-domain performance, many are task-specific and fall short in generalizability when applied to cross-domain data in few-shot or zero-shot scenarios.

**Large Time Series Models.** Pre-training on large-scale sequence data has significantly advanced modality understanding in language and vision domains [10, 34]. Building on this progress, self-supervised learning has been extensively developed for time series [52], employing masked reconstruction [50, 27] or contrastive learning [53, 48, 47]. However, these methods are often limited in both data and model scale, with many focused on in-domain learning and transfer. Recently, general pre-training of time series models on large-scale datasets has emerged, though still in its early stages. Current efforts toward universal forecasting are in three categories: (1) *encoder-only models*, such as Moirai [45] and Moment [15], which use masked reconstruction and have been pre-trained on datasets with 27B and 1B time points, with model parameters up to 385M; (2) *encoder-decoder models*, represented by Chronos [1], which has been pre-trained on approximately 796K time series, containing up to 710M parameters; and (3) *decoder-only models*, including TimesFM [9] and Timer [25], where the most extensive models feature up to 200M parameters. Unlike these dense models, TIME-MOE introduces a scalable and unified framework for pre-training larger forecasting models, scaling up to 2.4B parameters and trained on over 300B time points. TIME-MOE is more capable while maintaining the same scale of activated parameters or computational budget as dense models.

**Sparse Deep Learning for Time Series.** Traditional deep learning models are dense and often over-parameterized [16], resulting in increased memory and computational demands during training and inference. However, in many fields, sparse networks, such as mixture-of-experts, have demonstrated comparable or superior generalization compared to dense models, while being more efficient [12, 32]. In time series research, model sparsification has received less attention, as time series models have traditionally been small in scale, with simple models like DLinear [49] and SparseTSF [23] excelling in specific tasks prior to the advent of large-scale, general pre-training. The most relevant recent work is MoLE [26], which trains multiple linear models to collaboratively predict time series. However, unlike TIME-MOE, where dense feed-forward layers in the transformer backbone are replaced with sparsely-activated subnetworks, MoLE is not a sparse model, as input data is passed to all heads and then combined to make predictions.

## A.2 Model Training

**Pre-training Data** Training time series foundation models requires extensive, high-quality data. However, large-scale datasets that are adequately processed remain scarce. Recent advancements have facilitated the collection of numerous time series datasets from various sources [13, 1, 45, 25]. Nonetheless, data quality remains a challenge, with prevalent issues such as *missing values* and *invalid observations* that can impair model performance and destabilize training. To mitigate these issues, we have developed a streamlined *data-cleaning pipeline* (see Appendix A.3) to filter and refine raw data, thereby constructing a collection of high-quality, large-scale time series datasets for model pre-training. We have collected a diverse range of publicly available datasets, including those from Chronos [1], LOTSA [45] and UTSD [25], spanning multiple domains such as energy, retail, healthcare, weather, finance, transportation, and web. We also include a portion of synthetic data to enhance the quantity and diversity of our training data. These datasets vary in sampling frequencies from 5 minutes to yearly and encompass over 300 billion time points in total.

Table 3: Key statistics of pre-training data across domains

| | Energy | Finance | Healthcare | Nature | Sales | Synthetic | Transport | Web | Other | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| **# Seqs.** | 1,817,756 | 1,683 | 2,275 | 3,257,621 | 95,402 | 7,406,529 | 101,011 | 1,069,565 | 356,147 | 14,107,989 |
| **# Obs.** | 16.133 B | 385.024 K | 293.888 K | 88.868 B | 5.323 M | 8.087 B | 1.271 B | 1.818 B | 899.713 M | 117.083 B |
| **%** | 13.791% | 0.0003% | 0.0002% | 75.881% | 0.004% | 6.913% | 1.086% | 1.554% | 0.769% | 100% |

**Loss Function** Pre-training time series foundation models in large scale presents significant challenges in training stability due to the massive datasets and the vast number of parameters involved. To address this, we use the Huber loss [18, 43], which provides greater robustness to outliers and improves training stability.

$$\mathcal{L}_{\text{ar}}\left(x_t, \hat{x}_t\right) = \begin{cases} \frac{1}{2}\left(x_t - \hat{x}_t\right)^2, & \text{if } |x_t - \hat{x}_t| \leq \delta, \\ \delta \times \left(|x_t - \hat{x}_t| - \frac{1}{2} \times \delta\right), & \text{otherwise}, \end{cases} \tag{8}$$

where $\delta$ is a hyperparameter that balances the L1 and L2 loss components. When training the model with a Mixture-of-Experts architecture, focusing solely on optimizing prediction error often leads to load imbalance issues among the experts. A common problem is routing collapse [37], where the model predominantly selects only a few experts, limiting training opportunities for others. To mitigate this, following the approaches of [6, 12], we achieve expert-level balancing with an auxiliary loss to reduce routing collapse. The auxiliary loss is computed as follows:

$$\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i r_i, \quad f_i = \frac{1}{KT} \sum_{t=1}^{T} \mathbb{I}\left(\text{Time point } t \text{ selects Expert } i\right), \quad r_i = \frac{1}{T} \sum_{t=1}^{T} s_{i,t}, \tag{9}$$

where $f_i$ represents the fraction of tokens assigned to expert $i$, and $r_i$ denotes the proportion of router probability allocated to expert $i$. $\mathbb{I}$ is the indicator function. Finally, we combine the auto-regressive losses across all multi-resolution projections with the auxiliary balance loss to form the final loss:

$$\mathcal{L} = \frac{1}{P} \sum_{i=1}^{P} \mathcal{L}_{\text{ar}}\left(\mathbf{X}_{t+1:t+p_i}, \hat{\mathbf{X}}_{t+1:t+p_i}\right) + \alpha \mathcal{L}_{\text{aux}}, \tag{10}$$

**Training Details** Informed by the scaling laws demonstrated by Llama [11, 39], which show that a 7- or 8-billion parameter model continues to improve performance even after training on over one trillion tokens, we chose to scale our model to 1 billion activated parameters. Notably, TIME-MOE$_{ultra}$ supports inference on consumer-grade GPUs with less than 8GB of VRAM. We have also developed two smaller models: TIME-MOE$_{base}$, with 50 million activated parameters, and TIME-MOE$_{large}$, with 200 million activated parameters, both specifically designed for fast inference on CPU architectures. These streamlined models are strategically developed to ensure broader accessibility and applicability in resource-constrained environments. Each model undergoes training for $100,000$ steps with a batch size of $1024$, where the maximum sequence length is capped at $4096$. This setup results in the consumption of $4$ million time points per iteration. We choose $\{1, 8, 32, 64\}$ as different forecast horizons in the output projection and set the factor of the auxiliary loss $\alpha$ to $0.02$. For optimization, we employ the AdamW optimizer, configured with the following hyperparameters: lr = 1e-3, weight_decay = $0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.95$. A learning rate scheduler with a linear warmup over the initial $10,000$ steps followed by cosine annealing is also utilized. Training is conducted on 128 $\times$ A100-80G GPUs using BF16 precision. To enhance batch processing efficiency and accommodate varying sequence lengths, we implement sequence packing [30], significantly reducing the need for extensive padding.

Table 4: A high-level summary of model configurations.

| | Layers | Heads | Experts | $K$ | $d_{\text{model}}$ | $d_{\text{ff}}$ | $d_{\text{expert}}$ | Activated Params | Total Params |
|---|---|---|---|---|---|---|---|---|---|
| TIME-MOE$_{base}$ | 12 | 12 | 8 | 2 | 384 | 1536 | 192 | 50 M | 113 M |
| TIME-MOE$_{large}$ | 12 | 12 | 8 | 2 | 768 | 3072 | 384 | 200 M | 453 M |
| TIME-MOE$_{ultra}$ | 36 | 16 | 8 | 2 | 1024 | 4096 | 512 | 1.1 B | 2.4 B |

### A.3 Implementation Details

**Missing Value Processing.** In time-series observations, missing values frequently manifest as 'NaN' (Not a Number) or 'Inf' (Infinity). Previous studies often addressed this issue by replacing

missing values with the mean value, which we believe can distort the original pattern of the time series. Instead, we utilize a method that divides the original sequence into multiple sub-sequences at points where missing values occur. This approach effectively removes the segments with missing values while preserving the integrity of the original time-series pattern.

**Invalid Observation Processing.** In some data collection systems, missing values are often filled with 0 or another constant value, resulting in sequences that contain constant values. These values represent invalid patterns for the model. Consequently, we have developed a filtering method that employs a fixed-length window to traverse the entire sequence. This method calculates the ratio of first-order to second-order differences within the window. The window sequence is discarded if the ratio exceeds a pre-specified threshold (0.2 in our setting). Subsequently, we concatenate the continuous, satisfactory window sequences into a single sequence. This process transforms the original sequence into multiple sub-sequences, effectively excluding segments with unacceptable window sequences.

**Scheduling Algorithm for Arbitrary Horizons.** We define $P$ output projections, each for a distinct forecasting horizon, denoted as $(p_1, p_2, \ldots, p_P)$. Each output projection corresponding to horizon $p_i$ is utilized to forecast the subsequent $p_i$ time steps, as shown below:

$$\hat{\mathbf{X}}_{t+1:t+p_i} = \mathbf{W}_{p_i} \mathbf{h}_t^L, \tag{11}$$

where $\mathbf{W}_{p_i} \in \mathbb{R}^{p_i \times D}$ is the learnable parameter matrix associated with that horizon, $\mathbf{h}_t^L$ denotes the output hidden state from the last MoE Transformer block. All output projections are optimized simultaneously in model training. During inference, for each prediction in the auto-regressive forecasting process, we select a projection $p_i$ with the closest forecasting horizon that is less than or equal to the required forecast length. This allows TIME-MOE to make predictions beyond the next time step or a fixed horizon, significantly enhancing the model's utility and forecasting performance.

### A.4 Experiment Details

**Problem Statement.** We address the problem of predicting future values in a time series: given a sequence of historical observations $\mathbf{X}_{1:T} = (x_1, x_2, \ldots, x_T) \in \mathbb{R}^T$ spanning $T$ time steps, our objective is to forecast the next $H$ time steps, i.e., $\hat{\mathbf{X}}_{T+1:T+H} = f_\theta(\mathbf{X}_{1:T}) \in \mathbb{R}^H$. Here, $f_\theta$ represents a time series model, where $T$ is the context length and $H$ is the forecasting horizon. Notably, both $T$ and $H$ are *arbitrary* when inferencing TIME-MOE, distinguishing it from task-specific models with fixed configurations. Additionally, channel independence [27] is adopted to transform a multivariate input into univariate series, allowing TIME-MOE to handle *any-variate* forecasting problems in real-world applications.

**Datasets Details.** We evaluate the performance of different models for long-term forecasting on 8 well-established datasets, including Weather, Electricity, and ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2). We detail the descriptions of the dataset in Table 5.

Table 5: Detailed dataset descriptions. Dataset sizes are listed as (Train, Validation, Test).

| Tasks | Dataset | Dim | Series Length | Dataset Size | Frequency | Forecastability∗ | Information |
|---|---|---|---|---|---|---|---|
| Long-term Forecasting | ETTm1 | 7 | {96, 192, 336, 720} | (34465, 11521, 11521) | 15min | 0.46 | Temperature |
| | ETTm2 | 7 | {96, 192, 336, 720} | (34465, 11521, 11521) | 15min | 0.55 | Temperature |
| | ETTh1 | 7 | {96, 192, 336, 720} | (8545, 2881, 2881) | Hourly | 0.38 | Temperature |
| | ETTh2 | 7 | {96, 192, 336, 720} | (8545, 2881, 2881) | Hourly | 0.45 | Temperature |
| | Weather | 21 | {96, 192, 336, 720} | (36792, 5271, 10540) | 10 min | 0.75 | Weather |
| | Global Temp | 1000 | {96, 192, 336, 720} | (12280, 1755, 3509) | Hourly | 0.78 | Temperature |

∗ The forecastability is calculated by one minus the entropy of Fourier decomposition of time series [14]. A larger value indicates better predictability.

**Metric Details.** Regarding metrics, we utilize the mean square error (MSE) and mean absolute error (MAE) for long-term forecasting. The calculations of these metrics are:

$$\text{RMSE} = (\sum_{i=1}^{H}(\mathbf{X}_i - \widehat{\mathbf{X}}_i)^2)^{\frac{1}{2}}, \qquad \text{MAE} = \sum_{i=1}^{H}|\mathbf{X}_i - \widehat{\mathbf{X}}_i|,$$

where $\mathbf{X}, \widehat{\mathbf{X}} \in \mathbb{R}^{H \times C}$ are the ground truth and prediction results of the future with $H$ time pints and $C$ dimensions. $\mathbf{X}_i$ means the $i$-th future time point.

**Baselines Details.** We evaluate TIME-MOE against 14 baselines, representing state-of-the-art models in long-term forecasting. These baselines are categorized into two groups. The first group, for zero-shot forecasting evaluation, includes pre-trained foundation models such as Moirai [2024], TimesFM [2024], Timer [2024b], Moment [2024], and Chronos [2024]. The second group, for in-distribution (full-shot) forecasting evaluation, consists of deep time series models such as iTransformer [2024a], TimeMixer [2024a], TimesNet [2023], PatchTST [2023], Crossformer [2023], TiDE [2023], DLinear [2023], and Fedformer [2022].

**Setup Details.** We use four different prediction horizons, which are $\{96, 192, 336, 720\}$, with the corresponding input time series lengths $\{512, 1024, 2048, 3072\}$. The evaluation metrics adopt mean square error (MSE) and mean absolute error (MAE). In *zero-shot forecasting* experiments, we conducted experiments on the six well-known long-term forecasting benchmarks for which datasets were not included in the pre-training corpora. Regarding *in-distribution forecasting*, we simultaneously fine-tune the pre-trained TIME-MOE models on the training datasets of the above-mentioned six benchmarks and set the number of epochs to 1. This implies that the model is trained for approximately only 200 steps.

### A.5 Ablation Study

Table 6: Ablation studies. (**Left**) Average MSE for horizon-96 forecasting across six benchmarks, evaluated with different model components. (**Right**) Analysis of various multi-resolution forecasting configurations.

| | Average MSE |
|---|---|
| TIME-MOE$_{base}$ | **0.262** |
| w/o Huber loss | 0.267 |
| w/o multi-resolution layer | 0.269 |
| w/o mixture-of-experts | 0.272 |
| w/o auxiliary loss | 0.275 |

| | Average MSE | Inference Speed |
|---|---|---|
| TIME-MOE$_{base}$ | **0.262** | **0.095 s/iter** |
| TIME-MOE$_{base}$ w/ {1,8,32} | 0.273 | 0.130 s/iter |
| TIME-MOE$_{base}$ w/ {1,8} | 0.320 | 0.411 s/iter |
| TIME-MOE$_{base}$ w/ {1} | 1.382 | 2.834 s/iter |

To assess the effectiveness of our design for large time series models, we performed a detailed ablation analysis on the key architectural components and loss functions across all experiment benchmarks. We have the following observations from Table 6.

**Model Architecture.** Replacing the MoE with a vanilla FFN led to a performance drop from $0.262$ to $0.272$, highlighting the significant advantage of the MoE architecture, as in the left of Table 6. Further comparisons between dense and sparse models are discussed in detail in Section A.6. In the TIME-MOE$_{base}$, we removed the multi-resolution output layers, retaining only the horizon-32 output layer and thus excluding the multi-task learning component. This modification resulted in a slight performance degradation compared to the original TIME-MOE$_{base}$. As illustrated on the right side of Table 6, the default configuration of four multi-resolution output projections with receptive horizons of 1, 8, 32, and 64 achieves the best balance between predictive performance and inference speed. Reducing the number of projections consistently worsens performance while significantly improving inference speed, underscoring the effectiveness of our multi-resolution output projection design.

**Training Loss.** We note that the model trained with Huber loss demonstrates enhanced performance over the one using MSE loss due to the superior robustness of Huber loss in handling outlier time points. We removed the auxiliary loss from the objective function, keeping only the auto-regressive loss while still employing the MoE architecture. This modification caused the expert layers of the model to collapse to a smaller FFN in training, as the activation score of the more effective expert became disproportionately stronger without the load balance loss. As a result, its performance was notably worse than the TIME-MOE$_{base}$.

Table 7: Comparison of BF16 and FP32 in terms of training and inference efficiency. FA denotes flash-attention.

| | Average MSE | Training Speed | Inference Speed | Training Memory | Inference Memory |
|---|---|---|---|---|---|
| TIME-MOE$_{base}$ | 0.262 | 0.84 s/iter | 0.095 s/iter | 1.77 GB | 226.70 MB |
| TIME-MOE$_{base}$ w/o FA | 0.262 | 1.09 s/iter | 0.118 s/iter | 1.77 GB | 226.70 MB |
| TIME-MOE$_{base}$ w/ FP32 | 0.261 | 1.24 s/iter | 0.133 s/iter | 2.21 GB | 453.41 MB |

### A.6 Scalability Analysis

**Dense and Sparse Models.** To evaluate the performance and efficiency advantages of the Mixture of Experts (MoE) architecture in time-series forecasting, we substituted the MoE layer with a dense layer having an equivalent number of parameters to the activated parameters of the MoE layer. Employing identical training settings and data, we trained three dense models corresponding to the sizes of the three TIME-MOE models. The zero-shot performance comparison between the dense and MoE models is presented on the left of Figure 3. Compared to Sparse models (MoE) and Dense models, we observe that our approach achieved an average reduction of **78%** in training costs and **39%** in inference costs. This greatly demonstrates the advantages of Time-MoE, especially in maintaining exceptional performance while offering extreme cost-effectiveness.

**Model and Data Scaling.** We save the checkpoints for each model at intervals of every 20 billion time points during training. This enables us to plot performance traces for models of different sizes trained across various data scales. The right of Figure 3 illustrates that models trained with more data generally exhibit better performance across all model sizes.

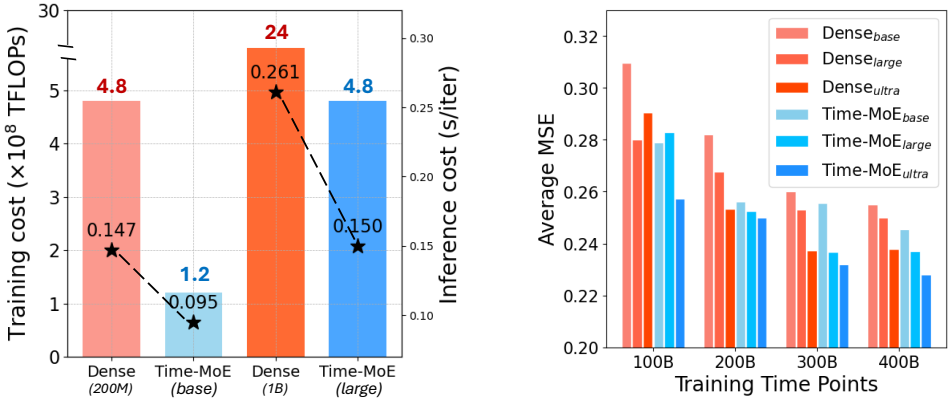


Figure 3: Scalability analysis. **(Left)** Comparison of dense models and TIME-MOE models in terms of training cost and inference cost. (**Right**) Average MSE for a 96-horizon forecast across six benchmarks, comparing TIME-MOE and the dense models, both trained from scratch with different training time points.

**Training Precision.** We trained a new model, TIME-MOE$_{base}$ (FP32), using identical configurations but with float32 precision instead of bfloat16. As shown in Table 7, the forecasting performance of both models is comparable. However, the bfloat16 model achieves a **12%** improvement in training speed and reduces memory consumption by **20%** compared to the float32 model. Moreover, the bfloat16 model can seamlessly integrate with flash-attention [7], further boosting training and inference speed by **23%** and **19%** respectively.

### A.7 Sparsification Analysis

As shown in Figure 4, we can observe that TIME-MOE activates different experts across various datasets, with each expert learning distinct knowledge. This results in different activation values on datasets from diverse domains. This also reveals why TIME-MOE, as a large time series foundation model, possesses remarkable transferability and generalization capabilities.

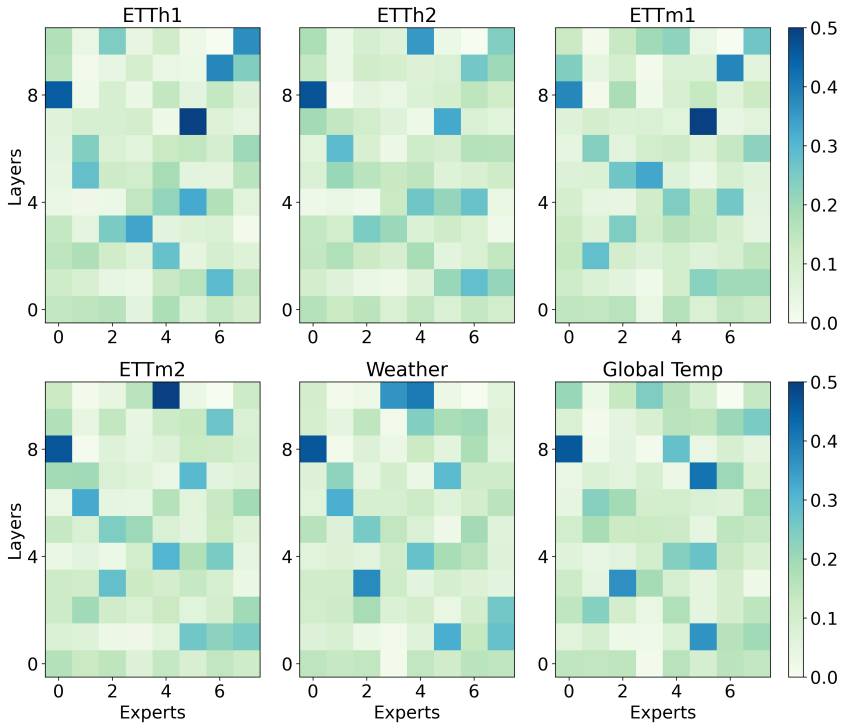


Figure 4: The gate scores for experts across different layers in the six benchmarks

Furthermore, we conducted a sensitivity analysis on the number of experts as $\text{top}_k$ within the MoE architecture, as shown in Table 8. As $k$ increases, it can be observed that there is no significant improvement in performance, and inference speed decreases markedly. This indicates that for the MoE architecture, sparsity not only does not degrade performance but also greatly enhances inference efficiency. This is crucial for training large models, as we must consider performance, efficiency, and cost. Models based on the sparse MoE architecture have inherent advantages in these aspects.

Table 8: Performance and inference speed among the different $\text{top}_k$. The average MSE for horizon-96 forecasting across six benchmarks.

| TIME-MOE$_{base}$ | **Average MSE** | **Inference Speed** |
|---|---|---|
| w/ {Top$_1$} | 0.264 | 0.082 s/iter |
| w/ {Top$_2$} | **0.262** | **0.095 s/iter** |
| w/ {Top$_4$} | 0.262 | 0.109 s/iter |
| w/ {Top$_6$} | 0.265 | 0.120 s/iter |
| w/ {Top$_8$} | 0.269 | 0.129 s/iter |