TimeSqueeze: Dynamic Patching for Efficient Long-Context Time Series Forecasting

Anonymous Author(s)

Abstract

Recent progress in time series forecasting has produced large foundation models with strong generalization across domains. However, their effectiveness is constrained by the computational cost of long-context processing. Existing designs face a core trade-off: *point-wise embeddings* preserve high-frequency information but scale quadratically with sequence length, while *patch-based embeddings* improve efficiency by downsampling, at the expense of discarding critical temporal details. We present TimeSqueeze, a hybrid forecasting architecture that resolves this trade-off through **dynamic input compression**. TimeSqueeze employs a two-stage process: (1) a lightweight Mamba encoder extracts fine-grained features at full resolution, and (2) an adaptive patching module assigns smaller patches to information-rich regions and larger patches to redundant segments. This produces a variable-resolution representation that allocates computation where it is most beneficial for forecasting. The compressed sequence is then passed to a Transformer backbone, yielding substantial reductions in token length while retaining critical temporal features. Extensive experiments demonstrate that TimeSqueeze achieves comparable performance with substantially reduced computational cost, or alternatively, enables processing much longer contexts within the same budget for significantly improved accuracy. This results in an average zero-shot MSE reduction of up to 24% compared to equivalent point embedding models, while maintaining similar computational requirements. These results position TimeSqueeze as a scalable and effective architecture for next-generation time series foundation models.

1 Introduction

Accurate time series forecasting is vital across various domains, including energy, finance, climate, and healthcare. Traditional forecasting has long relied on task-specific statistical models, but recent breakthroughs in deep learning have enabled versatile generalist models capable of cross-domain transfer. Most notably, the emergence of *time series foundation models*, large, self-supervised models trained on heterogeneous datasets, promises flexible zero-shot and few-shot generalization across forecasting tasks. Yet a persistent bottleneck remains: efficiently modeling long historical contexts. Correctly forecasting long horizons often requires processing thousands of timesteps, which poses severe computational and memory challenges.

Long-sequence architectures. Transformers [1] introduced attention-based sequence modeling to time series forecasting, achieving strong results but suffering quadratic complexity in context length. Subsequent innovations such as Informer, which introduces ProbSparse self-attention and distilling strategies, reduced this cost while retaining accuracy [2]. More recently, foundation-scale approaches such as Time-MoE exploit Mixture-of-Experts (MoE) routing to scale models to billions of parameters while maintaining tractable inference [3]. However, even with architectural efficiency, long-context modeling remains expensive.

Under review at the NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models (BERT^{2}S). Do not distribute.

Patch-based compression. An influential line of work compresses long series into patches. *PatchTST* demonstrated that treating sub-series as tokens, with channel-independent modeling, reduces sequence length while preserving local semantics, improving scalability [4]. *MOIRAI* extends this idea with multi-scale patching, any-variate attention, and masked self-supervision, enabling a single universal model to handle series from minute- to year-level resolution across diverse domains [5]. These methods highlight the promise of patching, but their reliance on fixed patch sizes inevitably discards high-frequency details in some regions while under-utilizing redundancy in others.

Insights from language modeling. A parallel challenge appears in large language models (LLMs). Tokenization introduces biases and vulnerabilities, motivating the development of tokenizer-free, byte-level models. However, byte-level inputs lead to prohibitive sequence lengths [6]. To address this, adaptive compression has emerged: the Byte Latent Transformer (BLT) dynamically merges predictable byte spans into longer latent tokens, guided by entropy-based segmentation [7]. Similarly, H-Net [8], inspired by U-Net architectures [9], compresses and reconstructs sequences hierarchically, allocating resolution adaptively across text. These approaches demonstrate that allocating finer granularity to high-information regions and aggressive compression to redundant regions can yield both efficiency and fidelity.

Our approach: TimeSqueeze. We introduce TimeSqueeze, a hybrid forecasting architecture that adapts these insights to time series. Unlike prior fixed-patch methods, TimeSqueeze employs **dynamic, content-aware patching**. First, a lightweight Mamba encoder extracts local fine-grained features at full resolution. Then, an adaptive patching module applies smaller patches to information-rich regions and larger patches to redundant ones, yielding a variable-resolution representation. This compressed sequence is processed by a Transformer backbone (e.g., Time-MoE), which dramatically reduces token length while preserving the salient temporal dynamics.

Contributions. Our contributions are as follows:

- We identify the core challenge in long-context forecasting: balancing efficiency and temporal fidelity under limited computational budgets.
- We propose TimeSqueeze, the first forecasting architecture to incorporate **dynamic, content-aware patching** for adaptive compression.
- We show that TimeSqueeze integrates seamlessly with strong backbones (e.g., Time-MoE), enabling either comparable accuracy with lower compute, or substantially longer context horizons at the same budget.
- We validate TimeSqueeze across diverse zero-shot benchmarks, achieving up to 24% MSE improvements over point embedding baseline and establishing a scalable foundation architecture for next-generation time series modeling.

2 Methodology

Problem Statement. The fundamental objective of a time-series forecasting model is to predict future values in a time series from its historical observations. Mathematically, given a sequence of T past data points, $\mathbf{X}_{1:T} = (x_1, x_2, \ldots, x_T) \in \mathbb{R}^T$, we aim to predict the next H values. This is accomplished by a model f_θ, which maps the historical context to a future forecast, denoted as $\hat{\mathbf{X}}_{T+1:T+H} = f_\theta(\mathbf{X}_{1:T}) \in \mathbb{R}^H$. By adopting the channel independence principle from [10], the model can process a multivariate input as a collection of univariate series. This versatility enables TimeSqueeze to tackle forecasting problems of any variate, making it broadly applicable to diverse real-world scenarios.

2.1 Architectural Overview

The architecture of TimeSqueeze is comprised of four key components. First, a low-complexity encoder-decoder, based on the Mamba architecture, to efficiently extract features from the input signal at full resolution. Second, patching and unpatching modules select a salient subset of features to be processed by the backbone. Third, the backbone is a decoder-only MoE transformer, responsible for computing causal and contextual representations. Finally, a multi-horizon forecasting head.

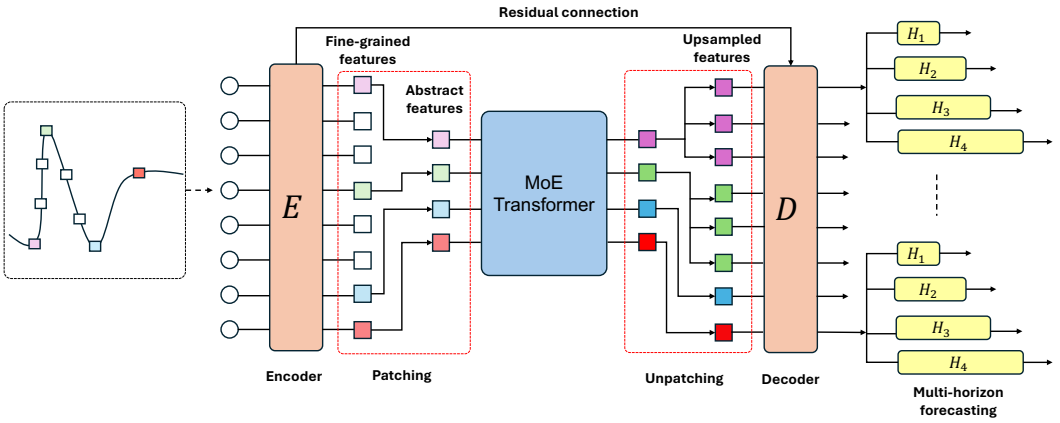

Figure 1: Architectural overview of **TimeSqueeze**. Input time series undergoes full-resolution processing by a Mamba encoder, followed by dynamic compression that reduces token count before passing to the transformer backbone, optimizing computational efficiency while preserving temporal fidelity.

Mamba Encoder-Decoder. The encoder and decoder modules are designed to process the input time series at its native resolution, preserving the fine-grained details crucial for accurate forecasting. This requirement presents a significant computational challenge, as the architecture must efficiently handle long, uncompressed sequences. Furthermore, the encoder must produce representations that are not only detailed but also structured for effective compression by the subsequent patching module.

To address these constraints, we build our encoder and decoder using Mamba layers. As a State Space Model (SSM), Mamba is particularly well-suited for this task. Recent studies have shown that SSMs excel at processing high-resolution data while scaling nearly linearly with sequence length [11]. This choice enables our model to capture intricate local patterns from long contexts without incurring the quadratic complexity associated with traditional Transformer-based approaches.

Dynamic Patching. After the encoder generates fine-grained representations, the patching module compresses the sequence of embeddings before it is passed to the transformer backbone. The objective is to allocate computational resources efficiently by creating a dynamic patching strategy that adapts to the signal's local complexity. This strategy forms larger patches to compress regions of low information density and smaller patches to preserve detail in regions with high information content.

Time-series data is uniquely suited for this approach because its local statistical properties, such as mean and variance, provide a reliable proxy for information density. We leverage this by implementing a simple yet effective metric: the deviation of a sample from a local mean. A detailed description and visualization of the patching and unpatching is presented in Appendix B.1 and B.3.

MoE Transformer. The backbone of TimeSqueeze is based on a decoder-only Transformer taken from Time-MoE-50M, incorporating several enhancements to improve performance and stability. Similar to Time-MoE, the standard feed-forward network (FFN) is replaced with an MoE layer. This layer features a sparsely activated pool of N "non-shared" experts and one designated "shared" expert, which helps consolidate common knowledge across the experts. For each input token, a routing mechanism selects the top K non-shared experts to process the signal.

Multi-horizon forecasting. To enhance forecasting flexibility and robustness, we utilize a multi-horizon forecasting head, as introduced in [3]. This component allows the model to predict outcomes across various horizons concurrently, unlike traditional models that operate at a single scale. This is achieved using multiple output projections, each a single-layer FFN dedicated to a different forecast duration. The model is trained on a composite loss function that aggregates errors from all horizons, a method designed to improve generalization.

3 Experimental Results

Baselines. Our primary objective is to demonstrate the efficiency and performance improvements over point embedding models through dynamic context compression. We use TimeMoE-50M as our primary baseline, pretrained following the training methodology from [3] with a maximum context length of 4096 and forecasting horizons $\{1, 8, 32, 64\}$, learning rate of 1e-3, AdamW optimizer, 100,000 training steps, and a reduced batch size of 256 (versus 1024), due to computational constraints,

using only 25% of the data. We refer to this model, trained with limited training data, as TimeMoE-limited and acknowledge that its performance is lower than that of the fully pretrained version in [3]. For completeness, we also compare against MOIRAI-small [5], TimesFM [12], Moment [13], and Chronos-small [14], with results taken from [3].

We specifically target a compression rate of 4 (average patch size of 4, maximum patch size of 8) for TimeSqueeze to balance computational savings with information preservation. The details of model training and the corresponding loss used are provided in Appendix B.2.

Performance. Detailed zero-shot forecasting results are provided in Table 1. For TimeMoE-limited, we consider input context lengths of (512, 1024) and forecasting horizons of (96, 192), respectively. Since TimeSqueeze uses a context compression rate of $4\times$, we scale the input context lengths to $(2048, 3904)$ respectively, to equalize the computational budget. Note that 3904 is the max input context length supported for forecasting horizon 192, before exceeding the total length of 4096. Despite significantly reduced pretraining data and computational budget, TimeSqueeze achieves comparable or superior zero-shot performance to state-of-the-art baselines across multiple datasets and forecasting horizons. Specifically, when compared with an equivalent point embedding baseline trained with the same amount of data, TimeMoE-limited, TimeSqueeze shows a reduction of up to 24% in MSE, demonstrating substantial improvements by leveraging dynamic compression to optimally allocate computational resources during inference while processing much longer contexts.

Table 1: Performance comparison of zero-shot forecasting. **Bold** for best and underscore for 2nd best.

Models	Metrics	Time-MoE$_{\text{limited}}$		TimeSqueeze (Ours)		Moirai$_{\text{small}}$		TimesFM		Moment		Chronos$_{\text{small}}$	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	96	0.373	**0.398**	**0.367**	0.399	0.401	0.402	0.414	0.404	0.688	0.557	0.466	0.409
	192	0.414	0.425	**0.402**	**0.419**	0.435	0.421	0.465	0.434	0.688	0.560	0.530	0.450
ETTh1	Avg.	0.393	0.411	**0.370**	**0.394**	0.418	0.411	0.419	0.443	0.688	0.559	0.498	0.429
	96	0.434	0.438	0.308	0.359	**0.297**	**0.336**	0.315	0.349	0.342	0.396	0.307	0.356
	192	0.623	0.527	0.405	0.425	0.368	**0.381**	0.388	0.395	**0.354**	0.402	0.376	0.401
ETTh2	Avg.	0.528	0.428	0.348	0.369	**0.332**	**0.358**	0.351	0.372	0.339	0.399	0.341	0.378
	96	0.393	0.409	**0.328**	**0.363**	0.418	0.392	0.361	0.370	0.654	0.527	0.511	0.423
	192	0.453	0.451	**0.385**	**0.403**	0.431	0.405	0.414	0.405	0.662	0.532	0.618	0.485
ETTm1	Avg.	0.423	0.430	**0.329**	**0.369**	0.424	0.398	0.387	0.387	0.658	0.529	0.564	0.454
	96	0.413	0.439	**0.201**	0.297	0.214	**0.270**	0.202	0.288	0.260	0.335	0.209	0.288
	192	0.615	0.544	0.304	0.374	0.284	0.332	0.289	**0.321**	0.289	0.350	**0.280**	0.341
ETTm2	Avg.	0.514	0.491	0.285	0.357	0.249	0.310	**0.245**	**0.295**	0.274	0.343	**0.245**	0.316
	96	0.175	0.233	**0.166**	**0.219**	0.198	0.222	....	....	0.243	0.255	0.211	0.243
	192	0.246	0.296	**0.230**	0.279	0.247	**0.265**	....	....	0.278	0.329	0.263	0.294
Weather	Avg.	0.210	0.298	**0.208**	0.295	0.275	**0.286**	....	....	0.294	0.326	0.300	0.318
	96	0.213	0.343	**0.200**	**0.333**	0.227	0.354	0.255	0.375	0.363	0.472	0.234	0.361
	192	0.266	0.397	**0.228**	**0.364**	0.269	0.396	0.313	0.423	0.387	0.489	0.276	0.400
Global Temp	Avg.	0.239	0.370	**0.214**	**0.348**	0.285	0.409	0.354	0.451	0.440	0.524	0.311	0.424
Average		0.384	0.404	**0.292**	**0.355**	0.330	0.367	0.351	0.389	0.448	0.446	0.376	0.386

4 Conclusion

We present **TimeSqueeze**, the first forecasting architecture with **dynamic, content-aware patching** that combines the temporal fidelity of point embedding models with the computational efficiency of patch-based approaches. TimeSqueeze employs a lightweight Mamba encoder for full-resolution feature extraction, followed by adaptive patching that assigns variable patch sizes based on temporal information density. Our mean deviation-based boundary detection enables data-driven compression decisions, producing variable-resolution representations that optimally allocate computational resources to where they provide the most significant forecasting benefit.

Our work opens several promising research directions. On the pretraining front, TimeSqueeze could benefit from scaling the number of parameters in the backbone and the amount of training data, similar to Time-MoE [3]. Further, the boundary detection mechanism could be enhanced through end-to-end learning in embedding spaces [8] or auxiliary model guidance [7]. TimeSqueeze's modular design enables integration with any transformer backbone, and combining it with advances in lightweight forecasting models [15] could yield greater computational savings.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] 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*, volume 35, pages 11106–11115, 2021.

[3] 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.

[4] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[5] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. 2024.

[6] Kevin Slagle. Spacebyte: Towards deleting tokenization from large language modeling. *Advances in Neural Information Processing Systems*, 37:124925–124950, 2024.

[7] Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*, 2024.

[8] Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. *arXiv preprint arXiv:2507.07955*, 2025.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[10] 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.

[11] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[12] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[13] 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. *arXiv preprint arXiv:2403.07815*, 2024.

[14] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[15] Yihang Wang, Yuying Qiu, Peng Chen, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Lightgts: A lightweight general time series forecasting model. *arXiv preprint arXiv:2506.06005*, 2025.

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

[17] 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.

A Ablation Studies

We conduct systematic ablation studies to quantify the contributions of key components in TimeSqueeze. All experiments use an input context length of 2048, forecasting horizon of 96, and target average patch size of 4, evaluated on representative datasets from our benchmark. Adetailed comparsion for ablations across all datasets is listed in Table 2.

Performance without extending context length. A key advantage of our compression approach is achieving comparable or superior performance to point embedding models while using identical context lengths but substantially reduced computational cost. By dynamically compressing redundant temporal regions, TimeSqueeze utilizes the available computational budget more efficiently, demonstrating that intelligent resource allocation can yield better performance even without leveraging longer contexts.

Dynamic vs. Fixed Patching. We compare our adaptive mean deviation-based patching against fixed uniform patching with equivalent average patch sizes. Dynamic patching consistently outperforms fixed patching by effectively allocating computational resources to information-rich segments while applying aggressive compression to redundant regions. This demonstrates that temporal heterogeneity in time series data necessitates adaptive compression strategies rather than uniform approaches.

Mamba vs. Linear Encoder. We evaluate the contribution of the Mamba architecture by replacing our Mamba encoder-decoder with linear embedding layers, similar to approaches in MOIRAI [5] and TimesFM [12]. The Mamba-based design shows substantial performance gains, validating its effectiveness for processing fine-grained temporal features and its inductive bias for sequential compression tasks.

Importance of Fine-Grained Features. To assess the value of preserving detailed temporal information, we remove the residual connection in Figure 1 and rely solely on compressed features for forecasting. This ablation results in significant performance degradation, confirming that fine-grained temporal details are crucial for accurate long-horizon forecasting and cannot be fully captured through compression alone.

Positional Encoding Analysis. We examine the impact of positional information by comparing absolute position IDs of boundary elements against relative positional encoding of compressed embeddings. Using only relative positions leads to substantial performance drops, highlighting that absolute temporal positioning is essential for maintaining temporal coherence in the reconstructed sequences.

The results consistently demonstrate that each component contributes meaningfully to TimeSqueeze's overall performance, with dynamic patching and fine-grained feature preservation showing the most significant individual impacts.

Table 2: Ablation study on zero-shot forecasting performance.

Model / Variation	ETTh1		ETTh2		ETTm1		ETTm2		Weather		Global Temp		Average	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Ours	0.352	0.385	0.303	0.349	0.310	0.353	0.219	0.311	0.172	0.220	0.198	0.331	0.259	0.324
TimeMoE-base (w/ ctx 512)	0.373	0.398	0.434	0.438	0.393	0.409	0.413	0.439	0.175	0.233	0.213	0.343	0.333 (+28.6%)	0.377 (+16.4%)
Ours (w/ fixed patching)	0.369	0.395	0.334	0.371	0.304	0.349	0.206	0.298	0.168	0.220	0.204	0.336	0.264 (+1.9%)	0.328 (+1.2%)
Ours (w/ context 512)	0.376	0.399	0.350	0.389	0.410	0.418	0.250	0.335	0.170	0.223	0.218	0.348	0.295 (+13.9%)	0.355 (+9.6%)
Linear Patching	0.354	0.386	0.355	0.385	0.310	0.350	0.266	0.344	0.158	0.210	0.204	0.336	0.275 (+6.2%)	0.335 (+3.4%)
Ours w/o Residual	0.361	0.390	0.356	0.383	0.353	0.377	0.262	0.340	0.175	0.228	0.201	0.335	0.284 (+9.7%)	0.342 (+5.6%)
Ours w/o Abs. Pos. IDs	0.343	0.386	0.386	0.406	0.560	0.456	0.243	0.332	0.179	0.232	0.220	0.351	0.321 (+23.9%)	0.360 (+11.1%)

B Appendix

B.1 Patching and Unpatching

Patching. We partition the encoder's output embeddings into non-overlapping patches using a **mean deviation criterion** applied to the input time series. Since the Mamba encoder processes inputs causally, we assume a direct correspondence between salient anchor points in the input signal and their corresponding embedding vectors.

Our adaptive patching mechanism identifies boundaries by monitoring when new samples deviate significantly from the samples in current patch. We maintain a running mean μ_{current} for each patch and establish boundaries when the absolute deviation exceeds a relative threshold:

$$b_i = \begin{cases} 1 & \text{if } |x_i - \mu_{\text{current}}| > min(\tau \cdot |\mu_{\text{current}}|, \delta) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where b_i is the boundary indicator, x_i is the input sample at timestep i, and τ is the deviation threshold parameter and impose a minimum deviation δ to ensure stability in regions with $\mu = 0$. Specifically, we set $\tau = 0.3$ and $\delta = 0.05$ to achieve our target average patch size of ~ 4.

Once boundaries are computed, we compress the embeddings in each patch by retaining only the boundary elements and discarding intermediate embeddings (Figure 1). While alternative pooling strategies (min/max/mean pooling) exist, boundary-element retention proves most effective and ensures causality for the subsequent unpatching process(Figure 1).

Unpatching. The unpatching module restores the compressed sequence to its original resolution while preserving causal consistency. Following backbone processing of the patch representatives, each updated embedding is broadcasted across all timesteps within its corresponding patch span. Since boundary elements serve as patch representatives and originate from the beginning of the patch, the reconstructed output at any timestep t depends only on information from inputs up to time t, preventing future information leakage while restoring full sequence length.

B.2 Model Training

Training. The training of a robust foundation model necessitates a large and diverse dataset. For this purpose, we employ the Time-300B dataset, a high-quality, open-access dataset composed of time series from numerous public sources across sectors like weather, transportation, and finance, and is further expanded with synthetic data. It consists of a broad range of frequencies ranging from seconds to yearly and a massive scale of over 300 billion time points make it well-suited for pre-training large-scale models.

TimeSqueeze uses TimeMoE-50M as the transformer backbone, replacing the SwiGLU point embedding layer with our Mamba encoder and adding patching, unpatching, and decoder modules. Training follows the same constrained methodology (25% data) as TimeMoE-base for fair comparison. Our training objective is a composite loss function that combines a primary forecasting loss with an auxiliary term for load balancing, following the exact methodology from [3].

The primary auto-regressive loss, $\mathcal{L}_{\text{ar}}$, is the Huber Loss [16], chosen for its robustness against outliers:

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

where δ is a hyperparameter that balances the quadratic (L_2) and linear (L_1) penalties.

To ensure balanced expert utilization and prevent routing collapse, we incorporate an auxiliary loss, $\mathcal{L}_{\text{aux}}$, as proposed in [17]:

$$\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i r_i, \tag{3}$$

where f_i is the fraction of tokens dispatched to expert i, and r_i is the average router probability assigned to it.

The final training loss, $\mathcal{L}$, averages the auto-regressive loss across P multi-resolution projections and combines it with the weighted auxiliary loss:

$$\mathcal{L} = \frac{1}{P} \sum_{j=1}^{P} \mathcal{L}_{\text{ar}} \left(\mathbf{X}_{t+1:t+p_j}, \hat{\mathbf{X}}_{t+1:t+p_j} \right) + \alpha \mathcal{L}_{\text{aux}}, \tag{4}$$

where p_j is the forecast horizon for the j-th projection and α is a scaling coefficient.

B.3 Visualization of patching

B.4 Visualization of forecasts

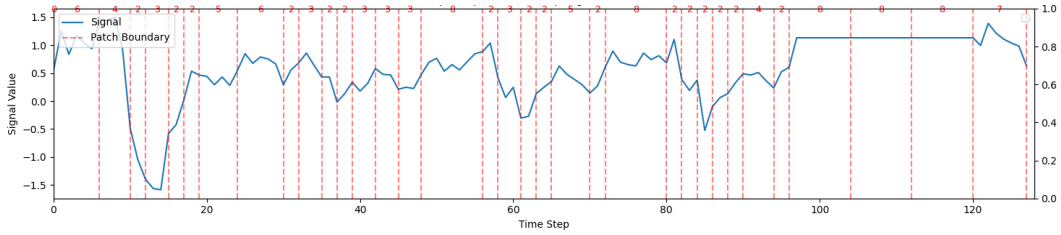

Figure 2: Patching on ETTm1 dataset using mean deviation metric. On the right, in regions of slow variation, we allocate a large patch size. But on the left, in regions of rapid variation, we choose a smaller patch size, preserving critical tempotal variations.

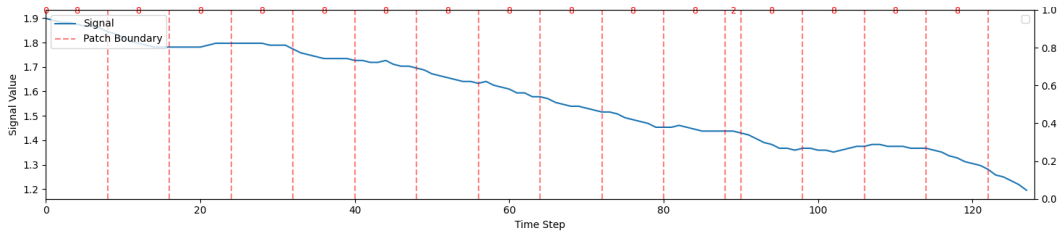

Figure 3: Patching on Weather dataset using mean deviation metric. Due to the inherently smooth temporal dynamics in weather data, the adaptive patching mechanism consistently produces patches approaching the maximum allowed size (8), demonstrating that low-frequency signals can be compressed aggressively without loosing performance.

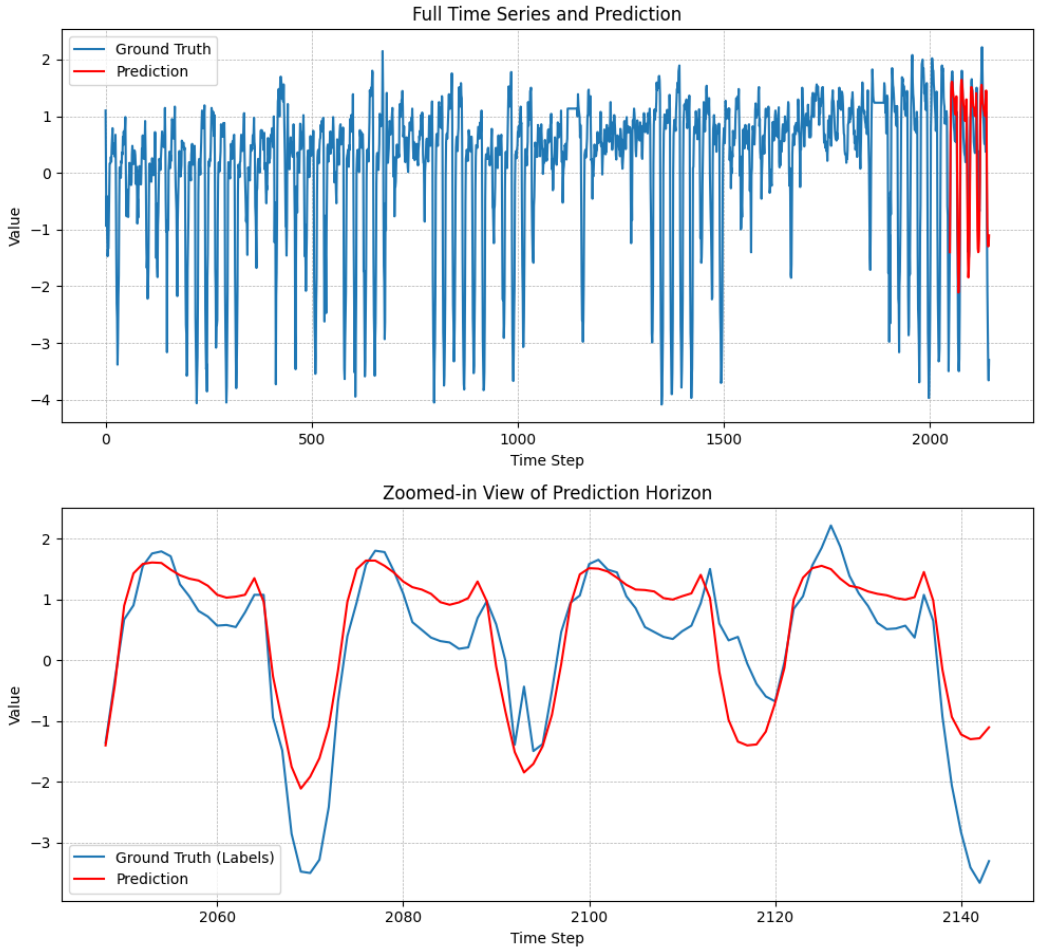

Figure 4: Forecasting on ETTh1 dataset using input context length of 2048 with average patch size 4 and a forecasting horizon of 96.

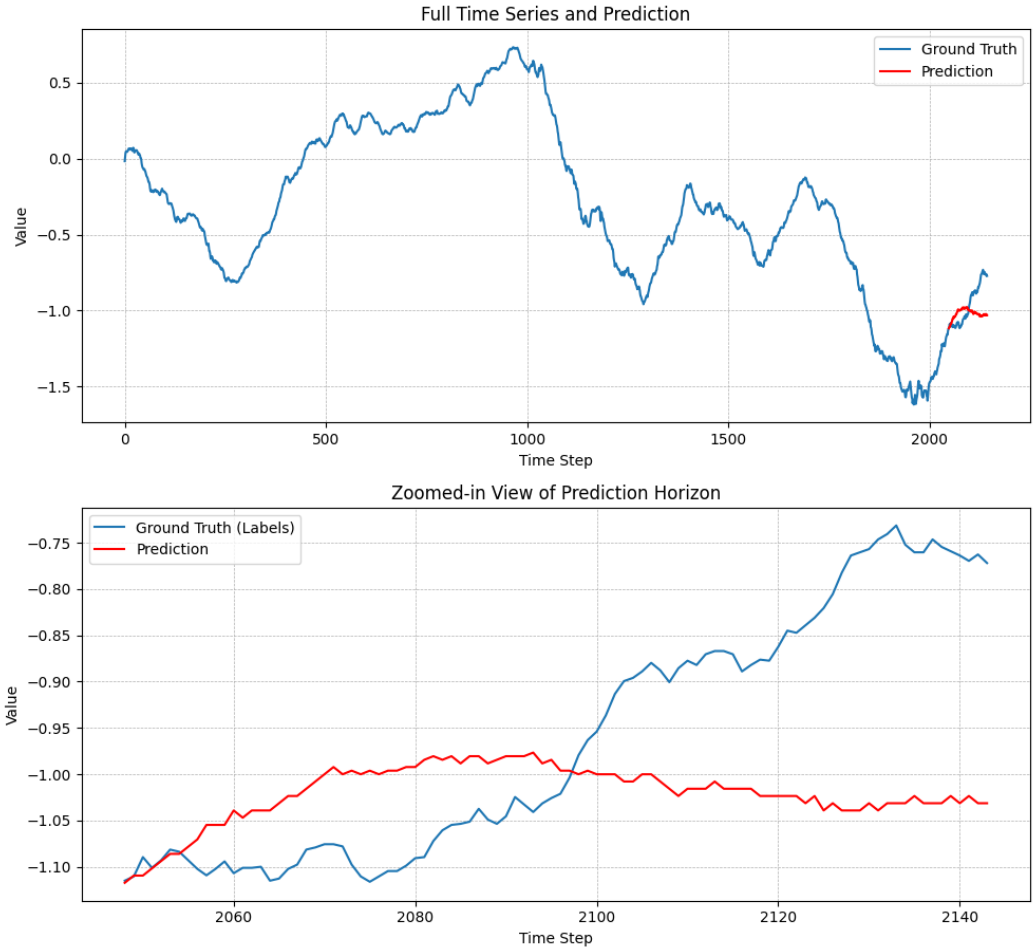

Figure 5: Forecasting on Weather dataset using input context length of 2048 with average patch size 4 and a forecasting horizon of 96.