Learning What Matters First: Sequential Adaptation of Time Series Foundation Models for Robust Financial Forecasting

Fatemeh Chitsaz [*1] **Saman Heratizadeh** [*1]

Abstract

Foundation models like TimesFM hold strong promise for financial forecasting, yet adapting them to noisy, low-resource domains such as equities remains a challenge. Through fine-tuning on S&P100 price data, we observe a trade-off between in-domain performance and generalization to unseen stocks. To better understand this behavior, we analyze layer-wise weight dynamics and find that early components contribute more consistently to adaptation, while later layers exhibit noisier and less effective updates. Guided by these findings, we propose a sequential fine-tuning strategy that updates one layer at a time, following an order determined by each layer's contribution to the overall adaptation trajectory as measured by both direction and magnitude. This targeted approach improves generalization to unseen financial data while reducing the number of trainable parameters at each stage.

1. Introduction

Foundation models have demonstrated impressive performance across domains such as language, vision, and, more recently, time series forecasting(Min et al., 2024). Among them, TimesFM (Cao et al., 2024) is a decoder-only transformer pretrained on a large corpus of real and synthetic time series, designed to support general-purpose forecasting across diverse domains.

However, their behavior when adapted to complex environments like financial markets remains poorly understood. Financial time series often exhibit irregular dynamics, distributional shifts, and limited supervision, making direct adaptation challenging.

[*]Equal contribution [1]School of Intelligent Systems, University of Tehran, Tehran, Iran. Correspondence to: Fatemeh Chitsaz <fatemeh.chitsaz@ut.ac.ir>, Saman Heratizadeh <haratizadeh@ut.ac.ir>.

Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data, Vancouver, Canada. 2025.

In such settings, it becomes crucial to move beyond black-box fine-tuning and ask: how does the model actually adapt internally? Which components are driving learning, and which may introduce instability? A deeper understanding of layer-wise adaptation is necessary to improve both efficiency and robustness

In this work, we explore the internal adaptation dynamics of TimesFM during fine-tuning on equity market data. For each model component, we measure how strongly its weight updates contribute toward the final adapted solution, capturing both the direction and scale of progress. We find that early layers tend to make more coherent and decisive contributions, while deeper layers produce smaller or more erratic changes.

Motivated by this, we introduce a sequential fine-tuning strategy that updates one layer at a time based on its contribution to the adaptation process. This method reduces unnecessary parameter drift, improves generalization to unseen stocks, and enables efficient learning in data-scarce financial settings

2. Related Work

2.1. Time Series Foundation Models

Recent time series foundation models such as Chronos(Ansari et al., 2024), Lag-Llama (Rasul et al., 2024), MOIRAI(Woo et al., 2024), and TimeGPT(Garza et al., 2023) introduce architectural innovations like memory-augmented attention and probabilistic decoding. However, they are often trained on broad time series corpora with limited exposure to real financial dynamics(Zeng et al., 2023; Fan & Shen, 2024; Cao et al., 2024). Adapting them to domains like equity markets, which are noisy, non-stationary, and data-scarce requires methods that balance generalization and stability while remaining computationally efficient. As these models grow larger, fine-tuning must avoid overfitting and excessive resource demands(Jia et al., 2024). This calls for a more transparent understanding of what changes internally during adaptation and how each layer contributes to learning under distribution shift.

2.2. Fine-Tuning and Generalization in Time Series Models

In adapting foundation models to domains like finance, characterized by limited data and distributional volatility full fine-tuning is not only prone to overfitting but also computationally expensive. This has motivated growing interest in parameter-efficient tuning (PEFT) methods such as LoRA(Hu et al., 2022), BitFit(Ben Zaken et al., 2022), and LayerNorm(Qi et al., 2022) tuning, which aim to reduce cost by updating only a small subset of weights. However, these techniques remain underexplored in time series contexts. In this work, we systematically evaluate their effectiveness and find that, while they may perform well in-domain, they often fail to generalize out-of-domain.

2.3. Layer-Wise Adaptation and Learning Dynamics

Recent work in vision and language modeling has shown that different layers in deep networks contribute unequally during adaptation: early layers tend to capture general patterns, while deeper layers are more task-specific (Han et al., 2019). Tracking weight changes across training has proven useful for understanding such dynamics (Agrawal et al., 2021), yet similar analysis remains scarce in time series foundation models. Our work extends this direction by studying layer-wise weight updates in TimesFM and using the results to design a more efficient fine-tuning strategy.

A widely used heuristic in transfer learning is Gradual Unfreezing (GU), which incrementally unfreezes layers from output to input with decaying learning rates (Howard & Ruder, 2018). GU assumes deeper layers should adapt earlier, but this assumption may not hold in time series models. Moreover, GU incurs two key drawbacks: (1) the number of trainable parameters grows cumulatively, leading to computational cost similar to full fine-tuning, and (2) this cumulative update process increases the risk of overfitting in low-resource domains. Motivated by these limitations, we analyze actual learning dynamics and propose a layer-wise fine-tuning strategy that updates one component at a time in a fixed, data-driven order, reducing both overfitting and training cost.

3. Layer-Wise Adaptation Analysis

To inform our fine-tuning strategy, we begin with an empirical analysis of how different components in TimesFM adapt during training. The model is first fully fine-tuned on financial data using pretrained weights as initialization. We apply early stopping with a patience of 5 epochs and select the checkpoint with the lowest validation loss as the target model.

For each parameter group ℓ (frequency embedding, input projection, transformer blocks, horizon projection), we define the global adaptation vector (GAV):

$$\mathrm{GAV}^{\ell} = \theta_T^{\ell} - \theta_0^{\ell}$$

where θ_0 and θ_T denote the model's parameters before and after fine-tuning, respectively. At each epoch t, we compute the update:

$$\Delta_t^{\ell} = \theta_t^{\ell} - \theta_{t-1}^{\ell}$$

To quantify the quality of each update, we compute the **Normalized Tuning Progress (NTP)**:

$$\mathrm{NTP}_t^{\ell} = \frac{\Delta_t^{\ell} \cdot \mathrm{GAV}^{\ell}}{\|\mathrm{GAV}^{\ell}\|^2}$$

where "$\cdot$" denotes the dot product. This metric captures how effectively each update advances the weights toward their target configuration.

Table 1. Mean Normalized Tuning Progress (NTP) for each model component, averaged over the initial epochs leading up to the target. Higher values indicate more consistent contribution toward the target.

MODULE	MEAN NTP
FREQUENCY EMBEDDING	**0.356**
INPUT PROJECTION	0.261
TRANSFORMER STACK	0.253
HORIZON PROJECTION	0.248

Table 1 reports the mean NTP for each module, averaged over the initial epochs leading up to the target checkpoint. The frequency embedding and input projection layers achieve the highest scores (0.356 and 0.261, respectively), indicating that they contribute most consistently during the early stages of adaptation. In contrast, the transformer stack and horizon projection yield lower values (0.253 and 0.248), suggesting a weaker role in guiding the model toward the target configuration.

To further understand the dynamics of learning over the course of training, we analyze how NTP values evolve before and after the target epoch, defined as the checkpoint with the lowest validation loss, selected via early stopping. This distinction enables us to separately examine the model's behavior as it moves toward the target versus after the optimal point has already been reached.

NTP quantifies the projection of the parameter update vector at each epoch, Δ_t^{ℓ}, onto the global adaptation vector GAV^{ℓ}. In other words, it measures how much of the change made at epoch t lies in the direction of the full trajectory from initialization to the target. This projection captures both direction and magnitude, providing insight into whether an update meaningfully contributes to learning or merely introduces noise.

In the **pre-target phase**, higher NTP values are beneficial. They indicate that updates are well-aligned with the intended adaptation trajectory, effectively steering the model toward the optimal configuration.

In the **post-target phase**, the model has already achieved its best validation state. From this point onward, continued updates can become counterproductive. High NTP values in this phase no longer reflect useful progress, but rather indicate overshooting the target and pushing the model beyond the optimal solution. This behavior corresponds with rising validation loss and reduced generalization.

Figure 1 offers a complementary view to the scalar summary in Table 1, revealing the temporal dynamics of layer-wise adaptation throughout training. In the pre-target phase , the frequency embedding and input projection layers dominate the normalized tuning progress, reflecting their central role in guiding the model toward the optimal configuration. In contrast, in the post-target phase , we observe a clear shift: contributions from deeper layers, particularly the transformer stack and horizon projection, increase markedly. This shift coincides with a rising validation loss, indicating that these layers are responsible for overfitting. The figure reinforces our key insight: early layers drive meaningful progress toward generalization, while updates beyond the target can degrade performance by fitting to noise rather than signal

4. Methodology

Building on our analysis of layer-wise weight dynamics, we propose a sequential fine-tuning strategy that updates one layer at a time, enabling efficient and robust model adaptation.

4.1. Forecasting Task

Let $x_1, x_2, \ldots, x_T$ be a univariate financial time series. The task is next-step forecasting: given a context window $X_t = (x_{t-63}, \ldots, x_t)$ of length 64, the model predicts x_{t+1}. The forecasting function is parameterized as f_θ, yielding:

$$\hat{x}_{t+1} = f_\theta(X_t)$$

4.2. TimesFM Architecture Overview

TimesFM is a decoder-only transformer model pretrained on a large corpus of real and synthetic time series. Its architecture includes four primary modules:

- **Frequency embedding layer**: encodes periodic and temporal structure.
- **Input projection layer**: transforms raw input patches into a latent space.
- **Transformer stack**: 20 layers of causal self-attention and feedforward sublayers for temporal modeling.
- **Horizon projection layer**: a final linear layer mapping internal representations to the predicted value.

4.3. Sequential Fine-Tuning Strategy

Based on our NTP analysis, we propose a progressive fine-tuning strategy that adapts model components one at a time, in a fixed order determined by their empirical contribution to adaptation. This ordering is derived from the NTP and is denoted by:

$$\mathcal{O} = [\mathcal{L}_{\text{freq\_emb}}, \mathcal{L}_{\text{input}}, \mathcal{L}_{T1}, \ldots, \mathcal{L}_{T20}, \mathcal{L}_{\text{horizon}}]$$

where $\mathcal{L}_{\text{freq\_emb}}$ and $\mathcal{L}_{\text{input}}$ denote the frequency embedding and input projection layers, $\mathcal{L}_{Ti}$ refers to the i-th transformer block, and $\mathcal{L}_{\text{horizon}}$ is the final horizon projection layer.

At each stage k, we optimize the model by updating only the parameters $\mathcal{O}_k$ while freezing all others:

$$\Theta^{(k)} = \{\mathcal{O}_1^*, \ldots, \mathcal{O}_{k-1}^*, \mathcal{O}_k, \mathcal{O}_{k+1}^0, \ldots, \mathcal{O}_n^0\}$$

Here, $\mathcal{O}_j^*$ indicates the frozen parameters from components that have already been fine-tuned, $\mathcal{O}_j^0$ represents the untouched pretrained parameters of components that are yet to be updated, and only $\mathcal{O}_k$ remains trainable at step k.

Training continues on $\mathcal{O}_k$ while the validation loss improves over the best value observed so far across all previous steps. If the validation loss fails to improve for a fixed number of epochs (patience = 5), optimization for $\mathcal{O}_k$ stops and the process moves to $\mathcal{O}_{k+1}$.

This formulation ensures that each component is fine-tuned only when it yields a measurable improvement in generalization, avoids unnecessary updates to components with lower impact, and maintains the compactness and robustness of the adapted model throughout the process.

5. Experiment Setup

We conduct experiments using daily closing prices for equities in the SP100 and SP500, retrieved from Yahoo Finance. The task is framed as one-step-ahead forecasting, where the model predicts the next value given a context window of 64 historical observations. Data is partitioned chronologically: 2000–2021 is used for training, 2022 for validation, and 2023–2024 for testing. All training and model selection are performed exclusively on SP100 stocks, with mean squared error used as the loss function during both optimization and early stopping.

To evaluate out-of-domain (OOD) generalization, we construct a disjoint set of 100 stocks randomly sampled from the

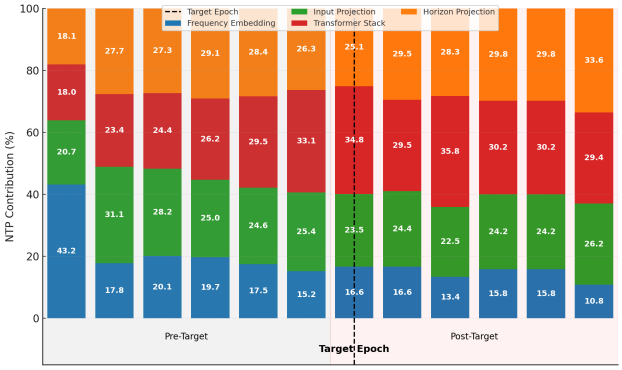

Figure 1. Normalized Tuning Progress (NTP) contribution of each model component across training epochs. Percentages are stacked and sum to 100% per epoch.

SP500 (excluding the SP100). These stocks are used solely for testing over the 2023–2024 window and remain unseen during training and validation. Model performance is reported as MSE, averaged across all stocks within each split. To ensure robustness, all experiments are repeated with five different random seeds, and final results are computed as the mean across runs.

6. Results and Discussion

We evaluate the effectiveness of our layer-wise sequential fine-tuning strategy for adapting the TimesFM foundation model to financial forecasting tasks. Experiments are conducted in both in-domain (ID) and out-of-domain (OOD) settings using daily price data from the S&P100 and S&P500, respectively. Performance is reported in terms of mean squared error, averaged across all stocks in each split and over five random seeds for robustness.

6.1. Compared Fine-Tuning Strategies

We compare our method against several established fine-tuning strategies applied to the TimesFM model. These include Low-Rank Adaptation (LoRA), which incorporates trainable low-rank matrices into the attention layers and is evaluated at rank 8; Bias Tuning (BitFit), where only the bias parameters across all layers are updated; and LayerNorm Tuning, which restricts learning to the scale and shift parameters of the normalization layers. We also consider Residual-Only Tuning, which updates solely the input and output feedforward modules while keeping the transformer stack frozen. Lastly, we include Gradual Unfreezing (GU) (Howard & Ruder, 2018), a technique where layers are sequentially unfrozen from output to input, accompanied by aggressive learning rate decay.

Table 2. Mean Squared Error for each fine-tuning strategy on in-domain and out-of-domain. Lower is better.

METHOD	ID	OOD
OURS	31.98	**20.67**
INPUT-FIRST GRADUAL UNFREEZING	32.48	20.74
FULL FINE-TUNING	32.83	20.79
LORA (RANK 8)	**31.68**	20.97
RESIDUAL-ONLY	33.04	21.02
BITFIT	32.67	21.04
LAYERNORM TUNING	32.58	21.05
GRADUAL UNFREEZING	35.27	22.08
ZERO-SHOT	39.76	24.12

6.2. Revisiting Gradual Unfreezing

We empirically evaluate Gradual Unfreezing (GU) by comparing its original top-down schedule, which unfreezes layers from output to input, with a reversed variant called **Input-First Gradual Unfreezing(IF-GU)** that starts from early components such as the frequency embedding and input projection layers.

The results show that IF-GU generalizes better than the standard version, consistent with our earlier analysis that early layers play a more critical role in adaptation for financial time series. However, both variants suffer from high computational cost due to cumulative updates and remain vulnerable to overfitting. In contrast, our proposed method fine-tunes one layer at a time based on its measured contribution, achieving better out-of-domain performance with fewer trainable parameters.

6.3. In-Domain and Out-of-Domain Evaluation

On the in-domain task, LoRA achieves the strongest performance, as shown in Table 2, reflecting strong alignment with the training distribution. However, its generalization degrades in the out-of-domain setting, suggesting signs of overfitting. Our sequential method is closely behind in-domain, yet clearly outperforms LoRA and full fine-tuning out-of-domain. Standard Gradual Unfreezing performs the worst, while Input-First Gradual Unfreezing yields a substantial improvement. These results highlight the importance of tuning order, showing that early layers contribute more to generalization than deeper ones.

7. Conclusion and Future Work

Our results highlight that analyzing weight dynamics enables more efficient and generalizable adaptation of time series foundation models through layer-wise fine-tuning. Future work will explore applying this method to other time series foundation models and incorporate more adaptive update rules and weight change metrics.

References

Agrawal, A. M., Tendle, A., Sikka, H., Singh, S., and Kayid, A. Investigating learning in deep neural networks using layer-wise weight change. In Arai, K. (ed.), *Intelligent Computing: Proceedings of the 2021 Computing Conference*, volume 284 of *Lecture Notes in Networks and Systems*, pp. 678–693. Springer International Publishing, 2021. doi: 10.1007/978-3-030-80126-7_48. URL https://doi.org/10.1007/978-3-030-80126-7_48.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Pineda Arango, S., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, Y. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024, 2024. URL https://www.stat.berkeley.edu/~mmahoney/pubs/2619_Chronos_Learning_the_Lang.pdf. Published October 2024.

Ben Zaken, E., Goldberg, Y., and Ravfogel, S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1/.

Cao, D., Jia, F., Ö. Arik, S., Pfister, T., Zheng, Y., Ye, W., and Liu, Y. TEMPO: Prompt-based generative pre-trained transformer for time-series forecasting. In *International Conference on Learning Representations*, 2024.

Fan, J. and Shen, Y. Stockmixer: A simple yet strong MLP-based architecture for stock price forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Garza, A., Challu, C., and Mergenthaler-Canseco, M. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023. URL https://arxiv.org/abs/2310.03589.

Han, Z., Zhao, J., Leung, H., Ma, K. F., and Wang, W. A review of deep learning models for time-series prediction. *IEEE Sensors Journal*, 19(20):9329–9348, 2019. doi: 10.1109/JSEN.2019.2922045.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031/.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations (ICLR 2022)*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jia, F., Wang, K., Zheng, Y., Cao, D., and Liu, Y. GPT4MTS: Prompt-based large language model for multimodal time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Min, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., and Wen, Q. Time-LLM: Time-series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, 2024.

Qi, W., Ruan, Y.-P., Zuo, Y., and Li, T. Parameter-efficient tuning on layer normalization for pre-trained language models. *arXiv preprint arXiv:2211.08682*, 2022. URL https://arxiv.org/abs/2211.08682.

Rasul, K., Ashok, A., Williams, A. R., Ghonia, H., Bhagwatkar, R., Khorasani, A., Bayazi, M. J. D., Adamopoulos, G., Riachi, R., Hassen, N., Biloš, M., Garg, S., Schneider, A., and Chapados, N. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024. arXiv preprint.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 12345–12356. PMLR, 2024. URL https://openreview.net/forum?id=9aHDI2uoi4.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time-series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10984–10992, 2023.