SCALE-INVARIANT TRAINING FOR TIME SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) are trained on large collections of time series datasets that span various morphologies and domains. This setting exposes models to series whose scales – typical magnitudes of their values – can differ substantially. Affine scaling methods such as Reversible Instance Normalization (ReVIN) scale model inputs and reverse the transform before computing the loss. We show that this inversion multiplies each series' gradient by b^p relative to loss on scaled targets, where b is the scaling denominator (e.g., standard deviation) and p is the loss degree. We call this *scale-contaminated training* (**SCALECON**), because the scale of each series consequently becomes an importance weight, causing high-scale series to dominate training. For any scale-equivariant scaler and residual loss that is homogeneous of degree p, including MSE, MAE, and Quantile Loss, we prove that computing loss on scaled targets makes every mini-batch gradient and, consequently, the full optimization trajectory invariant to arbitrary independent rescaling of the training series, yielding *scale-invariant training* (**SCALEIN**). Notably, existing TSFMs use both objectives, with neither consistent reporting nor a common convention on how to compute training loss. We isolate the convergence disparity induced by **SCALECON** and its correction under **SCALEIN** in controlled studies on synthetic and real data. In pretraining across four TSFM architectures, **SCALEIN** lowers MASE in all 24 architecture-benchmark comparisons, with average reductions across TSFMs of 18.8% on GIFT-Eval and 21.9% on the M-competitions. The gains extend to supervised neural forecasting, where it lowers MASE in 16 of 20 matched settings. Most existing time series forecasting pipelines can adopt **SCALEIN** with a one-line code change.

1 INTRODUCTION

Scale differences across time series pose a challenge for any forecasting model. Individual series can exhibit substantial scale variation across sliding windows due to nonstationarity, level shifts, or trends, and, in datasets containing multiple series, across series themselves. This challenge is particularly relevant in Time Series Foundation Models (TSFMs) (Garza et al., 2023; Das et al., 2024; Ansari et al., 2025; Woo et al., 2024; Goswami et al., 2024; Cohen et al., 2025; Ness et al., 2023), where pretraining combines series from diverse domains, so scale differences across series within a training batch arise by construction and can span many orders of magnitude.

Prior empirical studies highlight related scale sensitivities. Wolff et al. (2025) finds that models underperform on low-magnitude series, Ahmed et al. (2026) find that `ReVIN`'s effectiveness depends on where the loss is computed, and Berthelier et al. (2026) finds empirically that computing loss on scaled targets improves model performance, Olivares et al. (2025) reveals accuracy struggles of TSFMs and traces them to scale heterogeneity. A large body of work has proposed scaling methods for time series focused on *what statistics to use* and *how to compute them* (`ReVIN` (Kim et al., 2022), DAIN (Passalis et al., 2020), GAS-Norm (Urettini et al., 2024), Non-stationary Transformers (Liu et al., 2022)). An orthogonal design choice is *where the scaling boundary sits relative to the loss*, that is, whether the loss is computed after inverse scaling of model predictions or after scaling the target. While recent work has empirically compared these loss spaces, we formalize the consequences in training and show that this boundary determines whether training is **scale-invariant** or **scale-contaminated**.

When the loss is computed after rescaling predictions back to the original scale, as prescribed by `ReVIN` (Kim et al., 2022), the full series scale is reintroduced into the loss function. We refer to

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

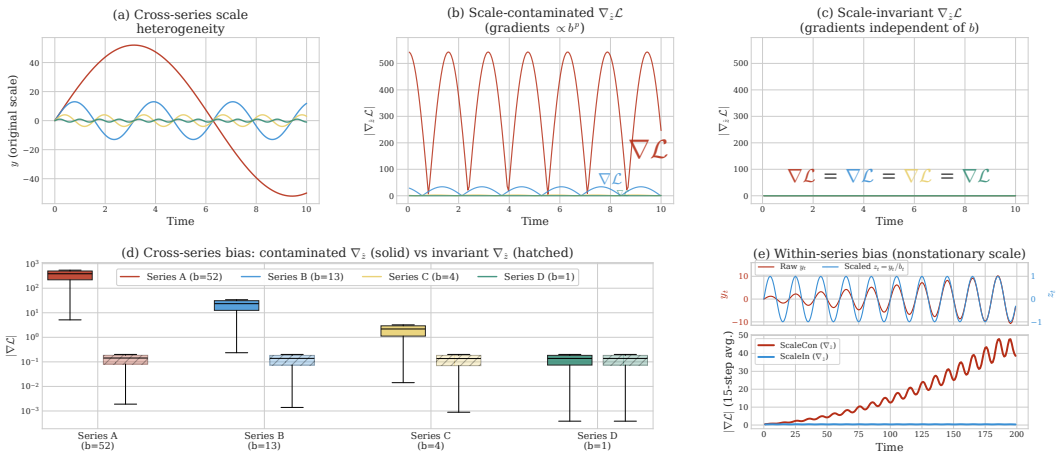


Figure 1: **ScaleIn** removes scale-driven gradient bias (Theorem 3.2). (a) Synthetic series scales span $b \in \{1, 4, 13, 52\}$. (b) Under **ScaleCon**, MSE gradients with respect to $\hat{z}$ scale as b^2, giving b=52 approximately 2,700× the gradient magnitude of b=1 for equal scaled residuals. (c) **ScaleIn** leaves gradients unchanged by rescaling, on the same y-axis. (d) Gradient distributions under **ScaleCon** (solid) and **ScaleIn** (hatched). (e) Growing scale induces gradient drift under **ScaleCon** but not **ScaleIn**. Lower-row curves use a 15-step moving average of $|\nabla\mathcal{L}|$.

this phenomenon as scale-contaminated training (**ScaleCon**). Under a degree-p loss, each series' gradient is weighted by b^p, creating an implicit and potentially unintended importance weighting as shown in Fig. 1 and discussed in Section 3. Nonstationarity within a single series causes this weighting to vary across forecast creation dates even in traditional single-series training. In TSFMs, cross-dataset training compounds the effect across series within a batch. In this work, we show that computing the loss directly on scaled targets yields Scale-Invariant Training (**ScaleIn**) and improves forecast accuracy in TSFM pretraining and supervised forecasting.

Yet despite its impact on TSFMs, loss space is rarely discussed or documented in published work. Tables 3 and 4 show inconsistent reporting across existing TSFMs and popular forecasting codebases. Of the 24 TSFMs surveyed (Table 3), only 9 explicitly document their loss space. For another 7, we infer the likely loss space by tracing how the paper describes normalization, model outputs, and loss computation, although the paper does not clearly establish it. Furthermore, inspecting the released code allows us to classify 5 more models, while 3 remain unresolved from either source. Among the 21 models assigned a loss space from these sources, 14 use **ScaleCon** and 7 use **ScaleIn**, showing that no prevailing convention exists. Moreover, among popular forecasting codebases (Table 4), only 2 support scale-invariant loss in some capacity but none make loss space user-configurable or document it explicitly. The practical change for training pipelines is one line of code that computes the loss on scaled targets instead of inverse scaling predictions, using existing context statistics (see Appendix C). We advocate for explicit reporting of loss space as a training design choice that affects convergence and downstream accuracy.

Our main contributions are as follows:

(i) We formally characterize the implicit b^p weighting induced by inverse scaling and prove that Scale-Invariant Training (**ScaleIn**) leaves every mini-batch gradient and the full optimization trajectory unchanged under arbitrary positive series-wise rescaling (Theorem 3.2).

(ii) We document inconsistent reporting of loss space across TSFM papers and forecasting codebases, establishing the need to report it explicitly as a design choice. The practical change is lightweight and computes the loss on scaled targets using existing context statistics.

(iii) We validate scale-invariance theory in four settings using controlled synthetic data, a real-world scale-swap intervention, TSFM pretraining, and supervised training. These experiments provide empirical evidence that **ScaleIn** removes scale-driven convergence disparities and improves forecast accuracy in both foundation model pretraining and supervised forecasting settings.

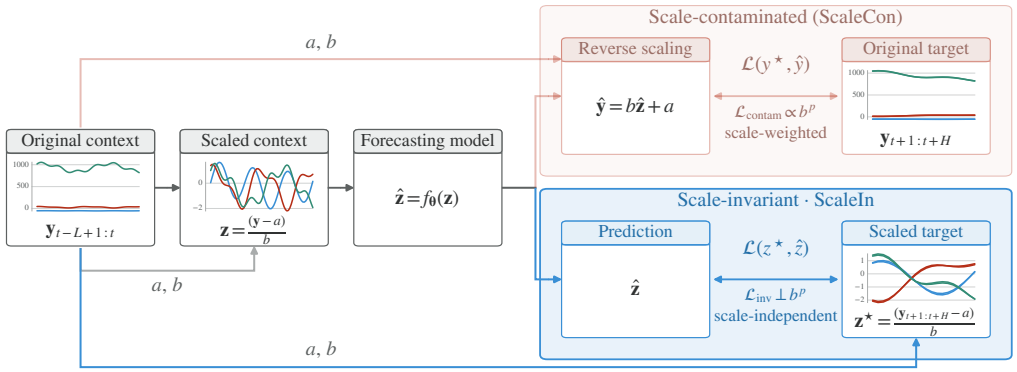


Figure 2: Loss placement determines whether inverse scaling introduces b^p gradient weighting. Both pipelines scale inputs using context statistics a, b. **Top: SCALECON** reverses scaling before computing the loss, introducing the b^p weight. **Bottom: SCALEIN** computes loss on scaled targets and reverses scaling only for inference, yielding scale-invariant training (Theorem 3.2).

2 RELATED WORK

Training time series models on heterogeneous data requires accounting for series whose scales differ by orders of magnitude. The most popular approach, and the one most frequently cited in TSFM papers, is Reversible Instance Normalization (`ReVIN`) (Kim et al., 2022), which scales each context window inside the model's forward pass and restores predictions to the original scale before computing the loss. However, these TSFMs differ in where they compute the loss (see the summary in Table 3). One group (Rasul et al., 2023; Goswami et al., 2024; Jin et al., 2024; Woo et al., 2024; Auer et al., 2025; Das et al., 2024; Liu et al., 2026; Graf et al., 2026) reverses the scaling before computing the loss, as prescribed by `ReVIN`. Another group (Ansari et al., 2025; Cohen et al., 2025; Khwaja et al., 2026; Xia et al., 2026) scales the targets and computes a scale-invariant loss. DAIN (Passalis et al., 2020) learns adaptive input shifts, scales, and feature gates. Other approaches including Non-stationary Transformers (Liu et al., 2022), GAS-Norm (Urettini et al., 2024), and Vilhes et al. (2026), differ in how they estimate scaling statistics but, like `ReVIN`, restore predictions before computing the loss.

Recent empirical studies examine some of these design choices. Ahmed et al. (2026) compare scaling schemes across TSFM architectures, including a comparison of loss computed on scaled targets with clipping and loss computed after returning predictions to the original scale. Berthelier et al. (2026) ablate the components of `ReVIN` on two supervised forecasters across four benchmark datasets. They observe that computing loss on scaled targets improves accuracy and hypothesize that the benefit comes from equal weighting of low- and high-scale series. We formally characterize this mechanism for homogeneous residual losses, deriving the scale-dependent gradient weights and establishing conditions for invariance of the full optimization trajectory. Moreover, our controlled experiments isolate the effect of loss space. On the empirical side, we evaluate TSFM pretraining and supervised forecasting on the M1, M3, M4, Tourism, Favorita, and GIFT-Eval datasets (Makridakis et al., 1982; Makridakis and Hibon, 2000; Makridakis et al., 2020; Athanasopoulos et al., 2011; Cook et al., 2021; Aksu et al., 2024). Our synthetic studies isolate the b^p mechanism, while pretraining and supervised experiments test its practical consequences (Section 4). These results establish why the loss space should be deliberately chosen and reported explicitly in forecasting papers as a design choice.

3 FORECASTING TASK AND SCALE-INVARIANT TRAINING LOSSES

We consider supervised univariate forecasting with real-valued targets. At forecast creation date t, the model receives a context $\mathbf{y}_{t-L+1:t} \in \mathbb{R}^L$, where L is the supported context length, and predicts the next H values $\mathbf{y}_{t+1:t+H} \in \mathbb{R}^H$. Before the model sees the context, a scaling (or normalization) method computes an offset a and a positive denominator b from observations available by time t and transforms each value as:

$$z_t = \frac{y_t - a}{b}. \tag{1}$$

After that, the model maps the scaled context $\mathbf{z}_{t-L+1:t}$ to a scaled prediction $\hat{\mathbf{z}}_{t+1:t+H}$,

$$\hat{\mathbf{z}}_{t+1:t+H} = f_{\boldsymbol{\theta}}\left(\mathbf{z}_{t-L+1:t}\right). \tag{2}$$

In this work, we study residual losses of the form $\mathcal{L}(y, \hat{y}) = \phi(y - \hat{y})$ that satisfy $\phi(br) = b^p \phi(r)$ for $b > 0$. The three cases used throughout this manuscript and popular in related work are:

$$\begin{aligned} \mathrm{MAE}(y, \hat{y}) &= |y - \hat{y}|, & p = 1, \\ \mathrm{MSE}(y, \hat{y}) &= (y - \hat{y})^2, & p = 2, \\ \mathrm{QL}_q(y, \hat{y}) &= q(y - \hat{y})_+ + (1 - q)(\hat{y} - y)_+, & p = 1. \end{aligned} \tag{3}$$

The central choice is whether in training the loss is computed before or after returning the prediction to the original scale. The default pipeline with `ReVIN` (Kim et al., 2022) first reverses the transform,

$$\hat{y}_{t+h} = b\hat{z}_{t+h} + a, \tag{4}$$

and then compares $\hat{y}_{t+h}$ with y_{t+h}. We call this objective scale-contaminated (**ScaleCon**),

$$\mathcal{L}_{\mathrm{contam}} = \frac{1}{H} \sum_{h=1}^{H} \phi\left(y_{t+h} - (b\hat{z}_{t+h} + a)\right). \tag{5}$$

The alternative is to scale the observed target with the same a and b,

$$z_{t+h} = \frac{y_{t+h} - a}{b}, \qquad h = 1, \ldots, H. \tag{6}$$

Note that the future observations do not contribute to a or b. As we show below, computing the loss before reversing the transform gives scale-invariant loss (**ScaleIn**),

$$\mathcal{L}_{\mathrm{inv}} = \frac{1}{H} \sum_{h=1}^{H} \phi\left(z_{t+h} - \hat{z}_{t+h}\right). \tag{7}$$

Lemma 3.1 (Inverse scaling creates importance weights)**.** *For every forecasting example, the scale-contaminated and scale-invariant objectives satisfy*

$$\mathcal{L}_{\mathrm{contam}} = b^p \mathcal{L}_{\mathrm{inv}}, \tag{8}$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{contam}} = b^p \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{inv}}. \tag{9}$$

The gradient relation also holds for subgradients at nondifferentiable residuals.

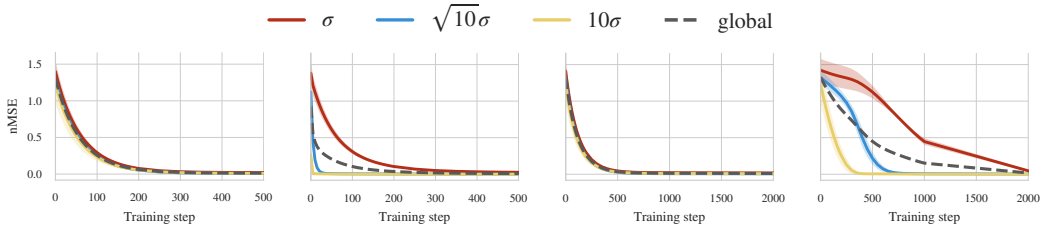
Proof in Appendix B. For a mini-batch $\mathcal{B}$, the resulting update is therefore

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{contam},\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} b_i^p \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{inv},i}. \tag{10}$$

The denominator b_i is an importance weight inside every gradient descent update. For instance, the weight is b_i for MAE and quantile loss and b_i^2 for MSE. Because b_i varies across series and forecast creation dates, scale-contaminated training changes the relative importance of forecasting examples even when their scaled forecasting problems are identical. Figure 1 visualizes the MSE gradient gap, and the left panel of Figure 9 confirms the b^2 pattern at initialization. In Figure 2 we visualize the difference between **ScaleCon** and **ScaleIn** in a typical forecasting pipeline.

Theorem 3.2 (Scale-Invariant Training)**.** *Assume that the scaling method is scale equivariant, meaning that multiplying a series by any $c > 0$ multiplies its offset a and denominator b by the same c. Under scale-invariant loss, independently multiplying each training series by any positive constant leaves the loss and parameter gradient of every forecasting example unchanged for all $\boldsymbol{\theta}$. Consequently, it leaves every mini-batch gradient and the full parameter trajectory unchanged when the initialization, mini-batch sequence, optimizer state, and optimizer randomness are fixed.*

Proof in Appendix B. The scale-equivariance assumption holds for commonly used scaling methods such as standardization ($\frac{x-\mu}{\sigma}$), min-max scaling ($\frac{x-\min}{\max-\min}$), and robust scaling ($\frac{x-m}{\mathrm{IQR}}$), all with positive denominators and without fixed numerical stabilizers. Thus, under **ScaleIn**, series' scale cannot change which forecasting examples dominate training. Fixed additive ϵ terms or scale floors can break exact equivariance, so practical invariance may be approximate. Importantly, they do not remove the b^p weighting in Lemma 3.1, which holds for the actual denominator used in the affine transform.



(a) Scale-invariant (b) Scale-contaminated (c) Equal-scale control (d) Unit gradient

Figure 3: Per-source validation nMSE over training steps. We report the mean ± 1 standard deviation over five seeds. **(a)** The scale-invariant loss yields parallel convergence. **(b)** The scale-contaminated loss produces scale-ordered convergence. **(c)** The disparity disappears when all sources share the same scale. **(d)** The disparity persists after matching the global gradient norm. The dashed grey curve is global nMSE, aggregated rather than disaggregated by source.

Remark 3.3 (Practical intuition). Consider two series with identical scaled contexts and targets, but denominators $b = 1$ and $b = 10$ – for instance, assume standardization with b corresponding to the standard deviation. If the model makes the same scaled error on both, for the second series, SCALECON yields a gradient that is 10 times larger under MAE or pinball loss and 100 times larger under MSE. The optimizer treats the second series as more important solely because of its raw scale. Under SCALEIN, both series produce the same gradient. Note, however, that even under scale invariance, different forecasting problems can still produce different gradients and achieve nonuniform convergence due to inherent data difficulty.

Some TSFMs (e.g., `Chronos2`) additionally apply an arcsinh transform after standardization. We note that scale contamination persists when this transform is reversed, but the nonlinearity changes the residual geometry, so the two objectives are not related solely by b^p. Appendix B.2 gives the corresponding identity and explains why SCALEIN remains scale-invariant under equivariance.

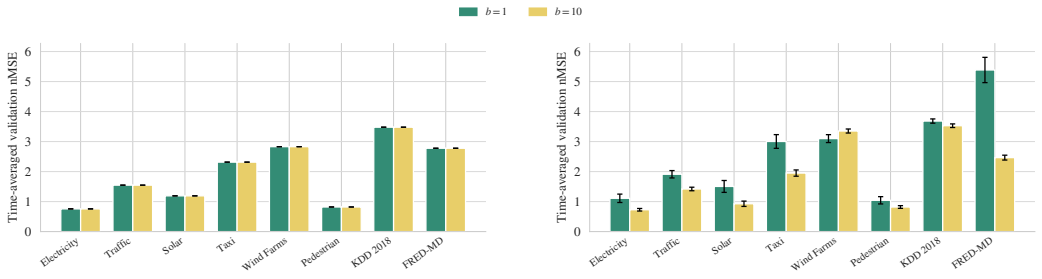
Although stated for univariate point forecasting, our result extends componentwise to multivariate objectives that sum homogeneous residual losses across channels (Liu et al., 2024; Ekambaram et al., 2024). It also covers quantile forecasting because pinball loss is homogeneous of degree one at every quantile level (Taylor, 2000; Ansari et al., 2025), as detailed in Appendix B.1. We include an algorithm implementation of SCALEIN in Section C. For clarity, we note that predictions $\hat{z}$ return to the original scale for inference and evaluation.

4 EXPERIMENTS

We evaluate SCALEIN in 4 complementary settings, from controlled isolation of the mechanism (Sections 4.1 and 4.2) to TSFM pretraining and supervised forecasting (Sections 4.3 and 4.4).

4.1 CONTROLLED SYNTHETIC DATA EXPERIMENT

In this experiment, we expose the disparate convergence per-series resulting from b^p gradient bias in standard `ReVIN` and demonstrate how SCALEIN fixes this. To isolate the effect in a simple controlled setting, we generate a synthetic data set consisting of three sinusoidal sources with scaling denominators in an approximate ratio $1 : \sqrt{10} : 10$. As shown in Lemma 3.1, `ReVIN`'s denormalization under MSE loss introduces b^2 scaling of loss values and gradient multipliers, yielding the ratio $1 : 10 : 100$ across the three sources. This scale-dependent gradient should cause the optimizer to disproportionately reduce error on high-scale series and lead to disparate convergence rates across sources. To test this, we train a 2-layer `PatchTST`-style Transformer with `ReVIN` for 30K SGD steps, fixing the random seed, parameter initialization, and the examples seen during optimization. With all model and training aspects held fixed, the only experimental parameter that varies is whether the model learns on a *scale-contaminated* or a *scale-invariant* loss. In the synthetic and scale-swap experiments (Section 4.2), we report normalized mean squared error (nMSE), computed between predictions and targets normalized using each window's context mean and standard deviation, then averaged over forecast timesteps and validation windows. Both loss conditions use this metric so errors are comparable across scales. The results show that under SCALECON, the high-scale source



(a) Scale-invariant loss (SCALEIN)

(b) Scale-contaminated loss (SCALECON)

Figure 4: Inflating scale leaves trajectory-level error nearly unchanged under SCALEIN **(a)**, but lowers it on average under SCALECON **(b)**. Bars show $A_{b,c}^{(k)}$ averaged over five seeds and 15 complementary assignment pairs. Error bars show 95% Student-t intervals across assignment pairs.

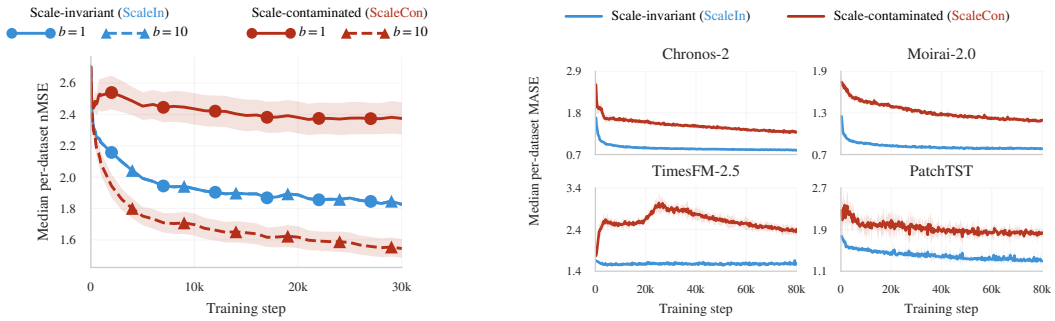
converges $50\times$ faster than the low-scale source (Figure 3b). In comparison, with SCALEIN in Figure 3a, all three scale sources converge in parallel at a similar pace. This confirms that the b^p gradient bias characterized by Lemma 3.1 does indeed produce disparate per-series convergence, as hypothesized. Appendix E.1 provides the full experimental settings.

In addition, we controlled the experiments to better attribute the gap to b^p and eliminate confounders. First, to isolate the effect of series scale, we give all sources unit amplitude and train again with the scale-contaminated loss (Figure 3c). The gap vanishes compared with Figure 3a, confirming that it stems from scale heterogeneity rather than intrinsic source difficulty. Second, to eliminate gradient-magnitude confounders, we rescale the global gradient to unit norm (Figure 3d). The convergence gap persists, matching the dynamics of SCALECON training in Figure 3b. Because b^2 scaling is applied item-wise within a batch, a single global scalar cannot fix the between-source ratio: it rescales all per-item gradients uniformly and preserves their relative magnitudes. These findings also hold under Adam (Kingma and Ba, 2015), where the gap narrows from approximately $50\times$ to $10\times$, consistent with adapting gradient magnitude but not its per-source composition (Appendix E.2).

4.2 SCALE-SWAP INTERVENTION EXPERIMENT

In this experiment, we confirm the effect of loss-space gradient bias on disparate per-series convergence using real-world data. We selectively rescale series $10\times$ as a diagnostic: under SCALEIN, Theorem 3.2 predicts rescaling leaves convergence unchanged, while under SCALECON, the b^p bias should cause the optimizer to favor the inflated series. We selected $K = 8$ datasets from different domains in GluonTS (Alexandrov et al., 2020) (Electricity, Traffic, Solar, Taxi, Wind Farms, Pedestrian, KDD Cup 2018, FRED-MD) and applied global standard scaling ($\frac{x-\mu}{\sigma}$). Then, for each dataset $D^{(k)}$ we create a matched pair $(D_{b=1}^{(k)}, D_{b=10}^{(k)})$: the original series at scale b=1 and a copy multiplied by b=10. Because both copies of a dataset differ only in scale, any convergence difference between them is attributable to the b^p-scaled gradient bias rather than to intrinsic difficulty.

We construct 15 configurations $D_c = \{D_b^{(k)}\}_{k=1}^{8}$, each assigning four datasets to b=1 and four to b=10, plus a complementary run that reverses every assignment. Across the 30 resulting runs, every dataset appears 15 times at each scale, and we average over configurations to minimize the impact of batch composition. We use the `PatchTST`-style Transformer from Section 4.1 with four layers instead of two, trained under scale-invariant and scale-contaminated losses with SGD on $S = 5$ seeds per condition. Because scale-contaminated MSE multiplies the gradient of a b=10 window by 100 while leaving a b=1 window unchanged, the two conditions cannot share a learning rate without confounding the diagnostic with raw update magnitude. We therefore divide the contaminated learning rate by the nominal average multiplier of a balanced batch, $(1 + 100)/2 = 50.5$, giving 10^{-4} for the scale-invariant condition and 1.9802×10^{-6} for the scale-contaminated one. This approximately compensates for the overall gradient amplification while preserving the relative $100:1$ weighting *within* each contaminated batch, which is the quantity our diagnostic tests. The comparison of interest is in either case within a condition, between its own b=1 and b=10 copies. Figure 5a shows the paired convergence trajectories, and the weighted-average analysis below quantifies it. Appendix F.1 provides the full experimental settings.



(a) Scale-swap corpus, K=8 datasets

(b) Full TSFM pretraining, four architectures

Figure 5: **ScaleIn** removes scale-driven convergence disparities and improves TSFM convergence. **(a)** Median per-dataset nMSE over 8 scale-swap datasets, aggregated across 15 assignment pairs and five seeds. The b=1 and b=10 curves overlap under **ScaleIn** and separate under **ScaleCon**. **(b)** Median per-dataset MASE for four architectures over 80,000 updates. Plotted are averages across 4 seeds, with 95% Student-t intervals.

To validate our hypothesis from Section 4.1 on real-world data, we compare each dataset's average validation nMSE over training when assigned b=1 versus b=10. We compute this average as the area under the validation curve divided by training duration, then average over 5 seeds and 15 complementary assignment pairs. Figure 4 shows that inflating scale leaves this error essentially unchanged under **ScaleIn**, with a mean difference across datasets below 0.002%. Under **ScaleCon**, the mean error falls from 2.6 to 1.9, a 26.8% reduction, and a one-sided paired t-test across the dataset-level differences is significant at $\alpha = 0.05$. This shows that the b^p gradient bias of Lemma 3.1 extends to real-world data. Full metric definitions, test statistics, and dataset-level trajectories are in Appendix F.

4.3 Time Series Foundation Model Pretraining

In this experiment, we empirically analyze the difference between using **ScaleIn** and **ScaleCon** in TSFM pretraining. We employ 5 forecasting benchmarks drawn from well-known forecasting competitions: `M1` (Makridakis et al., 1982), `M3` (Makridakis and Hibon, 2000), `M4` (Makridakis et al., 2020), `Favorita` (Cook et al., 2021), and `Tourism` (Athanasopoulos et al., 2011). We additionally evaluate on GIFT-Eval, a popular benchmark in the TSFM literature (Aksu et al., 2024). We adopt the data handling and pre-processing practices established in prior work leveraging heterogeneous-dataset pretraining (Alexandrov et al., 2020; Olivares et al., 2025; Potosnak et al., 2026). Benchmark training prefixes exclude reserved validation and test regions (Appendix G).

We pretrain four architectures for 80,000 updates at batch size 512, 4 seeds each, on GiftEval-Pretrain (Aksu et al., 2024) with the *training regions* of M1, M3, M4, Tourism, and Favorita included. Two are recent foundation models in their small configurations, `Chronos2` (27.9M parameters, Ansari et al. (2025)) and `Moirai-2.0` (5.0M parameters, Liu et al. (2025a)). The third is `TimesFM2.5` (49.6M parameters, Das et al. (2024)), and the fourth is a supervised `PatchTST` (1.5M parameters, Nie et al. (2023)). These architectures span a representative range of recent TSFMs in terms of scale (1.5M–49.6M parameters) and design (decoder-only and encoder-only). `PatchTST` was not originally proposed as a foundation model, but recent work has shown its effectiveness in multi-dataset pretraining (Wen et al., 2026). Its inclusion highlights that this work is applicable to any general forecaster, particularly those trained across heterogeneous datasets. Each model keeps the temporal scaling method of its own published implementation. Within each model, **ScaleIn** and **ScaleCon** runs see identical data in the same order and share initialization, optimization schedule, and learning rate, with no per-condition adjustment. AdamW normalizes each coordinate by the running root-mean-square of its own gradient, so a uniform rescaling of the objective leaves its updates unchanged (note, it does not wash out within batch item-wise b^p scaling).

We score the final checkpoint on one official test horizon per series for M1, M3, M4, and Tourism, 16 day holdout for Favorita, and GIFT-Eval's non-overlapping short-term windows across 55 configurations. Models receive up to 512 historical observations, with shorter contexts padded and masked. Within each benchmark, we average scores between frequency groups (dataset configurations for

Table 1: Forecasting performance across benchmark datasets. The results were averaged across 4 seeds. **ScaleIn** uses scale-invariant loss and **ScaleCon** uses scale-contaminated loss. These are the only differences between the two runs of each architecture. $\Delta\%$ reports **ScaleIn** relative to **ScaleCon**. Green denotes an improvement and red a regression. The better loss is bold within each architecture. References are fitted on the series they forecast and appear behind the double rule.

	Chronos2			Moirai-2.0			TimesFM2.5			PatchTST			Reference
Benchmark	**Sc.In**	**Sc.Con**	$\Delta\%$	**Sc.In**	**Sc.Con**	$\Delta\%$	**Sc.In**	**Sc.Con**	$\Delta\%$	**Sc.In**	**Sc.Con**	$\Delta\%$	S.Naive
MASE													
M1	**1.879**	2.388	-21.3	**2.013**	2.123	-5.2	**2.731**	2.863	-4.6	**2.091**	3.859	-45.8	2.117
M3	**1.608**	1.963	-18.1	**1.621**	1.800	-9.9	**2.233**	2.573	-13.2	**2.041**	4.840	-57.8	1.764
M4	**1.845**	2.294	-19.6	**1.879**	1.986	-5.4	**2.661**	3.208	-17.0	**2.235**	4.555	-50.9	2.057
Tourism	**2.350**	2.816	-16.5	**2.316**	2.564	-9.7	**3.820**	4.108	-7.0	**2.509**	3.123	-19.7	2.412
Favorita	**0.816**	0.862	-5.4	**0.825**	0.864	-4.5	**1.050**	1.360	-22.8	**0.889**	0.968	-8.2	1.079
GIFT-Eval	**1.495**	1.782	-16.1	**1.547**	1.676	-7.7	**2.162**	2.687	-19.5	**1.902**	2.813	-32.4	1.824
WQL													
M1	0.133	**0.119**	+11.7	0.174	**0.164**	+6.6	**0.208**	0.241	-13.7	**0.227**	0.284	-20.2	0.150
M3	**0.084**	0.088	-4.7	**0.109**	0.114	-4.4	**0.158**	0.176	-10.6	**0.153**	0.253	-39.3	0.110
M4	**0.084**	0.089	-5.7	0.110	**0.107**	+2.9	**0.161**	0.183	-11.7	**0.147**	0.229	-36.0	0.115
Tourism	**0.086**	0.113	-24.1	**0.116**	0.132	-12.0	**0.227**	0.266	-14.8	**0.149**	0.200	-25.5	0.123
Favorita	**0.297**	0.326	-9.1	**0.387**	0.422	-8.3	**0.638**	0.848	-24.8	**0.540**	0.595	-9.3	0.614
GIFT-Eval	**0.285**	0.320	-10.9	**0.371**	0.397	-6.5	**0.569**	0.681	-16.4	**0.518**	0.597	-13.3	0.705

GIFT-Eval), weighted by the number of forecasts evaluated, then average across 4 training seeds (Appendix G). We include Seasonal Naïve (`S.Naive`) as a reference, which repeats the last observed season using the benchmark's seasonal period and repeats the last observation when fewer than two seasons are available (Hyndman et al., 2026).

We report two scale-independent metrics. The Mean Absolute Scaled Error (MASE) (Hyndman and Koehler, 2006) normalizes the forecast error by the in-sample seasonal naive error,

$$\text{MASE} = \frac{1}{H}\sum_{h=1}^{H}\frac{|y_{t+h} - \hat{y}_{t+h}|}{\frac{1}{T-m}\sum_{j=m+1}^{T}|y_j - y_{j-m}|}, \tag{11}$$

where m is the seasonal period. The Weighted Quantile Loss (WQL) averages twice the quantile loss over the quantile levels $\mathcal{Q}$ supplied by each model,

$$\text{WQL} = \frac{2}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}\frac{\sum_{h=1}^{H}\text{QL}_q(y_{t+h}, \hat{y}^{(q)}_{t+h})}{\sum_{h=1}^{H}|y_{t+h}|}, \tag{12}$$

with QL_q defined in Section 3. For models that do not supply quantile forecasts, we treat the point prediction as the median and compute WQL with $\mathcal{Q} = \{0.5\}$.

Table 1 reports MASE and WQL across all benchmarks for all architectures at the final checkpoint. **ScaleIn** outperforms **ScaleCon** in terms of lower MASE across all 24 architecture-benchmark comparisons, with error reductions (averaged across architectures) between 10.2% and 24.1%. On WQL it wins 21 of 24 comparisons, with average performance gains between 3.3% and 19.1%. The improvement is consistent regardless of the normalization method: `TimesFM2.5` computes statistics from an expanding causal window, `Chronos2`, `Moirai-2.0`, and `PatchTST` from the context window, with `Chronos2` additionally applying an arcsinh transformation.

In an ablation study with `MOMENT` (Goswami et al., 2024), we also find that the benefit extends beyond forecasting objectives: **ScaleIn** substantially lowers the reconstruction MASE of `MOMENT`, trained on a masked reconstruction task (Appendix I). The benchmark-frequency breakdowns, standard deviations, and cross-dataset variance trajectories can be found in Appendix G.

We hypothesize that these accuracy improvements stem from more stable and scale-unbiased optimization. Figure 5b follows the median per-dataset MASE throughout the 80,000 optimization steps. The **ScaleIn** develops an advantage during training for all four architectures and retains it at the final checkpoint. Additionally, we observe that **ScaleCon** loss introduces excess difficulty and `TimesFM2.5` is the clearest case of that. Its scale-contaminated run degrades for the first 25,000 steps before recovering, while its scale-invariant run is flat from the start.

Table 2: Forecasting performance by benchmark and architecture. For each seed, we select the checkpoint with the lowest validation MAE. Within each frequency, we average MASE across forecasts and compute WQL as a ratio of sums. We then macro-average both metrics across frequencies and report their means across 4 seeds. **SCALEIN** uses scale-invariant loss and **SCALECON** uses scale-contaminated loss. Lower is better, and **bold** marks the better loss within each architecture.

Benchmark	Metric	Neural Forecast Baselines										Reference		
		N-HiTS		N-BEATS		PatchTST		TCN		LSTM		S.Naive	ETS	ARIMA
		SC.IN	SC.CON	SC.IN	SC.CON	SC.IN	SC.CON	SC.IN	SC.CON	SC.IN	SC.CON			
M1	MASE	**1.826**	1.855	**1.822**	1.835	1.917	**1.911**	**1.941**	1.987	**1.998**	2.057	2.478	1.904	1.977
	WQL	0.099	**0.098**	0.099	**0.098**	**0.106**	**0.106**	**0.100**	0.101	**0.105**	0.106	0.167	0.108	0.132
M3	MASE	1.652	**1.640**	**1.677**	1.767	**2.802**	3.083	**1.896**	2.060	**2.133**	2.134	2.011	1.477	1.687
	WQL	**0.090**	0.091	**0.089**	0.092	**0.116**	0.123	**0.099**	0.103	0.109	**0.107**	0.103	0.087	0.090
M4	MASE	**1.991**	2.007	**1.987**	2.036	3.639	**3.572**	**2.090**	2.198	**2.264**	2.268	2.446	2.030	2.351
	WQL	**0.072**	**0.072**	**0.072**	0.073	**0.093**	0.093	**0.075**	0.076	**0.080**	0.081	0.083	0.071	0.079
Tourism	MASE	**2.346**	2.420	**2.307**	2.392	**2.884**	2.960	**2.668**	2.688	2.643	**2.614**	2.232	2.259	2.503
	WQL	**0.133**	0.137	**0.132**	0.135	**0.164**	0.167	**0.153**	**0.153**	0.153	**0.152**	0.142	0.133	0.156

4.4 SUPERVISED NEURAL FORECASTING MODELS

The experiments in Section 4.3 establish that the scale-contamination effect persists under TSFM-scale, multi-dataset pretraining, affecting forecast error on both in-distribution and zero-shot benchmarks. Beyond the TSFM pretraining setting, we show that forecast error improvements with **SCALEIN** extend to standard deep forecasters trained across heterogeneous datasets on a smaller scale, with supervised evaluation only. We train 5 established architectures, `NHITS` (Challu et al., 2023), `NBEATS` (Oreshkin et al., 2020), `PatchTST` (Nie et al., 2023), `TCN` (Bai et al., 2018), and `LSTM` (Hochreiter and Schmidhuber, 1997), on M1, M3, M4, and Tourism, fitting 6 separate models, one for each sampling frequency (i.e., yearly, monthly, quarterly, weekly, daily, and hourly) by aggregating training splits across datasets. All neural networks runs share data splits, Adam optimizer (lr $= 10^{-3}$), and a 20,000-step budget across 4 seeds, selecting checkpoints via minimum validation MAE. See Appendix H for complete data, standard deviation, and aggregation details. For reference, we compare with classical statistical baselines `S.Naive`, `ETS`, and `ARIMA` (Hyndman et al., 2026).

Table 2 shows that the scale-invariant loss lowers MASE in 16 of 20 matched architecture-benchmark comparisons and WQL in 15 of 20. At the finer benchmark-frequency level, **SCALEIN** lowers MASE in 49 of 75 comparisons and WQL in 43 of 75, with the complete breakdown and standard deviations reported in Appendix H. Averaged over all architectures, benchmarks, and seeds, **SCALEIN** lowers MASE by 1.9% and WQL by 1.1% The effect is therefore broad, but rather modest, and it varies between architectures and benchmarks. The fitted statistical references remain competitive and lead the neural models on several benchmark-metric pairs.

5 CONCLUSION

In this work, we show that reversing an affine scaling before computing a degree-p homogeneous loss injects a b^p factor into the gradient that grows with series scale, imposing an implicit curriculum that favors high-scale series. The remedy is a one-line code change: compute the loss directly on scaled targets. We confirmed this across the full spectrum from controlled synthetic studies to real-world pretraining. Scale-invariant loss (**SCALEIN**) closes the convergence gap on synthetic data and is invariant to artificial scale assignments on real data (Sections 4.1 and 4.2). Employing **SCALEIN** in TSFM pre-training leads on average to substantial improvements in MASE and WQL performance metrics, reaching 18.8% on GIFT-Eval and 21.9% on M-competitions (Section 4.3). The effect on neural forecasting models is broadly positive, but expectedly modest (Section 4.4). Based on the results, we thus posit that loss space is a crucial training decision – especially for TSFMs – that should be made deliberately and reported explicitly in any forecasting paper to promote reproducibility.

Our findings motivate a few directions for future work. While the same scale bias persists in multivariate settings with independent scaling of each series, its practical impact and the effects of more general normalization schemes that jointly transform correlated channels remain to be studied. Another direction is to characterize how loss-space choices interact with probabilistic objectives and their parameterizations, beyond the homogeneous losses studied here.

AI USE STATEMENT

In this work, we used generative AI tools to collaboratively implement methods, clean and reformat datasets, write code, create visualizations, and edit the paper to improve readability. We have carefully reviewed all AI-assisted work. Any generated code and text was thoroughly verified and checked for correctness. AI tools have not been used as autonomous agents, but rather as smart assistants performing small and precise tasks that we scrupulously checked for correctness.

We have not used these tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, supporting qualitative and thematic data analysis, and interpreting results.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 states the assumptions behind the theoretical results, and Appendix B provides complete proofs. Algorithm 1 shows the practical change in loss computation, therefore, the **SCALEIN** mechanism can be re-implemented from the paper alone. All experiments have a reproducibility appendix detailing its data construction, model configuration, optimizer settings, seeds, and evaluation protocol: Appendix E.1 for the synthetic study, Appendices F.1 and F.2 for the scale-swap experiment, Appendix G.1 for the pretraining runs, Appendix H for the supervised study. Datasets and preprocessing techniques are described in Appendices D.1 and D.2. We will release the complete training repository, including scripts that produce every figure and table in this paper, upon publication.

REFERENCES

I. Ahmed, D. Krompaß, C. Feng, and V. Tresp. A comparative study on how data normalization affects zero-shot generalization in time series foundation models. In *ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3436–3440, 2026.

T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, S. Savarese, C. Xiong, and D. Sahoo. GIFT-eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024. URL https://openreview.net/forum?id=Z2cMOOANFX.

A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL http://jmlr.org/papers/v21/19-820.html.

A. F. Ansari, O. Shchur, J. Küken, A. Auer, B. Han, P. Mercado, S. S. Rangapuram, H. Shen, L. Stella, X. Zhang, M. Goswami, S. Kapoor, D. C. Maddix, P. Guerron, T. Hu, J. Yin, N. Erickson, P. M. Desai, H. Wang, H. Rangwala, G. Karypis, Y. Wang, and M. Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu. The Tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2010.04.009. URL https://www.sciencedirect.com/science/article/pii/S016920701000107X. Special Section 1: Forecasting with Artificial Neural Networks and Computational Intelligence. Special Section 2: Tourism Forecasting.

A. Auer, P. Podest, D. Klotz, S. Böck, G. Klambauer, and S. Hochreiter. TiRex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

J. Beitner. PyTorchForecasting: Forecasting with neural networks made simple. GitHub Repository, 2020. URL https://pytorch-forecasting.readthedocs.io/en/stable/.

G. Berthelier, T. Nabil, E. L. Naour, R. Niamke, S. Perlaza, and G. Neglia. On the role of reversible instance normalization, 2026. URL https://arxiv.org/abs/2603.11869.

C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski. NHITS: Neural Hierarchical Interpolation for Time Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):6989–6997, June 2023. ISSN 2374-3468. URL https://ojs.aaai.org/index.php/AAAI/article/view/25854.

B. Cohen, E. Khwaja, Y. Doubli, S. Lemaachi, C. Lettieri, C. Masson, H. Miccinilli, E. Ramé, Q. Ren, A. Rostamizadeh, J. O. du Terrail, A.-M. Toon, K. Wang, S. Xie, Z. Xu, V. Zhukova, D. Asker, A. Talwalkar, and O. Abou-Amal. This time is different: An observability perspective on time series foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1jDAYXfcS2.

A. Cook, DanB, inversion, and R. Holbrook. Store sales - time series forecasting. https://www.kaggle.com/competitions/store-sales-time-series-forecasting, 2021. Kaggle.

A. Das, W. Kong, R. Sen, and Y. Zhou. A decoder-only foundation model for time-series forecasting. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10148–10167. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/das24c.html.

V. Ekambaram, A. Jati, P. Dayama, S. Mukherjee, N. H. Nguyen, W. M. Gifford, C. Reddy, and J. Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 74147–74181. Curran Associates, Inc., 2024. doi: 10.52202/079017-2359. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/874a4d89f2d04b4bcf9a2c19545cf040-Paper-Conference.pdf.

K. Feng, S. Lan, Y. Fang, W. He, S. Lu, S. Gu, L. Ma, X. Lu, and K. Ren. Kairos: Toward adaptive and parameter-efficient time series foundation models, 2025. URL https://arxiv.org/abs/2509.25826.

Y. Fu, Z. Shao, C. Yu, Y. Li, Y. Xu, X. Cheng, and F. Wang. Zeus: Towards tuning-free foundation model for time series analysis. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Dtue4WLYSk.

A. Garza, M. M. Canseco, C. Challú, and K. G. Olivares. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022. URL https://github.com/Nixtla/statsforecast.

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

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

M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski. MOMENT: A family of open time-series foundation models. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16115–16152. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/goswami24a.html.

L. Graf, T. Ortner, S. Woźniak, and A. Pantazi. Flowstate: Sampling-rate-equivariant time-series forecasting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=9yNGBueKgZ.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *Journal of Economic and Social Measurement*, 29(1-3):123–125, 2004. URL https://journals.sagepub.com/doi/abs/10.3233/JEM-2004-0211.

A. Howard, H. Yui, M. McDonald, and W. Cukierski. Recruit restaurant visitor forecasting. https://www.kaggle.com/competitions/recruit-restaurant-visitor-forecasting, 2017. Kaggle Competition.

R. J. Hyndman and B. Billah. Unmasking the theta method. *International Journal of Forecasting*, 19(2):287–290, 2003. ISSN 0169-2070. URL https://www.sciencedirect.com/science/article/pii/S0169207001001431.

R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22, 2008. doi: 10.18637/jss.v027.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v027i03.

R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. ISSN 0169-2070. URL https://www.sciencedirect.com/science/article/pii/S0169207006000239.

R. J. Hyndman, G. Athanasopoulos, A. Garza, C. Challu, M. Mergenthaler, and K. G. Olivares. *Forecasting: Principles and Practice, the Pythonic Way*. OTexts, Melbourne, Australia, 2026. available at https://otexts.com/fpppy/.

A. Jain and R. Sen. TimesFM-3: A zero-shot foundation model for multivariate forecasting. https://research.google/blog/timesfm-3-a-zero-shot-foundation-model-for-multivariate-forecasting/, Aug. 2026. Google Research Blog.

M. Jin, S. Wang, L. Ma, Z. Chu, J. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen. Time-llm: Time series forecasting by reprogramming large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 23857–23880, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/680b2a8135b9c71278a09cafb605869e-Paper-Conference.pdf.

E. Khwaja, C. Lettieri, G. Woo, E. Belouadah, M. Cenac, G. Jarry, E. Paquin, X. Zhao, V. Zhukov, O. Abou-Amal, C. Liu, A. Talwalkar, and D. Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026. URL https://arxiv.org/abs/2605.20119.

T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

C. Liu, T. Aksu, J. Liu, X. Liu, H. Yan, Q. Pham, S. Savarese, D. Sahoo, C. Xiong, and J. Li. Moirai 2.0: When less is more for time series forecasting, 2025a. URL https://arxiv.org/abs/2511.11698.

Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9881–9893. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/4054556fcaa934b0bf76da52cf4f92cb-Paper-Conference.pdf.

Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. iTransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Y. Liu, G. Qin, Z. Shi, Z. Chen, C. Yang, X. Huang, J. Wang, and M. Long. Sundial: A family of highly capable time series foundation models. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=LO7ciRpjI5.

Y. Liu, X. Su, S. Wang, H. Zhang, H. Liu, Y. Wang, Z. Ye, Y. Xiang, J. Wang, and M. Long. Timer-s1: A billion-scale time series foundation model with serial scaling, 2026. URL https://arxiv.org/abs/2603.04791.

S. Makridakis and M. Hibon. The M3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. ISSN 0169-2070. doi: 10.1016/S0169-2070(00)00057-1. URL https://www.sciencedirect.com/science/article/pii/S0169207000000571. The M3- Competition.

S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, 1982. doi: 10.1002/for.3980010202. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980010202.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2019.04.014. URL https://www.sciencedirect.com/science/article/pii/S0169207019301128. M4 Competition.

P. Mancuso, V. Piccialli, and A. M. Sudoso. A machine learning approach for forecasting hierarchical time series. *Expert Systems with Applications*, 182:115102, 2021. ISSN 0957-4174. URL https://www.sciencedirect.com/science/article/pii/S0957417421005431.

M. V. Ness, H. Shen, H. Wang, X. Jin, D. C. Maddix, and K. Gopalswamy. Cross-frequency time series meta-forecasting, 2023. URL https://arxiv.org/abs/2302.02077.

Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

K. G. Olivares, C. Challú, A. Garza, M. M. Canseco, and A. Dubrawski. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022, 2022. URL https://github.com/Nixtla/neuralforecast.

K. G. Olivares, M. Wolff, T. Konstantinova, S. Ramasubramanian, B. N. Oreshkin, A. G. Wilson, R. K. Vadlamani, A. Potapczynski, W. Potosnak, M. W. Mahoney, M. Cao, and D. Efimov. A more realistic evaluation of cross-frequency transfer learning and foundation forecasting models. In *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?*, 2025. URL https://openreview.net/forum?id=hPqpEzumiC.

B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

S. Palaskar, V. Ekambaram, A. Jati, N. Gantayat, A. Saha, S. Nagar, N. H. Nguyen, P. Dayama, R. Sindhgatta, P. Mohapatra, et al. Automixer for improved multivariate time-series forecasting on business and it observability data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 22962–22968, 2024.

N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Deep adaptive input normalization for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9): 3760–3765, 2020.

P. Podest, M. Pichler, E. Bürger, L. Zólyomi, B. Voggenberger, W. Berghammer, D. Klotz, S. Böck, G. Klambauer, and S. Hochreiter. Tirex-2: Generalizing tirex to multivariate data and streaming, 2026. URL https://arxiv.org/abs/2607.01204.

W. Potosnak, M. Wolff, M. Cao, R. Ma, T. Konstantinova, D. Efimov, M. W. Mahoney, B. N. Oreshkin, and K. G. Olivares. Forking-sequences: Statistically and computationally efficient multi-horizon forecasting with reduced volatility. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=dXdycy7WCX.

K. Rasul, A. Ashok, A. R. Williams, A. Khorasani, G. Adamopoulos, R. Bhagwatkar, M. Biloš, H. Ghonia, N. Hassen, A. Schneider, S. Garg, A. Drouin, N. Chapados, Y. Nevmyvaka, and I. Rish. Lag-Llama: Towards foundation models for time series forecasting. In *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models (NeurIPS 2023 Workshop)*, 2023. URL https://openreview.net/forum?id=jYluzCLFDM.

S. Shen, V. Van Beek, and A. Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 465–474, 2015. doi: 10.1109/CCGrid.2015.60.

X. Shi, S. Wang, Y. Nie, D. Li, Z. Ye, Q. Wen, and M. Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 34635–34667, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/558d48c1f08675daa636e09bfe94a89e-Paper-Conference.pdf.

S. Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020. ISSN 0169-2070. URL https://www.sciencedirect.com/science/article/pii/S0169207019301153.

A. A. Syntetos, J. E. Boylan, and J. D. Croston. On the categorization of demand patterns. *Journal of the Operational Research Society*, 56(5):495–503, 2005. URL https://doi.org/10.1057/palgrave.jors.2601841.

J. W. Taylor. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *Journal of Forecasting*, 19(4):299–311, 2000. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/1099-131X%28200007%2919%3A4%3C299%3A%3AAID-FOR775%3E3.0.CO%3B2-V.

A. Trindade. ElectricityLoadDiagrams20112014 dataset. UCI Machine Learning Repository, 2015. URL https://archive.ics.uci.edu/ml/datasets/electricityloaddiagrams20112014.

E. Urettini, D. Atzeni, R. J. Ramjattan, and A. Carta. Gas-norm: Score-driven adaptive normalization for non-stationary time series forecasting in deep learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 2282–2291. ACM, Oct. 2024. doi: 10.1145/3627673.3679822. URL http://dx.doi.org/10.1145/3627673.3679822.

S.-M. Vilhes, G. Gasso, and M. Z. Alaya. Does normalization choice matter for causal large time-series models?, 2026. URL https://arxiv.org/abs/2606.09954. ICLR 2026 Workshop on Time Series in the Age of Large Models (TSALM).

J. Wang, J. Jiang, W. Jiang, C. Li, and W. X. Zhao. Libcity: An open library for traffic prediction. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '21, page 145–148, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386647. doi: 10.1145/3474717.3483923. URL https://doi.org/10.1145/3474717.3483923.

X. Wang, T. Zhou, J. Gao, B. Ding, and J. Zhou. Output scaling: Yinglong-delayed chain of thought in a large pretrained time series forecasting model, 2025. URL https://arxiv.org/abs/2506.11029.

Y. Wen, W. M. Gifford, C. Reddy, L. M. Nguyen, J. Kalagnanam, and A. A. Julius. Revisiting the generic transformer: Deconstructing a strong baseline for time series foundation models, 2026. URL https://arxiv.org/abs/2602.06909.

M. Wolff, M. Li, R. K. Selvam, H. Zhu, K. G. Olivares, R. Ma, A. Katoch, S. Ramasubramanian, M. Cao, R. Bandarra, R. Gopalsamy, S. L. Vattiata, S. Yang, and M. W. Mahoney. SPADE-S: A sparsity-robust foundational forecaster. In *KDD 2025 Workshop on AI for Supply Chain: Today and Future*, 2025. URL https://openreview.net/forum?id=HQWXh2BQTv.

G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo. Unified training of universal time series forecasting transformers. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 53140–53164. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22419–22430. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.

H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

H. Xia, Y. Liu, Y. Hu, P. Liu, and Z. Dong. Into the orbit for time series: Training regimes for foundation models, 2026. URL https://arxiv.org/abs/2608.13262.

H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. 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.

A SCALE-INVARIANCE AND CONTAMINATION IN THE TSFM OPEN-SOURCE ECOSYSTEM

Table 3: A survey of loss spaces used in time series foundation models. **Explicit**: the paper states the loss space explicitly in text or in equations that define the normalized or denormalized quantities entering the loss. **Inferred**: it is possible to trace the target from the input through the normalization to the loss and infer with high likelihood what is the loss space. **Code**: the loss space is determined by the publicly released code. A dash means that we were not able to determine the loss space (or loss function) from that source.

Model		Loss Function	Explicit	Inferred	Code
TimeGPT-1	Garza et al. (2023)	–	–	–	–
LagLLama	Rasul et al. (2023)	Student-t NLL	–	SCALECON ¶	SCALECON ¶
Time-LLM	Jin et al. (2024)	MSE	–	SCALECON	SCALECON
Moirai	Woo et al. (2024)	Mixture NLL	–	SCALECON ¶	SCALECON ¶
MOMENT	Goswami et al. (2024)	MSE	–	SCALECON	SCALECON ‡
TimesFM	Das et al. (2024)	MSE	–	SCALECON	SCALECON *‡
TimesFM-2.5	Das et al. (2024)	Quantile	–	–	SCALECON *
TimesFM-3	Jain and Sen (2026)	Quantile	–	–	–
Moirai-2.0	Liu et al. (2025a)	Quantile	–	SCALECON	SCALEIN *
Chronos2	Ansari et al. (2025)	Quantile	SCALEIN	SCALEIN	SCALEIN
Toto	Cohen et al. (2025)	NLL + Cauchy	–	–	SCALEIN *
Toto-2.0	Khwaja et al. (2026)	Quantile	SCALEIN	SCALEIN	SCALEIN *
TiRex	Auer et al. (2025)	Quantile	SCALECON	SCALECON	–
TiRex-2	Podest et al. (2026)	Quantile	SCALECON	SCALECON	SCALECON *
Kairos	Feng et al. (2025)	Weighted quantile	–	–	SCALEIN
Sundial	Liu et al. (2025b)	TimeFlow	–	–	–
YingLong	Wang et al. (2025)	Weighted quantile	SCALECON †	SCALECON †	SCALECON †
Falcon 2	Xia et al. (2026)	Quantile	SCALEIN	SCALEIN	–
Zeus	Fu et al. (2026)	Quantile	–	–	SCALEIN *
Timer-S1	Liu et al. (2026)	Weighted quantile	SCALECON †	SCALECON †	–
Time-MoE	Shi et al. (2025)	Huber	–	–	SCALECON §
TTM	Ekambaram et al. (2024)	MSE	SCALECON	SCALECON	SCALECON
FlowState	Graf et al. (2026)	Quantile	SCALECON	SCALECON	–
PatchTST-FM	Wen et al. (2026)	Quantile	–	SCALEIN	SCALEIN

*Inferred from released model, fine-tuning, or inference code, as the pretraining code is not released. ¶If NLL is the loss function, the original-space loss differs from the normalized one only by a constant $\log \sigma$ which does not affect gradients. †Pinball loss divided by the mean absolute target. ‡Additionally standardization globally per dataset and channel. §Series are globally standardized, with no per-window scaling.

Table 4: Training properties of time series forecasting codebases.

Codebase	Reference	SCALECON	Invariant (SCALEIN)	Fixed/User-specified
GluonTS	Alexandrov et al. (2020)	NLL, CRPS, EnergyScore	—	Fixed
PyTorch Forecasting	Beitner (2020)	MAE, RMSE, MAPE, SMAPE, QuantileLoss, NormalDist, LogNormal, NegBinomial, Beta, MQF2	—	Fixed
TSLib	Wu et al. (2023)	—	MSE, MAPE, MASE, SMAPE	Fixed
NeuralForecast	Olivares et al. (2022)	NLL (Normal, StudentT, NegBinomial, Poisson, Tweedie)	MAE, MSE, RMSE, MAPE, SMAPE, MASE, QuantileLoss, MQLoss	Fixed

Training loss space is a fundamental design choice that directly affects what a model learns, yet it is not consistently documented in papers or codebases. Tables 3 and 4 survey the loss space used by prominent TSFMs and open-source forecasting codebases respectively. Across the 24 models in the survey, we distinguish explicit reporting from conclusions drawn indirectly from the available sources. The papers of 9 models explicitly specify the loss space. For 7 others, the descriptions of normalization, outputs, and the objective suggest a likely loss space without establishing it explicitly. Inspecting released code provides a classification for a further 5 models, leaving 3 unresolved. Under

these classifications, 7 models use **ScaleIn**. The survey thus identifies both scale-contaminated and scale-invariant training in current practice, with substantial gaps in how these choices are reported.

We classify each model from its paper and its public code separately. A paper counts as *explicit* if it states the loss space in text, or if its equations define the normalized or denormalized quantities that enter the loss. A paper counts as *inferred* if it gives no such statement, but its description of the normalization, the model outputs, and the loss lets us trace the target from the input to the loss and infer the loss space with high likelihood. We note, that citing RevIN (Kim et al., 2022) or writing the loss on a generic target y is not enough on its own to infer either **ScaleIn** or **ScaleCon**. In code column, we mark TSFMs that released code which allows to determine where the loss is computed, and we flag cases where this follows from model, fine-tuning, or inference code rather than a released pretraining loop.

Furthermore, prominent open-source codebases do not explicitly document loss space choice, nor support user specification of the loss space. The lack of documentation raises reproducibility concerns because practitioners who train, fine-tune or extend models may inadvertently change the loss space without realizing it, producing results that are not directly comparable to the original. More broadly, without awareness of the implications, practitioners may simply propagate the use of scale-contaminated losses, which we show has consequences for training convergence and downstream accuracy in cross-frequency training scenarios. We hypothesize that the lack of proper loss space reporting is a result of the widespread adoption of instance normalization schemes such as RevIN, which (by default) denormalize predictions before returning them. However, as we show in this work with both theoretical and empirical evidence, loss space selection is an important design choice.

B Scale-Contamination and Scale-Invariance Proofs

Proof of Lemma 3.1. For one series contribution, the observed and predicted targets on the original scale are $y_{t+h} = bz_{t+h} + a$ and $\hat{y}_{t+h} = b\hat{z}_{t+h} + a$. Their residual is therefore

$$y_{t+h} - \hat{y}_{t+h} = b\left(z_{t+h} - \hat{z}_{t+h}\right). \tag{13}$$

Substituting Equation 13 into the scale-contaminated objective and applying $\phi(br) = b^p\phi(r)$ at every horizon step gives

$$\mathcal{L}_{\text{contam}} = \frac{1}{H}\sum_{h=1}^{H}\phi\left(b\left(z_{t+h} - \hat{z}_{t+h}\right)\right) \tag{14}$$

$$= \frac{b^p}{H}\sum_{h=1}^{H}\phi\left(z_{t+h} - \hat{z}_{t+h}\right) \tag{15}$$

$$= b^p\mathcal{L}_{\text{inv}}. \tag{16}$$

The scaling offset a cancels, while the scaling denominator remains as the multiplicative factor b^p. The statistics a and b depend only on the observed history, so they are constant with respect to the model parameters $\boldsymbol{\theta}$. Differentiating the loss identity therefore yields

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{contam}} = b^p\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv}}. \tag{17}$$

When ϕ is nondifferentiable, multiplication by the positive constant b^p scales every valid subgradient by the same factor. For a mini-batch $\mathcal{B}$, linearity of summation then gives

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{contam},\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} b_i^p\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv},i}, \tag{18}$$

which is Equation 10. □

Proof of Theorem 3.2. Consider multiplying every observation of one series by an arbitrary constant $c > 0$. Write $y_s^{(c)} = cy_s$ for the rescaled observation. Scale equivariance gives $a^{(c)} = ca$ and $b^{(c)} = cb$, so every scaled context and target value satisfies

$$z_s^{(c)} = \frac{y_s^{(c)} - a^{(c)}}{b^{(c)}} = \frac{cy_s - ca}{cb} = \frac{y_s - a}{b} = z_s. \tag{19}$$

The model consequently receives the same scaled context and produces the same scaled prediction for every $\boldsymbol{\theta}$,

$$\hat{\mathbf{z}}^{(c)}_{t+1:t+H} = f_{\boldsymbol{\theta}}\left(\mathbf{z}^{(c)}_{t-L+1:t}\right) = f_{\boldsymbol{\theta}}\left(\mathbf{z}_{t-L+1:t}\right) = \hat{\mathbf{z}}_{t+1:t+H}. \tag{20}$$

Equations 19 and 20 make the scale-invariant objective identical before and after the rescaling,

$$\mathcal{L}^{(c)}_{\text{inv}}(\boldsymbol{\theta}) = \frac{1}{H}\sum_{h=1}^{H} \phi\left(z^{(c)}_{t+h} - \hat{z}^{(c)}_{t+h}\right) \tag{21}$$

$$= \frac{1}{H}\sum_{h=1}^{H} \phi\left(z_{t+h} - \hat{z}_{t+h}\right) \tag{22}$$

$$= \mathcal{L}_{\text{inv}}(\boldsymbol{\theta}). \tag{23}$$

Because this equality holds as a function of $\boldsymbol{\theta}$, the parameter gradients are also identical,

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{(c)}_{\text{inv}}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv}}(\boldsymbol{\theta}). \tag{24}$$

The rescaling constants may differ across series because Equation 24 holds for every $c > 0$ separately. If series i is multiplied by $c_i > 0$, the scale-invariant mini-batch gradient therefore satisfies

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{(\mathbf{c})}_{\text{inv},\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_{\boldsymbol{\theta}}\mathcal{L}^{(c_i)}_{\text{inv},i} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv},i} = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv},\mathcal{B}}. \tag{25}$$

For contrast, the same constant leaks directly into a scale-contaminated gradient. After rescaling, the scaling denominator in the inverse transform is $b^{(c)} = cb$, so Lemma 3.1 gives

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{(c)}_{\text{contam}} = (b^{(c)})^p \nabla_{\boldsymbol{\theta}}\mathcal{L}^{(c)}_{\text{inv}} \tag{26}$$

$$= (cb)^p \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv}} \tag{27}$$

$$= c^p \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{contam}}. \tag{28}$$

Thus **ScaleIn** cancels c before the loss is evaluated, while a scale-contaminated loss multiplies the same series contribution by c^p. For the same independent rescalings, the scale-contaminated mini-batch gradient becomes

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{(\mathbf{c})}_{\text{contam},\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} (c_i b_i)^p \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{inv},i}, \tag{29}$$

Comparing Equations 25 and 29 shows that only the scale-contaminated update changes with the chosen c_i values. Fix the initialization, mini-batch sequence, optimizer state, and optimizer randomness as stated in the theorem. At the first step, Equation 25 gives both scale-invariant runs the same gradient, so the optimizer produces the same next parameters and state. Induction over training steps proves that the complete scale-invariant optimization trajectory is unchanged. □

B.1 Extensions to Multivariate and Quantile Forecasting

Multivariate forecasting. Suppose the target has D channels and channel j uses scaling offset a_j and denominator b_j. For a loss ϕ homogeneous of degree p, the scale-invariant and scale-contaminated objectives are

$$\mathcal{L}^{\text{multi}}_{\text{inv}} = \frac{1}{HD}\sum_{h=1}^{H}\sum_{j=1}^{D} \phi\left(z_{t+h,j} - \hat{z}_{t+h,j}\right), \tag{30}$$

$$\mathcal{L}^{\text{multi}}_{\text{contam}} = \frac{1}{HD}\sum_{h=1}^{H}\sum_{j=1}^{D} b_j^p \phi\left(z_{t+h,j} - \hat{z}_{t+h,j}\right). \tag{31}$$

The derivation applies Lemma 3.1 separately to each channel. Multiplying channel j by any $c_j > 0$ leaves its scaled values unchanged under a channel-wise scale-equivariant scaling method, so the proof of Theorem 3.2 applies to the complete scaled input and target tensors. Cross-channel dependencies inside the forecasting model do not affect this conclusion because the model receives the same scaled tensor after rescaling.

Quantile forecasting. Let $\mathcal{Q}$ be a set of quantile levels and let $\hat{z}_{t+h}^{(q)}$ be the scaled prediction for level q. Writing the pinball loss as $\mathrm{QL}_q(r) = q(r)_+ + (1-q)(-r)_+$, the quantile objective is

$$\mathcal{L}_{\text{inv}}^{\text{quant}} = \frac{1}{H|\mathcal{Q}|} \sum_{h=1}^{H} \sum_{q \in \mathcal{Q}} \mathrm{QL}_q \left(z_{t+h} - \hat{z}_{t+h}^{(q)} \right). \tag{32}$$

This objective averages pinball loss across all H horizon steps and all requested quantile levels, so the model learns the quantile forecasts jointly. For every $b > 0$, pinball loss satisfies $\mathrm{QL}_q(br) = b\mathrm{QL}_q(r)$. The same b applies to every quantile level of a series, which gives

$$\mathcal{L}_{\text{contam}}^{\text{quant}} = b\mathcal{L}_{\text{inv}}^{\text{quant}}, \qquad \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{contam}}^{\text{quant}} = b \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{inv}}^{\text{quant}}. \tag{33}$$

Computing pinball loss on scaled targets removes this factor jointly across all quantile levels, so Theorem 3.2 applies without changing the model or the quantile set.

B.2 Nonlinear Scaling in Chronos2

Write $u = \operatorname{arcsinh}((y-a)/b)$ and let $\hat{u}$ be the model output. Reversing the transform gives $\hat{y} = b \sinh(\hat{u}) + a$, so

$$\phi(y - \hat{y}) = b^p \phi\big(\sinh(u) - \sinh(\hat{u})\big). \tag{34}$$

The b^p factor still weights the loss and parameter gradient, but the remaining loss differs from $\phi(u - \hat{u})$ because of the nonlinear inverse. This comparison therefore changes residual geometry as well as scale weighting. Under scale-equivariant statistics, independently rescaling a series leaves u unchanged, so training on transformed targets retains scale invariance. Fixed numerical stabilizers can qualify this invariance as discussed in Section 3.

C Algorithm

Algorithm 1 contrasts the two training paradigms. Both begin identically: a scaling offset a and denominator b are computed from the context window, the context is normalized, and the model predicts in scaled space. The two approaches diverge only in how the loss is computed. In scale-contaminated training, predictions are denormalized back to the original scale before the loss is applied, coupling the gradient magnitude to b^p and thereby making optimization sensitive to the absolute scale of each series. In scale-invariant training, the targets are instead normalized to match the prediction space, and the loss is computed entirely in scaled space, making the gradient independent of b. The key observation is that the difference between the two paradigms reduces to a single step: whether the targets are denormalized predictions or normalized targets are passed to the loss function, a change that amounts to one line of code in practice.

Algorithm 1 Scale-ScaleCon and scale-invariant (ScaleIn) objectives

Require: Context $\mathbf{y}_{t-L+1:t}$, targets $\mathbf{y}_{t+1:t+H}$, model $f_{\boldsymbol{\theta}}$, loss ϕ, scaling method (a, b)

Compute offset a and denominator b from context
Scale context: $z_{t-l} \leftarrow (y_{t-l} - a)/b$ for $l = 0, \ldots, L-1$
Predict scaled outputs: $\hat{\mathbf{z}}_{t+1:t+H} \leftarrow f_{\boldsymbol{\theta}}\left(\mathbf{z}_{t-L+1:t}\right)$
// Scale-ScaleCon objective
Reverse scaling: $\hat{y}_{t+h} \leftarrow b\hat{z}_{t+h} + a$ for $h = 1, \ldots, H$
$\mathcal{L}_{\text{contam}} \leftarrow \frac{1}{H} \sum_{h=1}^{H} \phi\left(y_{t+h} - \hat{y}_{t+h}\right)$ ▷ Loss in original scale; gradient $\propto b^p$
// Scale-invariant (ScaleIn) objective
Scale targets: $z_{t+h} \leftarrow (y_{t+h} - a)/b$ for $h = 1, \ldots, H$
$\mathcal{L}_{\text{inv}} \leftarrow \frac{1}{H} \sum_{h=1}^{H} \phi\left(z_{t+h} - \hat{z}_{t+h}\right)$ ▷ Loss in scaled space; gradient invariant to b
Backpropagate chosen loss and update $\boldsymbol{\theta}$

D DATASET DETAILS

D.1 M-SERIES BENCHMARK DETAILS

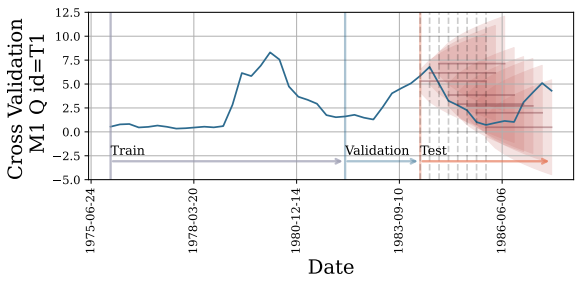


Figure 6: Example time series from the M1 competition dataset. The train and validation sets consist of all observations preceding the first dotted line. The cross validation test set is comprised of a set of forecast creation dates with lengths defined in Table 5.

Table 5: Summary of forecasting datasets used in our empirical study.

	Frequency	Seasonality	Horizon	Series	Min Length	Max Length	% Erratic
M1	Monthly	12	18	617	48	150	0
	Quarterly	4	8	203	18	114	0
	Yearly	1	6	181	15	58	0
M3	Other	4	8	174	71	104	0
	Monthly	12	18	1428	66	144	2
	Quarterly	4	8	756	24	72	1
	Yearly	1	6	645	20	47	10
M4	Hourly	24	48	414	748	1008	17
	Daily	1	14	4,227	107	9933	2
	Weekly	1	13	359	93	2610	16
	Monthly	12	18	48,000	60	2812	6
	Quarterly	4	8	24,000	24	874	11
	Yearly	1	6	23,000	19	841	18
Tourism	Monthly	12	18	366	91	333	51
	Quarterly	4	8	427	30	130	39
	Yearly	1	6	518	11	47	23

The early `M1` competition (Makridakis et al., 1982) competition features 1,001 demographic, industrial, and economic time series spanning varying frequencies with lengths from 9 to 132 observations. Its key finding, that simple methods like ETS (Holt, 2004) often outperform complex approaches, shaped a legacy emphasizing accuracy, automation, and overfitting caution while shifting focus from traditional forecasting.

Held two decades after `M1`, the `M3` competition (Makridakis and Hibon, 2000) featured 3,003 positive-valued time series across business, demographic, financial, and economic domains with lengths from 14 to 126 observations. Reinforcing the trend that simple methods often outperform complex alternatives, the Theta method (Hyndman and Billah, 2003) emerged as the top-performing approach.

The `M4` competition (Makridakis et al., 2020) massively scaled up size and diversity to 100,000 time series across six frequencies and multiple domains. Expanding evaluation beyond point forecasts to include prediction intervals, it featured 18% erratic series (Syntetos et al., 2005) and marked the first time a neural model, ESRNN (Smyl, 2020), outperformed traditional methods.

The `Tourism` dataset (Athanasopoulos et al., 2011) comprises 1,311 monthly, quarterly, and yearly series designed to evaluate demand forecasting. It used Mean Absolute Scaled Error (MASE) for scaled point forecasts and evaluated forecast intervals, with 36% of series classified as erratic or intermittent, making `S.Naive` difficult to beat annually.

D.2 GIFT-EVAL BENCHMARK DETAILS

Table 6: Summary of GiftEval forecasting datasets used in our empirical study.

Dataset	Source	Domain	Frequency	# Series	Avg	Min	Max	# Obs
Jena Weather	Autoformer (Wu et al., 2021)	Nature	10T	1	52,704	52,704	52,704	52,704
Jena Weather	Autoformer (Wu et al., 2021)	Nature	H	1	8,784	8,784	8,784	8,784
Jena Weather	Autoformer (Wu et al., 2021)	Nature	D	1	366	366	366	366
BizITObs - Application	AutoMixer (Palaskar et al., 2024)	Web/CloudOps	10S	1	8,834	8,834	8,834	8,834
BizITObs - Service	AutoMixer (Palaskar et al., 2024)	Web/CloudOps	10S	21	8,835	8,835	8,835	185,535
BizITObs - L2C	AutoMixer (Palaskar et al., 2024)	Web/CloudOps	5T	1	31,968	31,968	31,968	31,968
BizITObs - L2C	AutoMixer (Palaskar et al., 2024)	Web/CloudOps	H	1	2,664	2,664	2,664	2,664
Bitbrains - Fast Storage	Grid Workloads Archive (Shen et al., 2015)	Web/CloudOps	5T	1,250	8,640	8,640	8,640	10,800,000
Bitbrains - Fast Storage	Grid Workloads Archive (Shen et al., 2015)	Web/CloudOps	H	1,250	721	721	721	901,250
Bitbrains - rnd	Grid Workloads Archive (Shen et al., 2015)	Web/CloudOps	5T	500	8,640	8,640	8,640	4,320,000
Bitbrains - rnd	Grid Workloads Archive (Shen et al., 2015)	Web/CloudOps	H	500	720	720	720	360,000
Restaurant	Recruit Rest. Comp. (Howard et al., 2017)	Sales	D	807	358	67	478	289,303
ETT1	Informer (Zhou et al., 2021)	Energy	15T	1	69,680	69,680	69,680	69,680
ETT1	Informer (Zhou et al., 2021)	Energy	H	1	17,420	17,420	17,420	17,420
ETT1	Informer (Zhou et al., 2021)	Energy	D	1	725	725	725	725
ETT1	Informer (Zhou et al., 2021)	Energy	W-THU	1	103	103	103	103
ETT2	Informer (Zhou et al., 2021)	Energy	15T	1	69,680	69,680	69,680	69,680
ETT2	Informer (Zhou et al., 2021)	Energy	H	1	17,420	17,420	17,420	17,420
ETT2	Informer (Zhou et al., 2021)	Energy	D	1	725	725	725	725
ETT2	Informer (Zhou et al., 2021)	Energy	W-THU	1	103	103	103	103
Loop Seattle	LibCity (Wang et al., 2021)	Transport	5T	323	105,120	105,120	105,120	33,953,760
Loop Seattle	LibCity (Wang et al., 2021)	Transport	H	323	8,760	8,760	8,760	2,829,480
Loop Seattle	LibCity (Wang et al., 2021)	Transport	D	323	365	365	365	117,895
SZ-Taxi	LibCity (Wang et al., 2021)	Transport	15T	156	2,976	2,976	2,976	464,256
SZ-Taxi	LibCity (Wang et al., 2021)	Transport	H	156	744	744	744	116,064
M_DENSE	LibCity (Wang et al., 2021)	Transport	H	30	17,520	17,520	17,520	525,600
M_DENSE	LibCity (Wang et al., 2021)	Transport	D	30	730	730	730	21,900
Solar	LSTNet (Lai et al., 2018)	Energy	10T	137	52,560	52,560	52,560	7,200,720
Solar	LSTNet (Lai et al., 2018)	Energy	H	137	8,760	8,760	8,760	1,200,120
Solar	LSTNet (Lai et al., 2018)	Energy	D	137	365	365	365	50,005
Solar	LSTNet (Lai et al., 2018)	Energy	W-FRI	137	52	52	52	7,124
Hierarchical Sales	Mancuso et al. (2021)	Sales	D	118	1,825	1,825	1,825	215,350
Hierarchical Sales	Mancuso et al. (2021)	Sales	W-WED	118	260	260	260	30,680
M4 Yearly	Monash (Godahewa et al., 2021)	Econ/Fin	A-DEC	22,974	37	19	284	845,109
M4 Quarterly	Monash (Godahewa et al., 2021)	Econ/Fin	Q-DEC	24,000	100	24	874	2,406,108
M4 Monthly	Monash (Godahewa et al., 2021)	Econ/Fin	M	48,000	234	60	2,812	11,246,411
M4 Weekly	Monash (Godahewa et al., 2021)	Econ/Fin	W-SUN	359	1,035	93	2,610	371,579
M4 Daily	Monash (Godahewa et al., 2021)	Econ/Fin	D	4,227	2,371	107	9,933	10,023,836
M4 Hourly	Monash (Godahewa et al., 2021)	Econ/Fin	H	414	902	748	1,008	373,372
Hospital	Monash (Godahewa et al., 2021)	Healthcare	M	767	84	84	84	64,428
COVID Deaths	Monash (Godahewa et al., 2021)	Healthcare	D	266	212	212	212	56,392
US Births	Monash (Godahewa et al., 2021)	Healthcare	D	1	7,305	7,305	7,305	7,305
US Births	Monash (Godahewa et al., 2021)	Healthcare	W-TUE	1	1,043	1,043	1,043	1,043
US Births	Monash (Godahewa et al., 2021)	Healthcare	M	1	240	240	240	240
Saugeen	Monash (Godahewa et al., 2021)	Nature	D	1	23,741	23,741	23,741	23,741
Saugeen	Monash (Godahewa et al., 2021)	Nature	W-THU	1	3,391	3,391	3,391	3,391
Saugeen	Monash (Godahewa et al., 2021)	Nature	M	1	780	780	780	—
Temperature Rain	Monash (Godahewa et al., 2021)	Nature	D	32,072	725	725	725	780
KDD Cup 2018	Monash (Godahewa et al., 2021)	Nature	H	270	10,898	9,504	10,920	2,942,364
KDD Cup 2018	Monash (Godahewa et al., 2021)	Nature	D	270	455	396	455	122,791
Car Parts	Monash (Godahewa et al., 2021)	Sales	M	2,674	51	51	51	136,374
Electricity	UCI ML Archive (Trindade, 2015)	Energy	15T	370	140,256	140,256	140,256	51,894,720
Electricity	UCI ML Archive (Trindade, 2015)	Energy	H	370	35,064	35,064	35,064	12,973,680
Electricity	UCI ML Archive (Trindade, 2015)	Energy	D	370	1,461	1,461	1,461	540,570
Electricity	UCI ML Archive (Trindade, 2015)	Energy	W-FRI	370	208	208	208	76,960

The GiftEval (Aksu et al., 2024) benchmark leverages 10 distinct data sources to construct a comprehensive testing suite. This incorporates the Jena Weather collection from Autoformer (Wu et al., 2021), business and IT infrastructure observability data AutoMixer (Palaskar et al., 2024), cloud workloads sourced from the Grid Workloads Archive via Bitbrains (Shen et al., 2015), temperature profiles from Informer (Zhou et al., 2021), solar generation series from LSTNet (Lai et al., 2018), sales histories (Mancuso et al., 2021), recruit restaurant visitors (Howard et al., 2017), urban traffic logs from LibCity (Wang et al., 2021), selected non-overlapping subsets from the Monash repository (Godahewa et al., 2021), and smart-meter electricity tracking from the UCI Machine Learning Archive (Trindade, 2015). Summary statistics for each source including frequencies, prediction lengths, variate counts, and scale, are detailed in Table 6.

E Controlled Synthetic Data Experiment Details

E.1 Reproducibility Details

Data. We generate 3 noiseless sinusoidal sources. For source k and training phase s,

$$y_t^{(k,s)} = 1 + \sqrt{2v_k}\sin\left(\frac{2\pi f_k t}{32} + \phi_k + \frac{2\pi s}{8}\right), \tag{35}$$

where (v_k, f_k, ϕ_k) is $(1, 1, 0)$, $(10, 2, \pi/3)$, or $(100, 3, 2\pi/3)$. Each 64-point context has population variance v_k, giving scaling denominators in the approximate ratio $1:\sqrt{10}:10$ and MSE multipliers b^2 in the ratio $1:10:100$. Each source has 8 training series of length 2,048. We use stride-1 windows with context length 64 and horizon 16. Validation contains 512 fixed windows per source sampled from 4 series on a phase grid shifted by $\pi/8$ from training.

Model and scaling. The univariate patch Transformer uses 8 nonoverlapping patches of length 8, embedding dimension 64, 2 encoder layers, 4 attention heads, feedforward width 128, dropout 0, and a direct linear head for the 16-step horizon. Each context is scaled by its sample mean and sample standard deviation with $\epsilon = 10^{-5}$. We compute MSE either on scaled targets for the scale-invariant condition or after reversing the affine transform for the scale-contaminated condition.

Optimization. We train for 30,000 updates with batch size 24 (8 windows per source) and seeds $\{0, 1, 2, 3, 4\}$. SGD uses learning rate 10^{-3}, momentum 0, and weight decay 0. Adam uses the same learning rate with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 0. Learning rates are constant. Within each seed, all conditions share the initialization and sampled batches.

Controls and evaluation. The equal-scale control sets all amplitudes to 1. The gradient-matched control rescales the full parameter gradient to unit Euclidean norm after every backward pass. We report validation nMSE on scaled targets as the mean ± 1 standard deviation over 5 seeds. Evaluation occurs every 5 updates through step 200, every 50 through step 1,000, and every 1,000 thereafter.

E.2 Scale-invariance Comparison Under Adam

A natural objection to the controlled synthetic experiment of Section 4.1 is that an adaptive optimizer should absorb the b^p factor. Adam (Kingma and Ba, 2015) is invariant to a global rescaling of the loss: scaling every gradient by c scales the first moment by c and the second by c^2, leaving the update unchanged. If the b^p factor applied uniformly to the objective, Adam would cancel it. It does not apply uniformly. Each source enters the batch gradient weighted by its own b^p, and the mean reduction fixes those relative weights before the optimizer acts, so Adam rescales the mixture without changing its composition.

We repeat the synthetic loss-space comparison of Figures 3a and 3b, with Adam in place of SGD, keeping all other knobs identical. Adam normalizes each coordinate by the running root-mean-square of its own gradient. This cancels the *global* magnitude inflation of the scale-contaminated gradient, so scale-contaminated training no longer risks divergence and converges faster overall, but it does not change which source dominates training, and the step-0 per-source gradient ratio is identical to SGD.

Under the scale-contaminated loss the highest-scale source reaches nMSE < 0.1 within $\sim$10 steps (Figure 7), the next within $\sim$20, and the lowest within $\sim$105. The time-to-threshold gap is down to $\sim 10\times$ from the $\sim 50\times$ under SGD, but it remains clearly scale-ordered. Under the scale-invariant loss all three sources reach the threshold together in about 10 steps. The equal-scale and gradient-norm-matched controls (Figure 8), as under SGD, confirm the spread is not an artifact of a confounding factor. Together with the gradient-norm-matched control in the main text, this places the effect in the composition of the batch gradient rather than in its magnitude, which is what the learning rate, gradient clipping, and adaptive preconditioning all act on.

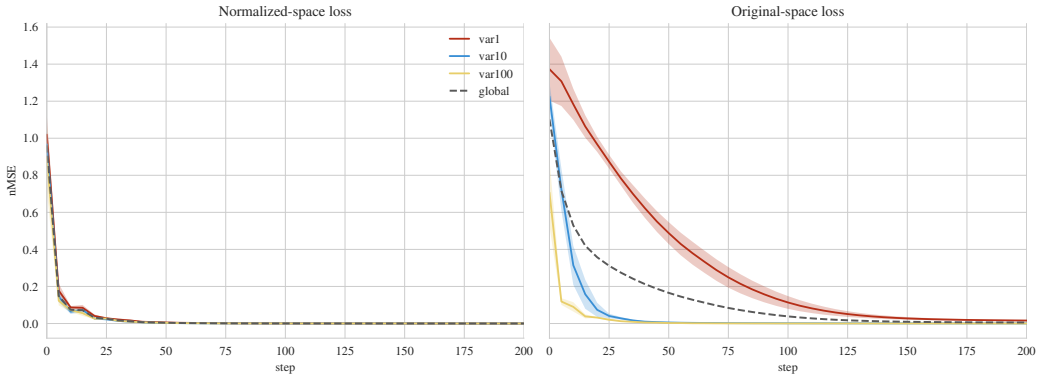


Figure 7: Adam, per-source nMSE over the first 200 steps (linear axis, mean over 5 seeds with ± 1 std bands; dashed grey = global nMSE). **Left:** under the scale-invariant loss all three sources converge together. **Right:** under the scale-contaminated loss the curves spread out in scale order (high-scale first, low-scale last), the same disparity SGD shows but compressed into the first $\sim$150 steps.

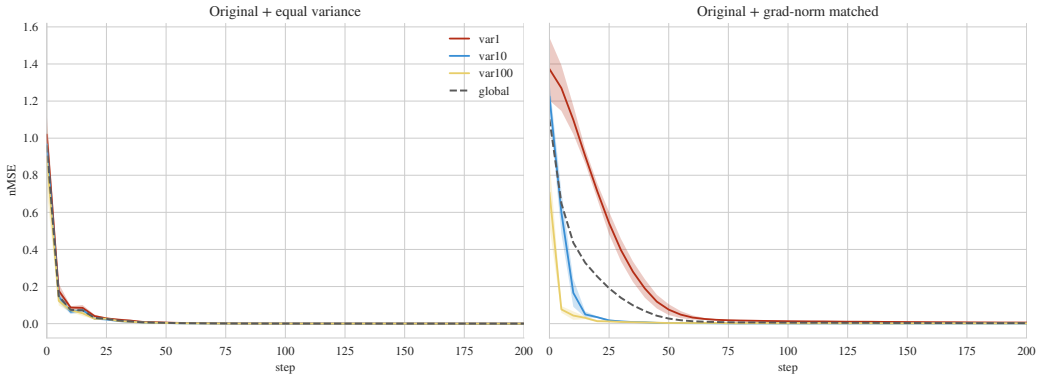
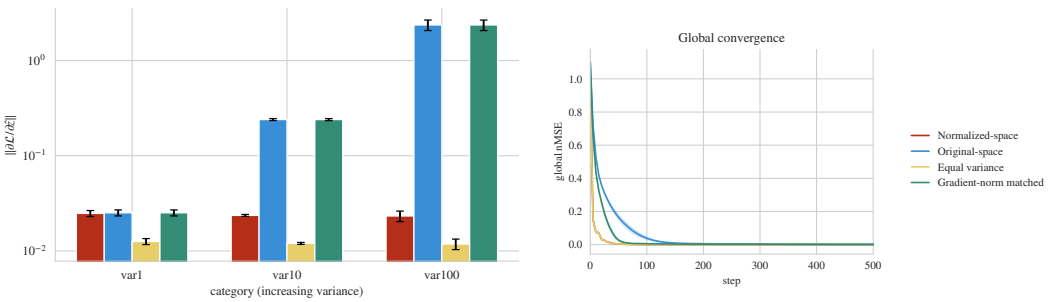


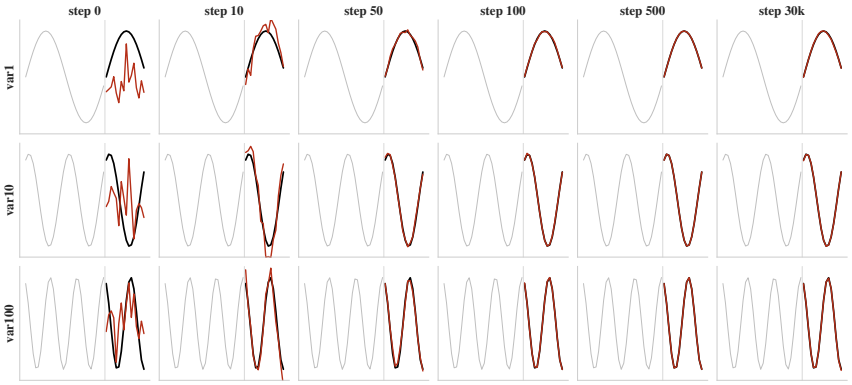
Figure 8: Adam control experiments (first 200 steps, linear axis, mean ± 1 std over 5 seeds). **Left:** on equal-scale data the scale-contaminated spread collapses to the scale-invariant baseline. **Right:** under gradient-norm-matched scale-contaminated training the scale-ordered spread persists.



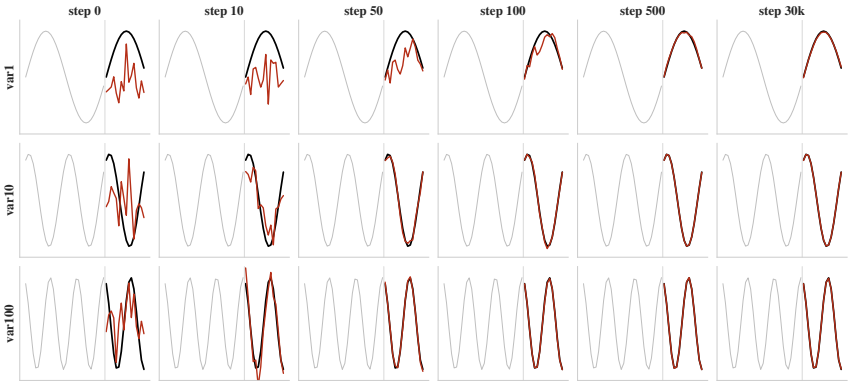
(a) Per-source gradient magnitude at initialization.

(b) Global nMSE (first 500 steps).

Figure 9: Adam, gradient magnitude and global convergence (mean ± 1 std over 5 seeds). The initialization gradient ratio is unchanged from SGD and scales as b^p under the scale-contaminated loss. Adam rescales coordinate magnitude but not the scale-driven ordering of convergence.



(a) Scale-invariant loss (SCALEIN)



(b) Scale-SCALECON loss

Figure 10: Adam, qualitative per-source forecasts at training steps 0, 10, 50, 100, 500, and 30,000.

F Scale-Swap Intervention Experiment

F.1 Reproducibility Details

Data. We use the 8 datasets listed in Section 4.2, represented as univariate series of length 128. We split across complete series, assigning a fixed random 80% of the series to training and the remaining 20% to validation. No series appears in both splits. We form stride-1 windows only after this split, using context length 64 and horizon 16 and removing windows with nonfinite or constant contexts. Each window is scaled to zero context mean and unit context standard deviation, then multiplied by its assigned $b \in \{1, 10\}$. Validation uses 512 fixed windows per dataset.

Assignments. Each configuration assigns 4 datasets to each scale. Its complement reverses all 8 assignments. We use 15 complementary pairs, giving 30 configurations in which every dataset appears 15 times at each scale. Each pair of data sets shares the larger scale in 6 or 7 configurations.

Model and optimization. The patch Transformer uses 8 nonoverlapping patches of length 8, embedding dimension 128, 4 encoder layers, 8 attention heads, feedforward width 256, dropout 0, and a direct linear head for the 16-step horizon. Scale each context using its sample mean and sample standard deviation with $\epsilon = 10^{-5}$. We train for 30,000 SGD updates with batch size 32 (4 windows per dataset), momentum 0, weight decay 0, and seeds $\{0, 1, 2, 3, 4\}$. Within each seed and assignment pair, conditions share the initialization and sampled batches.

Learning-rate control. Both conditions use constant learning rates. The scale-invariant condition uses 10^{-4}. Under scale-contaminated MSE, $b = 10$ multiplies a window's gradient by 100, while $b = 1$ leaves it unchanged. Because every batch contains equal numbers from the 2 scales, the nominal average multiplier is $(1 + 100)/2 = 50.5$. We therefore use $10^{-4}/50.5 = 1.9802 \times 10^{-6}$ for the scale-contaminated condition. This keeps the average update magnitude comparable without removing the relative $100:1$ weighting between scales.

Evaluation. We report validation nMSE on scaled targets. Evaluation occurs every 5 updates through step 200, every 50 through step 1,000, and every 1,000 thereafter. The trajectory summary and paired test are defined below.

F.2 Convergence Statistic and Test

Let $\tau_1 < \cdots < \tau_J$ be the training steps at which we evaluate held-out nMSE. For dataset $D^{(k)}$, scale b, complementary assignment pair c, seed s, and checkpoint j, write this error as $e^{(k)}_{b,c,s,j}$. We summarize the complete validation trajectory by integrating the straight lines between consecutive checkpoints and dividing by the training duration,

$$A^{(k)}_{b,c} = \frac{1}{S} \sum_{s=1}^{S} \frac{1}{\tau_J - \tau_1} \sum_{j=1}^{J-1} \frac{\tau_{j+1} - \tau_j}{2} \left(e^{(k)}_{b,c,s,j} + e^{(k)}_{b,c,s,j+1} \right). \tag{36}$$

This weighted average remains on the nMSE scale. It gives more weight to checkpoint intervals that cover more training steps. Because all runs share the same checkpoints, it compares their full trajectories on the same basis.

Each complementary assignment pair places every dataset at b=1 in one run and at b=10 in the other. For each dataset, we average over the $C = 15$ paired assignments,

$$\bar{A}^{(k)}_{b} = \frac{1}{C} \sum_{c=1}^{C} A^{(k)}_{b,c}, \qquad \Delta_k = \bar{A}^{(k)}_{1} - \bar{A}^{(k)}_{10}. \tag{37}$$

A positive Δ_k means dataset $D^{(k)}$ has lower validation error over training when assigned the larger scale. The aggregate effect is

$$\bar{\Delta} = \frac{1}{K} \sum_{k=1}^{K} \Delta_k. \tag{38}$$

The one-sided paired t-test uses the $K = 8$ dataset-level differences $d_k = \Delta_k$ as its statistical units. With sample mean $\bar{d}$ and sample standard deviation s_d, the test statistic is

$$t = \frac{\bar{d}}{s_d / \sqrt{K}}, \qquad s_d^2 = \frac{1}{K - 1} \sum_{k=1}^{K} (d_k - \bar{d})^2. \tag{39}$$

The null hypothesis is $\mathbb{E}[d_k] \leq 0$ and the one-sided alternative is $\mathbb{E}[d_k] > 0$. Under the scale-invariant loss, the mean trajectory-level errors across datasets are 1.964752 at b=1 and 1.964783 at b=10, a relative difference below 0.002%. For the scale-contaminated loss, the test gives t_7=2.013 and p=0.042. We compute the reported percentage from its means across datasets at each scale,

$$\bar{A}_b = \frac{1}{K}\sum_{k=1}^{K} \bar{A}_b^{(k)}, \qquad \bar{A}_1 = 2.5925, \qquad \bar{A}_{10} = 1.8990, \tag{40}$$

which gives

$$100 \times \frac{\bar{A}_1 - \bar{A}_{10}}{\bar{A}_1} = 26.8\%. \tag{41}$$

We repeat the trajectory and paired-effect calculations separately for the scale-invariant and scale-contaminated losses.

Figure 11 disaggregates the aggregate scale-swap trajectory in Figure 5a. The 2 scales overlap under the scale-invariant loss for every dataset. Under the scale-contaminated loss, the inflated copy converges faster in 7 of 8 datasets. Wind Farms is the only exception. These are the same dataset-level trajectories used for the paired analysis in Section 4.2.

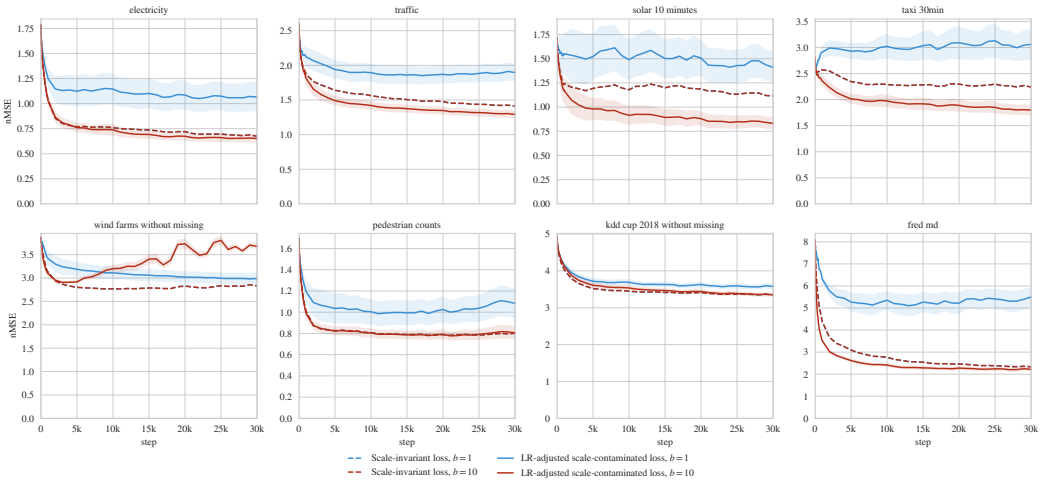


Figure 11: Dataset-level trajectories under scale-invariant loss and learning-rate-adjusted scale-contaminated loss. Under scale-invariant loss, the $b = 1$ and $b = 10$ trajectories overlap because the loss is computed on scaled targets. Lines and bands show the mean and 95% Student-t interval across 15 assignment contexts after averaging 5 seeds within each context.

G TSFM PRETRAINING DETAILS

G.1 REPRODUCIBILITY DETAILS

Corpus and windows. We pretrain on GiftEvalPretrain together with the training prefixes of M1, M3, M4, Tourism, and Favorita. All benchmark validation and test observations are excluded. The resulting corpus contains 73 datasets with usable windows. Before constructing windows, a fixed series-level split assigns 90% of the series in each dataset to training and 10% to validation, so no series appears in both splits. The maximum context and prediction lengths are 512 and 128, with minima of 64 and 8. For a series of length n, we set $H = \min(128, \max(8, \lfloor n/5 \rfloor))$ and $L = \min(512, n - H)$. If $L < 64$, we instead set $L = 64$ and $H = n - 64$. Series shorter than 72 observations are excluded. We use stride 512, retain windows with at least 90% observed values, and pad shorter windows to the maximum shape with masks.

Sampling. Each training example first samples a dataset uniformly, then a series uniformly within that dataset, and finally a window uniformly within that series. Thus, datasets and series with more available windows do not automatically receive more updates. The paired loss conditions use the same fixed sequence of sampled batches.

Models and objectives. All four architectures start from random weights. The two foundation models use their Small configurations. `Chronos2` Small uses patch size 16, hidden dimension 512, feed-forward dimension 2048, 6 layers, 8 attention heads, and dropout 0.1. It applies its native context standardization and arcsinh transform, then optimizes its native CRPS-style objective over 13 quantiles in $\{0.01, 0.05, 0.1, 0.2, \ldots, 0.9, 0.95, 0.99\}$. `Moirai-2.0` Small uses patch size 16, hidden dimension 384, feed-forward dimension 1024, 6 layers, and dropout 0.1. It applies its native context scaling and optimizes mean quantile loss over the 9 deciles from 0.1 to 0.9 on up to 64 target steps. `TimesFM2.5` is reduced to 49.6M parameters from the released 200M definition, keeping its head dimension of 80 and cutting width to 800 and depth to 12 layers. It keeps the input and output patch lengths of 32 and 128, so a single forward pass covers the 128-step horizon and no autoregressive decoding is needed. Its scaling is the per-patch expanding-window statistic that `TimesFM2.5` introduced, where patch i is scaled by the masked running mean and standard deviation of every observed point up to and including itself. We train the point head with MSE, as the released model does, and drop the separate 1024-step quantile head, which is unused at this horizon. `PatchTST` follows its original supervised configuration at 1.5M parameters, with patch length 16, stride 8, 3 encoder layers, 16 heads, hidden dimension 128, and dropout 0.2. It scales by RevIN with the affine parameters disabled, which is the setting its own released code defaults to, and optimizes MSE. Within each model, scale-invariant training computes the native objective on scaled targets, while scale-contaminated training reverses the model's scaling before computing the same objective. Everything else is held fixed. The four therefore differ in where their scaling statistics come from, which is what lets the loss-space comparison be read as independent of that choice.

Optimization and validation. We train for $80{,}000$ updates at batch size 512 with seeds $\{0, 1, 2, 3\}$. AdamW uses a constant learning rate of 10^{-4}, weight decay 0.01, and global gradient clipping at norm 1.0. Conditions within each model and seed share their initialization, batches, and optimizer settings. Every 250 updates, the training curves are evaluated on a fixed set of 64 validation windows per dataset.

Test splits and inference. M1, M3, and Tourism reserve the final official horizon, while M4 uses the released train/test files. Favorita uses the final 16 days of series ending on August 15, 2017, with missing sales filled by zero. For each GIFT-Eval short-term configuration, we use $W = \min(20, \max(1, \lceil 0.1 L_{\min}/H \rceil))$ non-overlapping windows of horizon H from each series' tail, where $L_{\min}$ is the shortest series length in that configuration. Its M4 configurations use $W = 1$, and multivariate series are evaluated channel-wise. Each origin uses only preceding observations, including observations from earlier windows when available. Models receive the most recent 512 observations, with shorter histories left-padded and masked, and their outputs are truncated to H without autoregressive rollout or benchmark-specific fine-tuning.

Metric aggregation. Within each subset, MASE is averaged over valid forecast cases using each case's full preceding history for the seasonal-naive denominator. WQL pools its numerator and denominator across cases before division, including the factor of two and averaging over supplied quantile levels. Missing targets and cases with undefined metric denominators are excluded from the corresponding metric. Benchmark scores average subset scores weighted by their total forecast-case

counts, then average over four seeds. Thus, benchmark WQL is a weighted average of subset ratios. The overlapping supervised test origins in Appendix H constitute a separate evaluation protocol.

Compute. All runs were performed with a single NVIDIA H200 GPU, where each 80k-step run took between 3.6 and 6.5 GPU-hours depending on the architecture. Generating Table 1 used 166 GPU-hours in total.

G.2 Cross-Dataset Dispersion and Pooled Accuracy During Training

Figure 12 complements the median MASE trajectory in Figure 5b for all four architectures. For each seed and checkpoint, it computes the variance across per-dataset MASE values before averaging across the 4 seeds. Scale-invariant training clearly reduces dispersion for `Chronos2` and modestly reduces it for `Moirai-2.0`, while the two losses converge to similar dispersion for `TimesFM2.5` and `PatchTST`. Low dispersion alone does not imply accurate forecasts, so Figure 13 also reports MASE pooled over all usable validation forecast cases in their natural dataset proportions. Scale-invariant training achieves lower pooled MASE throughout training for every architecture, including the two architectures whose final cross-dataset variance is similar across losses.

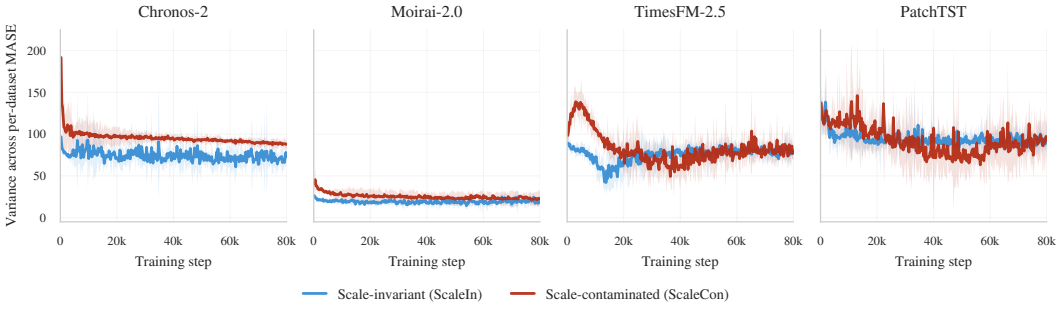


Figure 12: Cross-dataset variance of per-dataset MASE over 80,000 updates for all four architectures. Lines show the mean across 4 seeds and bands show 95% Student-t intervals. Lower variance indicates more uniform accuracy across datasets, but does not by itself imply lower forecast error.

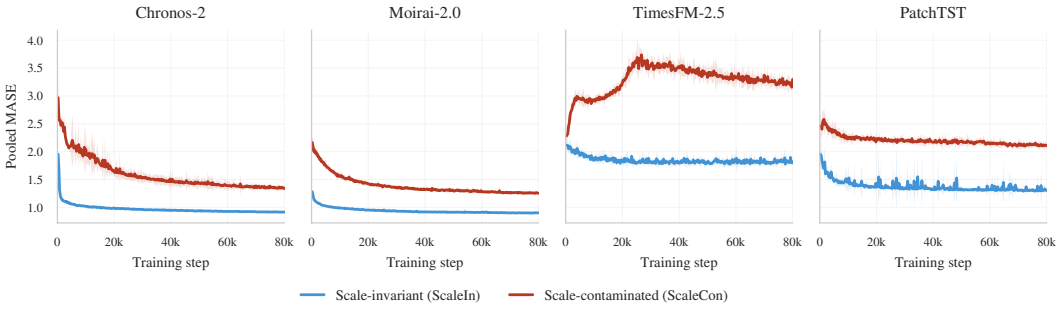


Figure 13: Pooled validation MASE over 80,000 updates for all four architectures. The pooled mean weights usable forecast cases in their natural dataset proportions, unlike the dataset-balanced median in Figure 5b. Lines show the mean across 4 seeds and bands show 95% Student-t intervals. Lower is better.

G.3 BENCHMARK-FREQUENCY RESULTS

Tables 7 and 8 give the full breakdown behind Table 1. GIFT-Eval reports some frequencies with multipliers or anchors. We pool `A-DEC` into Y, `Q-DEC` into Q, `10S` into S, `5T`, `10T`, and `15T` into T, and the 5 anchored weekly codes into W. Stars mark these pooled rows. The 6 GIFT-Eval rows derived from M4 duplicate the official M4 evaluation, so the paper does not average the two benchmarks together.

Table 7: In-domain MASE by benchmark and frequency after 80,000 updates, means with sample standard deviations (±, below) across 4 seeds. A star marks a base frequency that pools anchored or multiplied GIFT-Eval aliases. **SCALEIN** uses scale-invariant loss and **SCALECON** uses scale-contaminated loss. These are the only differences between the two runs of each architecture. Δ% reports **SCALEIN** relative to **SCALECON**. Green denotes an improvement and red a regression. The better loss is bold within each architecture. References are fitted on the series they forecast and appear behind the double rule.

		Chronos2			Moirai-2.0			TimesFM2.5			PatchTST			Reference
Benchmark	**Freq.**	**SC.IN**	**SC.CON**	Δ%	**SC.IN**	**SC.CON**	Δ%	**SC.IN**	**SC.CON**	Δ%	**SC.IN**	**SC.CON**	Δ%	S.Naive
M1	Y	**4.124** ±0.445	6.492 ±0.450	-36.5	**4.894** ±0.150	5.175 ±0.406	-5.4	8.067 ±0.509	**7.765** ±0.291	+3.9	**4.707** ±0.104	7.600 ±0.823	-38.1	4.894
	Q	**1.827** ±0.056	1.997 ±0.035	-8.5	**1.814** ±0.056	1.885 ±0.053	-3.8	**1.943** ±0.042	2.394 ±0.088	-18.8	**1.963** ±0.078	3.625 ±0.917	-45.9	2.078
	M	**1.238** ±0.019	1.312 ±0.013	-5.6	**1.234** ±0.011	1.306 ±0.024	-5.5	**1.425** ±0.018	1.579 ±0.015	-9.8	**1.366** ±0.018	2.838 ±1.050	-51.9	1.314
M3	Y	**3.070** ±0.045	4.085 ±0.241	-24.8	**3.119** ±0.127	3.643 ±0.171	-14.4	**4.880** ±0.198	5.309 ±0.188	-8.1	**3.889** ±0.465	9.529 ±2.353	-59.2	3.154
	Q	**1.258** ±0.060	1.461 ±0.072	-13.9	**1.255** ±0.039	1.295 ±0.077	-3.1	**1.476** ±0.016	1.846 ±0.110	-20.1	**1.590** ±0.142	3.867 ±1.043	-58.9	1.425
	M	**0.955** ±0.018	1.012 ±0.009	-5.6	**0.956** ±0.014	1.011 ±0.020	-5.4	**1.116** ±0.014	1.389 ±0.024	-19.6	**1.220** ±0.007	2.666 ±0.799	-54.2	1.146
M4	Y	**3.537** ±0.163	4.607 ±0.401	-23.2	**3.672** ±0.166	3.899 ±0.150	-5.8	**6.294** ±0.350	7.422 ±0.432	-15.2	**4.044** ±0.056	7.474 ±0.569	-45.9	3.974
	Q	**1.327** ±0.040	1.553 ±0.038	-14.6	**1.333** ±0.024	1.360 ±0.027	-2.0	**1.559** ±0.029	1.956 ±0.015	-20.3	**1.603** ±0.047	3.039 ±0.697	-47.2	1.602
	M	**1.123** ±0.035	1.248 ±0.042	-10.0	**1.131** ±0.028	1.160 ±0.008	-2.5	**1.294** ±0.024	1.545 ±0.016	-16.3	**1.378** ±0.065	3.299 ±0.835	-58.2	1.260
	W	**2.793** ±0.210	4.823 ±0.694	-42.1	**2.615** ±0.073	3.516 ±0.149	-25.6	**3.906** ±0.385	5.805 ±0.083	-32.7	**5.135** ±1.455	8.625 ±0.881	-40.5	2.777
	D	**3.765** ±0.114	5.502 ±0.406	-31.6	**3.656** ±0.106	4.354 ±0.243	-16.0	**4.451** ±0.236	5.743 ±0.029	-22.5	**5.375** ±0.745	11.379 ±1.590	-52.8	3.278
	H	**1.166** ±0.043	3.175 ±0.698	-63.3	**1.784** ±0.076	2.302 ±0.098	-22.5	**3.894** ±1.002	6.283 ±0.832	-38.0	3.066 ±0.253	**2.786** ±0.179	+10.0	1.193
Tourism	Y	**3.104** ±0.139	3.654 ±0.367	-15.0	**3.019** ±0.050	3.173 ±0.117	-4.8	**5.873** ±0.805	6.250 ±0.512	-6.0	**3.005** ±0.062	3.844 ±0.126	-21.8	3.552
	Q	**1.868** ±0.015	2.269 ±0.067	-17.7	**1.875** ±0.022	2.067 ±0.042	-9.3	**2.363** ±0.065	2.403 ±0.032	-1.7	**2.182** ±0.084	2.653 ±0.161	-17.8	1.699
	M	**1.845** ±0.030	2.269 ±0.015	-18.7	**1.834** ±0.038	2.284 ±0.039	-19.7	**2.616** ±0.149	3.068 ±0.138	-14.7	**2.189** ±0.056	2.650 ±0.084	-17.4	1.631
Favorita	D	**0.816** ±0.004	0.862 ±0.011	-5.4	**0.825** ±0.001	0.864 ±0.007	-4.5	**1.050** ±0.015	1.360 ±0.010	-22.8	**0.889** ±0.005	0.968 ±0.026	-8.2	1.079
GIFT-Eval	Y*	**3.530** ±0.162	4.593 ±0.402	-23.2	**3.663** ±0.165	3.888 ±0.150	-5.8	**6.278** ±0.350	7.386 ±0.435	-15.0	**4.028** ±0.057	7.451 ±0.569	-45.9	3.966
	Q*	**1.327** ±0.040	1.553 ±0.038	-14.6	**1.333** ±0.024	1.360 ±0.027	-2.0	**1.559** ±0.029	1.956 ±0.015	-20.3	**1.603** ±0.047	3.039 ±0.697	-47.2	1.602
	M	**1.124** ±0.034	1.233 ±0.038	-8.9	**1.119** ±0.025	1.141 ±0.008	-1.9	**1.288** ±0.021	1.531 ±0.015	-15.9	**1.357** ±0.060	3.150 ±0.782	-56.9	1.249
	W*	**1.861** ±0.056	2.332 ±0.168	-20.2	**1.884** ±0.057	2.032 ±0.034	-7.3	**2.217** ±0.051	2.752 ±0.070	-19.5	**2.449** ±0.313	3.920 ±0.384	-37.5	1.916
	D	**1.572** ±0.012	1.798 ±0.021	-12.6	**1.598** ±0.013	1.763 ±0.016	-9.3	**2.299** ±0.021	2.870 ±0.023	-19.9	**2.051** ±0.017	2.497 ±0.108	-17.8	2.153
	H	**1.393** ±0.025	1.650 ±0.038	-15.6	**1.456** ±0.014	1.677 ±0.008	-13.2	**1.895** ±0.022	2.571 ±0.084	-26.3	**1.640** ±0.015	1.857 ±0.057	-11.7	1.572
	T*	**1.129** ±0.022	1.402 ±0.015	-19.4	**1.239** ±0.059	1.355 ±0.009	-8.5	**1.650** ±0.039	2.103 ±0.023	-21.6	**1.610** ±0.032	1.952 ±0.071	-17.5	1.320
	S*	**2.241** ±0.337	3.568 ±0.443	-37.2	**2.090** ±0.199	2.595 ±0.135	-19.5	**2.466** ±0.139	3.807 ±0.121	-35.2	**3.808** ±0.223	4.733 ±0.262	-19.5	1.272

Table 8: In-domain WQL by benchmark and frequency after 80,000 updates, means with sample standard deviations (±, below) across 4 seeds. A star marks a base frequency that pools anchored or multiplied GIFT-Eval aliases. **ScaleIn** uses scale-invariant loss and **ScaleCon** uses scale-contaminated loss. These are the only differences between the two runs of each architecture. Δ% reports **ScaleIn** relative to **ScaleCon**. Green denotes an improvement and red a regression. The better loss is bold within each architecture. References are fitted on the series they forecast and appear behind the double rule.

		Chronos2			Moirai-2.0			TimesFM2.5			PatchTST			Reference
Benchmark	**Freq.**	**Sc.In**	**Sc.Con**	Δ%	**Sc.In**	**Sc.Con**	Δ%	**Sc.In**	**Sc.Con**	Δ%	**Sc.In**	**Sc.Con**	Δ%	S.Naive
M1	Y	**0.110** ±0.006	0.126 ±0.018	-12.8	**0.154** ±0.007	0.157 ±0.006	-2.1	**0.248** ±0.009	0.287 ±0.018	-13.6	**0.196** ±0.009	0.307 ±0.036	-36.1	0.184
	Q	**0.065** ±0.008	0.076 ±0.007	-13.7	**0.094** ±0.004	0.095 ±0.008	-1.2	**0.115** ±0.013	0.142 ±0.008	-18.9	**0.118** ±0.009	0.148 ±0.024	-20.2	0.117
	M	0.162 ±0.016	**0.131** ±0.007	+23.5	0.207 ±0.024	**0.188** ±0.021	+10.0	**0.227** ±0.006	0.261 ±0.021	-12.9	**0.272** ±0.015	0.322 ±0.066	-15.7	0.150
M3	Y	**0.101** ±0.001	0.112 ±0.006	-10.1	**0.132** ±0.004	0.143 ±0.004	-7.3	**0.218** ±0.009	0.221 ±0.006	-1.3	**0.170** ±0.010	0.283 ±0.039	-40.0	0.118
	Q	**0.063** ±0.003	0.066 ±0.002	-4.8	**0.079** ±0.003	0.080 ±0.003	-0.5	**0.104** ±0.001	0.126 ±0.007	-17.7	**0.110** ±0.006	0.202 ±0.034	-45.4	0.082
	M	**0.085** ±0.001	0.086 ±0.002	-0.5	**0.110** ±0.002	0.115 ±0.002	-3.8	**0.151** ±0.002	0.177 ±0.003	-14.6	**0.167** ±0.005	0.262 ±0.055	-36.4	0.121
M4	Y	**0.101** ±0.003	0.119 ±0.005	-15.2	0.136 ±0.004	**0.134** ±0.004	+1.5	**0.256** ±0.013	0.287 ±0.015	-11.0	**0.166** ±0.001	0.258 ±0.005	-35.5	0.137
	Q	**0.067** ±0.002	0.074 ±0.002	-8.3	0.089 ±0.002	**0.088** ±0.001	+1.1	**0.119** ±0.001	0.141 ±0.001	-15.2	**0.119** ±0.003	0.175 ±0.021	-32.0	0.098
	M	0.090 ±0.004	**0.088** ±0.001	+2.4	0.117 ±0.004	**0.111** ±0.002	+5.0	**0.149** ±0.001	0.167 ±0.001	-10.3	**0.162** ±0.009	0.258 ±0.039	-37.4	0.122
	W	**0.041** ±0.000	0.055 ±0.004	-25.1	**0.054** ±0.001	0.061 ±0.003	-11.7	**0.078** ±0.004	0.100 ±0.001	-21.9	**0.089** ±0.013	0.125 ±0.007	-29.0	0.061
	D	**0.020** ±0.001	0.029 ±0.002	-32.4	**0.025** ±0.001	0.030 ±0.002	-17.5	**0.037** ±0.002	0.048 ±0.000	-22.5	**0.044** ±0.006	0.079 ±0.007	-43.9	0.024
	H	**0.019** ±0.001	0.023 ±0.001	-17.5	**0.026** ±0.001	0.029 ±0.004	-11.0	**0.057** ±0.010	0.076 ±0.009	-24.4	**0.045** ±0.007	0.048 ±0.003	-6.7	0.038
Tourism	Y	**0.115** ±0.010	0.151 ±0.014	-23.8	**0.153** ±0.005	0.171 ±0.010	-10.4	**0.332** ±0.029	0.342 ±0.042	-3.0	**0.178** ±0.012	0.233 ±0.009	-23.6	0.170
	Q	**0.056** ±0.002	0.077 ±0.006	-26.3	**0.083** ±0.018	0.094 ±0.006	-11.4	**0.115** ±0.002	0.120 ±0.003	-4.0	**0.110** ±0.008	0.150 ±0.017	-27.1	0.098
	M	**0.077** ±0.004	0.100 ±0.003	-22.7	**0.102** ±0.003	0.121 ±0.003	-15.8	**0.208** ±0.018	0.330 ±0.041	-36.9	**0.153** ±0.007	0.211 ±0.009	-27.3	0.086
Favorita	D	**0.297** ±0.004	0.326 ±0.005	-9.1	**0.387** ±0.002	0.422 ±0.004	-8.3	**0.638** ±0.011	0.848 ±0.007	-24.8	**0.540** ±0.004	0.595 ±0.019	-9.3	0.614
GIFT-Eval	Y*	**0.101** ±0.003	0.119 ±0.005	-15.2	0.136 ±0.004	**0.134** ±0.004	+1.5	**0.256** ±0.013	0.287 ±0.015	-11.0	**0.166** ±0.001	0.258 ±0.005	-35.5	0.137
	Q*	**0.067** ±0.002	0.074 ±0.002	-8.3	0.089 ±0.002	**0.088** ±0.001	+1.1	**0.119** ±0.001	0.141 ±0.001	-15.2	**0.119** ±0.003	0.175 ±0.021	-32.0	0.098
	M	0.146 ±0.005	**0.132** ±0.002	+11.0	0.175 ±0.002	**0.160** ±0.001	+9.5	**0.227** ±0.001	0.251 ±0.002	-9.7	**0.229** ±0.007	0.317 ±0.037	-27.5	0.194
	W*	**0.125** ±0.006	0.141 ±0.004	-11.8	**0.163** ±0.004	0.172 ±0.003	-5.0	**0.253** ±0.004	0.290 ±0.009	-12.9	**0.257** ±0.007	0.331 ±0.022	-22.2	0.268
	D	**0.410** ±0.000	0.468 ±0.007	-12.5	**0.533** ±0.002	0.592 ±0.003	-10.0	**0.888** ±0.006	1.088 ±0.008	-18.4	**0.771** ±0.003	0.847 ±0.028	-9.0	1.182
	H	**0.247** ±0.014	0.275 ±0.006	-10.1	**0.339** ±0.012	0.360 ±0.011	-5.8	**0.496** ±0.017	0.657 ±0.029	-24.5	**0.426** ±0.011	0.461 ±0.012	-7.7	0.425
	T*	**0.346** ±0.018	0.398 ±0.009	-13.0	**0.454** ±0.010	0.481 ±0.005	-5.5	**0.633** ±0.014	0.718 ±0.012	-11.8	**0.632** ±0.015	0.732 ±0.026	-13.7	0.868
	S*	**0.028** ±0.005	0.039 ±0.006	-28.9	**0.034** ±0.005	0.034 ±0.004	-1.0	**0.035** ±0.002	0.069 ±0.004	-49.9	**0.064** ±0.005	0.076 ±0.004	-16.4	0.040

H SUPERVISED FORECASTING DETAILS

Data and pooling. The experiment pools eligible M1, M3, M4, and Tourism series by frequency. One model is trained for each of the yearly, quarterly, monthly, weekly, daily, and hourly pools. M1, M3, and Tourism contribute only their available yearly, quarterly, and monthly series, while M4 contributes all six frequencies. The model horizon is the largest official horizon in its pool. The input context is twice that horizon. Table 9 reports the resulting dimensions and eligible population.

Table 9: Frequency-specific dimensions and eligible training populations in the supervised experiment. The population is fixed before comparing loss spaces and is shared by the neural and statistical methods.

Frequency	Horizon	Context	Eligible series
Yearly	8	16	7,841
Quarterly	8	16	23,239
Monthly	24	48	34,183
Weekly	13	26	359
Daily	14	28	4,227
Hourly	48	96	414

For a series with official horizon H, the final $2H - 1$ observations form the rolling test region. The immediately preceding model-horizon block is used for validation. A series is eligible only when a complete input context and training horizon remain before validation. Evaluation uses all H rolling origins in the test region. At origin j, only observations through the end of that origin's input context are available to the model, scaler, and fitted references.

Loss-space comparison. Both conditions receive inputs under standard scaling whose offset a and denominator b are estimated from the input context. The scale-invariant loss (SCALEIN) computes MAE between predictions and targets after this scaling. The scale-contaminated loss (SCALECON) applies the same fixed inverse transform before computing MAE in original scale. The scaler has no learnable affine parameters in either condition, so matched runs differ only in the space where MAE is evaluated.

Optimization and checkpointing. Every model trains from scratch for exactly 20,000 optimizer updates. Training does not stop early. Validation is evaluated every 250 updates, and the best checkpoint is the state with the lowest validation MAE in original scale. All results average seeds 0, 1, 2, and 3. Table 10 gives the shared settings.

Table 10: Shared supervised training and evaluation settings.

Setting	Value
Optimizer	Adam
Training updates	20,000
Series batch size	32
Sampled windows per batch	512
Learning rate	10^{-3}
Learning-rate decays	3, evenly spaced
Validation interval	250 updates
Early stopping	Disabled
Training loss	MAE
Input scaler	Standard scaling from the input context
Start padding	Disabled
Forecast type	Point forecast
Random seeds	0, 1, 2, 3

We use NeuralForecast 3.2.1 (Olivares et al., 2022) and retain its architecture defaults except for the shared settings above, the frequency-specific horizon and context. The architecture parameters that materially determine model capacity are listed in Table 11.

Metrics and aggregation. MASE is computed for each series-origin forecast using the seasonal-naive in-sample MAE from all history available at that origin. WQL is the ratio of summed median

Table 11: Architecture-specific parameters for the supervised experiment. Parameters not shown use the NeuralForecast 3.2.1 defaults.

Architecture	Parameters
`NHITS`	Three identity stacks, one block per stack, two hidden layers of 512 units per block, pooling kernels $(2, 2, 1)$, frequency downsampling $(4, 2, 1)$, ReLU activation, and linear interpolation.
`NBEATS`	Identity, trend, and seasonality stacks, one block per stack, two hidden layers of 512 units per block, ReLU activation, and no shared block weights.
`PatchTST`	Three encoder layers, 16 attention heads, hidden width 128, feed-forward width 256, patch length 16, stride 8, dropout 0.2, GELU activation, and internal RevIN disabled.
`TCN`	Kernel size 2, dilations $(1, 2, 4, 8, 16)$, encoder width 128, context width 10, and a two-layer decoder of width 128.
`LSTM`	Two encoder layers of width 128, no encoder dropout, and a two-layer decoder of width 128. Forecasts use the direct decoder rather than recursive generation.

quantile loss to summed absolute targets within each benchmark-frequency pool. For each neural model, metrics are first computed within a seed. The main table macro-averages eligible frequencies within each benchmark and then averages the four seed values. Statistical references have no seed dimension. `S.Naive`, `ETS` (Holt, 2004), and `ARIMA` (Hyndman and Khandakar, 2008; Hyndman et al., 2026) are refitted at every rolling origin using the same available history and exactly the same eligible series as the neural models. We use the baselines implementation available in StatsForecast 2.1.1 (Garza et al., 2022).

Compute. All runs were performed with a single NVIDIA GeForce RTX 2080 Ti GPU. Generating Table 2 used 51 GPU-hours in total.

Benchmark-frequency results. Tables 12 and 13 give the full breakdown underlying the benchmark aggregates in Table 2.

The frequency-level results support the same findings that are in the main paper and somewhat strengthen them. **ScaleIn** lowers MASE in 49 of 75 architecture-benchmark-frequency comparisons and WQL in 43 of 75. The effect is strongest at the lower frequencies. It wins 17 of 20 yearly and 15 of 20 quarterly comparisons on MASE, with corresponding WQL counts of 15 and 14. Monthly results split 8 to 12 for both metrics, while the weekly, daily, and hourly M4 pools are also mixed. The benchmark-level advantage is therefore broad across architectures but concentrated in the yearly and quarterly pools. The detailed tables also show why the statistical references remain important. `ETS` or `ARIMA` often leads on M3 and the higher-frequency M4 pools even when **ScaleIn** improves the matched neural model.

Table 12: Best-checkpoint MASE by benchmark and frequency in the supervised experiment. Neural values are mean ± sample standard deviation across four seeds. **ScaleIn** uses scale-invariant MAE and **ScaleCon** uses scale-contaminated MAE, both with the same input-window standard scaling. References are fitted at every rolling origin on the same eligible series. Lower is better, and bold marks the better loss within each matched pair.

Architecture	Benchmark	Frequency	ScaleIn	ScaleCon	S.Naive	ETS	ARIMA
N-HiTS	M1	Y	**2.368** ±0.051	2.461 ±0.171	3.046	2.476	2.632
		Q	1.859 ±0.004	**1.857** ±0.012	2.527	1.899	1.966
		M	1.251 ±0.012	**1.248** ±0.009	1.859	1.338	1.333
	M3	Y	2.972 ±0.346	**2.934** ±0.092	3.333	2.384	3.030
		Q	**1.121** ±0.006	1.127 ±0.014	1.457	1.115	1.130
		M	0.863 ±0.012	**0.860** ±0.009	1.243	0.932	0.901
	M4	Y	**3.264** ±0.036	3.324 ±0.083	4.557	3.250	4.754
		Q	**1.192** ±0.002	1.199 ±0.003	1.590	1.196	1.207
		M	**0.917** ±0.005	0.919 ±0.003	1.301	0.956	0.938
		W	2.477 ±0.051	**2.466** ±0.055	2.799	2.471	2.843
		D	**3.160** ±0.006	3.167 ±0.007	3.207	3.178	3.234
		H	**0.937** ±0.066	0.966 ±0.074	1.220	1.127	1.133
	Tourism	Y	**3.689** ±0.112	3.878 ±0.346	2.956	3.460	4.135
		Q	**1.689** ±0.010	1.735 ±0.047	1.924	1.701	1.750
		M	1.658 ±0.025	**1.647** ±0.005	1.817	1.615	1.623
N-BEATS	M1	Y	**2.356** ±0.030	2.386 ±0.084	3.046	2.476	2.632
		Q	**1.858** ±0.018	1.859 ±0.007	2.527	1.899	1.966
		M	**1.252** ±0.005	1.259 ±0.012	1.859	1.338	1.333
	M3	Y	**3.059** ±0.146	3.327 ±0.449	3.333	2.384	3.030
		Q	1.120 ±0.016	**1.118** ±0.010	1.457	1.115	1.130
		M	**0.853** ±0.006	0.856 ±0.004	1.243	0.932	0.901
	M4	Y	**3.246** ±0.034	3.448 ±0.300	4.557	3.250	4.754
		Q	**1.188** ±0.001	1.191 ±0.002	1.590	1.196	1.207
		M	**0.914** ±0.001	0.915 ±0.004	1.301	0.956	0.938
		W	**2.479** ±0.021	2.497 ±0.045	2.799	2.471	2.843
		D	**3.164** ±0.003	**3.164** ±0.007	3.207	3.178	3.234
		H	**0.933** ±0.015	1.001 ±0.046	1.220	1.127	1.133
	Tourism	Y	**3.580** ±0.116	3.847 ±0.073	2.956	3.460	4.135
		Q	1.704 ±0.007	**1.695** ±0.014	1.924	1.701	1.750
		M	1.637 ±0.023	**1.635** ±0.019	1.817	1.615	1.623
PatchTST	M1	Y	**2.167** ±0.104	2.194 ±0.115	3.046	2.476	2.632
		Q	2.056 ±0.079	**2.018** ±0.040	2.527	1.899	1.966
		M	1.527 ±0.025	**1.520** ±0.033	1.859	1.338	1.333
	M3	Y	**6.081** ±0.790	6.911 ±1.352	3.333	2.384	3.030
		Q	**1.337** ±0.021	1.351 ±0.030	1.457	1.115	1.130
		M	0.989 ±0.017	**0.988** ±0.018	1.243	0.932	0.901
	M4	Y	**3.807** ±0.242	3.867 ±0.282	4.557	3.250	4.754
		Q	1.352 ±0.015	**1.341** ±0.025	1.590	1.196	1.207
		M	1.029 ±0.019	**1.027** ±0.022	1.301	0.956	0.938
		W	**3.929** ±0.098	**3.929** ±0.214	2.799	2.471	2.843
		D	4.198 ±0.145	**4.152** ±0.189	3.207	3.178	3.234
		H	7.518 ±1.134	**7.118** ±2.043	1.220	1.127	1.133
	Tourism	Y	**3.283** ±0.500	3.487 ±0.483	2.956	3.460	4.135
		Q	**3.168** ±0.087	3.188 ±0.142	1.924	1.701	1.750
		M	**2.200** ±0.100	2.204 ±0.083	1.817	1.615	1.623
TCN	M1	Y	**2.467** ±0.127	2.606 ±0.123	3.046	2.476	2.632
		Q	**1.903** ±0.014	1.936 ±0.052	2.527	1.899	1.966
		M	1.452 ±0.013	**1.419** ±0.011	1.859	1.338	1.333
	M3	Y	**3.570** ±0.438	4.035 ±0.372	3.333	2.384	3.030
		Q	**1.160** ±0.029	1.202 ±0.042	1.457	1.115	1.130
		M	0.957 ±0.008	**0.943** ±0.007	1.243	0.932	0.901
	M4	Y	**3.636** ±0.147	3.751 ±0.151	4.557	3.250	4.754
		Q	**1.229** ±0.011	1.268 ±0.044	1.590	1.196	1.207
		M	1.003 ±0.008	**0.991** ±0.007	1.301	0.956	0.938
		W	**2.468** ±0.024	2.545 ±0.100	2.799	2.471	2.843
		D	3.171 ±0.006	**3.169** ±0.014	3.207	3.178	3.234
		H	**1.035** ±0.033	1.465 ±0.258	1.220	1.127	1.133
	Tourism	Y	**3.951** ±0.371	3.976 ±0.494	2.956	3.460	4.135
		Q	**1.963** ±0.127	2.062 ±0.219	1.924	1.701	1.750
		M	2.089 ±0.073	**2.026** ±0.069	1.817	1.615	1.623
LSTM	M1	Y	**2.477** ±0.044	2.607 ±0.124	3.046	2.476	2.632
		Q	**2.000** ±0.023	2.042 ±0.016	2.527	1.899	1.966
		M	**1.516** ±0.007	1.521 ±0.009	1.859	1.338	1.333
	M3	Y	4.113 ±0.157	**4.104** ±0.717	3.333	2.384	3.030
		Q	**1.235** ±0.046	1.246 ±0.031	1.457	1.115	1.130
		M	**1.050** ±0.004	1.053 ±0.005	1.243	0.932	0.901
	M4	Y	**3.877** ±0.149	3.909 ±0.168	4.557	3.250	4.754
		Q	**1.293** ±0.002	1.303 ±0.009	1.590	1.196	1.207
		M	**1.056** ±0.008	1.059 ±0.007	1.301	0.956	0.938
		W	**2.780** ±0.041	2.813 ±0.107	2.799	2.471	2.843
		D	**3.492** ±0.010	3.494 ±0.009	3.207	3.178	3.234
		H	1.087 ±0.074	**1.030** ±0.024	1.220	1.127	1.133
	Tourism	Y	4.238 ±0.531	**4.077** ±0.137	2.956	3.460	4.135
		Q	**1.864** ±0.009	1.939 ±0.056	1.924	1.701	1.750
		M	1.827 ±0.009	**1.826** ±0.015	1.817	1.615	1.623

Table 13: Best-checkpoint WQL by benchmark and frequency in the supervised experiment. Neural values are mean ± sample standard deviation across four seeds. **ScaleIn** uses scale-invariant MAE and **ScaleCon** uses scale-contaminated MAE, both with the same input-window standard scaling. References are fitted at every rolling origin on the same eligible series. Lower is better, and bold marks the better loss within each matched pair.

Architecture	Benchmark	Frequency	ScaleIn	ScaleCon	S.Naive	ETS	ARIMA
N-HiTS	M1	Y	0.119 ±0.003	**0.117** ±0.007	0.219	0.144	0.204
		Q	**0.096** ±0.002	**0.096** ±0.001	0.139	0.094	0.106
		M	0.083 ±0.001	**0.081** ±0.003	0.144	0.088	0.086
	M3	Y	**0.105** ±0.005	0.107 ±0.006	0.105	0.093	0.100
		Q	**0.081** ±0.001	0.082 ±0.001	0.092	0.079	0.081
		M	**0.083** ±0.001	**0.083** ±0.001	0.111	0.089	0.088
	M4	Y	**0.120** ±0.001	0.123 ±0.004	0.146	0.118	0.152
		Q	**0.099** ±0.000	**0.099** ±0.000	0.118	0.097	0.100
		M	**0.077** ±0.000	**0.077** ±0.000	0.102	0.080	0.080
		W	**0.061** ±0.002	**0.061** ±0.002	0.066	0.064	0.064
		D	**0.026** ±0.000	**0.026** ±0.000	0.027	0.026	0.027
		H	0.048 ±0.001	**0.046** ±0.001	0.039	0.042	0.050
	Tourism	Y	**0.146** ±0.004	0.156 ±0.016	0.125	0.138	0.168
		Q	**0.113** ±0.001	0.115 ±0.001	0.130	0.120	0.152
		M	0.140 ±0.002	**0.139** ±0.003	0.171	0.141	0.146
N-BEATS	M1	Y	0.120 ±0.005	**0.118** ±0.007	0.219	0.144	0.204
		Q	**0.093** ±0.002	0.094 ±0.002	0.139	0.094	0.106
		M	**0.082** ±0.001	0.083 ±0.002	0.144	0.088	0.086
	M3	Y	**0.104** ±0.003	0.113 ±0.009	0.105	0.093	0.100
		Q	**0.081** ±0.001	**0.081** ±0.001	0.092	0.079	0.081
		M	**0.082** ±0.000	**0.082** ±0.000	0.111	0.089	0.088
	M4	Y	**0.119** ±0.001	0.126 ±0.010	0.146	0.118	0.152
		Q	**0.098** ±0.000	**0.098** ±0.000	0.118	0.097	0.100
		M	**0.077** ±0.000	**0.077** ±0.000	0.102	0.080	0.080
		W	**0.062** ±0.002	**0.062** ±0.001	0.066	0.064	0.064
		D	**0.026** ±0.000	**0.026** ±0.000	0.027	0.026	0.027
		H	0.047 ±0.001	**0.046** ±0.002	0.039	0.042	0.050
	Tourism	Y	**0.142** ±0.005	0.153 ±0.003	0.125	0.138	0.168
		Q	**0.115** ±0.002	0.116 ±0.002	0.130	0.120	0.152
		M	**0.138** ±0.002	**0.138** ±0.002	0.171	0.141	0.146
PatchTST	M1	Y	**0.116** ±0.010	0.117 ±0.010	0.219	0.144	0.204
		Q	0.100 ±0.005	**0.097** ±0.001	0.139	0.094	0.106
		M	**0.103** ±0.002	0.104 ±0.005	0.144	0.088	0.086
	M3	Y	**0.165** ±0.015	0.185 ±0.031	0.105	0.093	0.100
		Q	**0.090** ±0.003	0.092 ±0.002	0.092	0.079	0.081
		M	**0.092** ±0.001	**0.092** ±0.001	0.111	0.089	0.088
	M4	Y	**0.139** ±0.006	0.143 ±0.012	0.146	0.118	0.152
		Q	**0.110** ±0.003	**0.110** ±0.003	0.118	0.097	0.100
		M	**0.084** ±0.001	**0.084** ±0.001	0.102	0.080	0.080
		W	**0.072** ±0.001	**0.072** ±0.002	0.066	0.064	0.064
		D	**0.034** ±0.001	**0.034** ±0.001	0.027	0.026	0.027
		H	0.120 ±0.015	**0.117** ±0.029	0.039	0.042	0.050
	Tourism	Y	**0.133** ±0.019	0.142 ±0.021	0.125	0.138	0.168
		Q	**0.173** ±0.004	0.174 ±0.005	0.130	0.120	0.152
		M	**0.186** ±0.011	**0.186** ±0.009	0.171	0.141	0.146
TCN	M1	Y	**0.111** ±0.015	0.117 ±0.010	0.219	0.144	0.204
		Q	0.094 ±0.003	**0.092** ±0.002	0.139	0.094	0.106
		M	0.096 ±0.003	**0.094** ±0.001	0.144	0.088	0.086
	M3	Y	**0.123** ±0.009	0.135 ±0.010	0.105	0.093	0.100
		Q	**0.083** ±0.002	0.085 ±0.002	0.092	0.079	0.081
		M	0.091 ±0.000	**0.089** ±0.000	0.111	0.089	0.088
	M4	Y	**0.133** ±0.005	0.137 ±0.005	0.146	0.118	0.152
		Q	**0.101** ±0.001	0.104 ±0.002	0.118	0.097	0.100
		M	0.084 ±0.000	**0.083** ±0.001	0.102	0.080	0.080
		W	**0.061** ±0.000	0.062 ±0.001	0.066	0.064	0.064
		D	**0.026** ±0.000	**0.026** ±0.000	0.027	0.026	0.027
		H	**0.043** ±0.001	0.044 ±0.001	0.039	0.042	0.050
	Tourism	Y	**0.159** ±0.014	0.161 ±0.023	0.125	0.138	0.168
		Q	**0.123** ±0.007	0.126 ±0.009	0.130	0.120	0.152
		M	0.176 ±0.009	**0.172** ±0.007	0.171	0.141	0.146
LSTM	M1	Y	**0.121** ±0.020	0.125 ±0.012	0.219	0.144	0.204
		Q	**0.098** ±0.002	0.100 ±0.001	0.139	0.094	0.106
		M	0.096 ±0.002	**0.095** ±0.001	0.144	0.088	0.086
	M3	Y	0.142 ±0.007	**0.136** ±0.014	0.105	0.093	0.100
		Q	**0.089** ±0.003	**0.089** ±0.003	0.092	0.079	0.081
		M	**0.098** ±0.000	**0.098** ±0.000	0.111	0.089	0.088
	M4	Y	0.140 ±0.005	**0.139** ±0.005	0.146	0.118	0.152
		Q	**0.106** ±0.000	**0.106** ±0.001	0.118	0.097	0.100
		M	**0.088** ±0.000	**0.088** ±0.000	0.102	0.080	0.080
		W	**0.064** ±0.001	**0.064** ±0.001	0.066	0.064	0.064
		D	**0.029** ±0.000	**0.029** ±0.000	0.027	0.026	0.027
		H	**0.056** ±0.005	0.057 ±0.003	0.039	0.042	0.050
	Tourism	Y	0.169 ±0.022	**0.161** ±0.006	0.125	0.138	0.168
		Q	**0.123** ±0.004	0.126 ±0.006	0.130	0.120	0.152
		M	**0.168** ±0.002	0.170 ±0.005	0.171	0.141	0.146

I MASKED RECONSTRUCTION WITH MOMENT

We also compare the two loss spaces for masked reconstruction task using `MOMENT` (Goswami et al., 2024). The matched runs use 512-point input, a 30% masking ratio, 60,000 steps, and a batch size of 512. The results are averaged across 4 seeds. Evaluation covers held-out series from GiftEvalPretrain datasets. We measure MAE on masked positions divided by the input window's seasonal-naive MAE (reconstruction MASE). After averaging each dataset across seeds, **SCALEIN** lowers error on all datasets and reduces median reconstruction MASE from 1.42 to 0.69 (51%). Figure 14 shows that the advantage is large and persists throughout training.

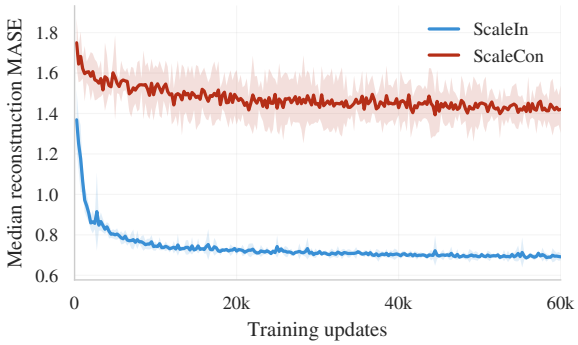


Figure 14: MOMENT reconstruction on held-out GiftEvalPretrain series. Plotted are average per-seed median dataset MASE, with 95% Student-t intervals across 4 seeds.