TSTAR: TEMPORAL STRUCTURE-AWARE TRAINING DATA CONSTRUCTION FOR TIME-SERIES MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series foundation models increasingly rely on large, heterogeneous pretraining datasets, yet how to construct these datasets has received far less systematic study than in natural language processing and computer vision. **Dataset scale and source diversity alone do not determine how training exposure is allocated across temporal structures**, so even a diverse corpus can leave some structures rarely seen. To address this, we introduce **Tstar**, a temporal structure-aware method that organizes training windows from heterogeneous datasets into **an interpretable atlas of temporal structures**. The atlas guides real-window selection and sampling, with optional targeted synthesis, to increase exposure to underrepresented patterns. We pretrain TinyTimeMixer and MOMENT-341M from random initialization under fixed budgets. For TinyTimeMixer, coverage-guided selection **reduces the six-seed means of four TIME and FEV error aggregates by 23.41%–41.81%** relative to matched uniform selection, with every paired seed improving. For MOMENT-341M, **real-window rebalancing alone reduces MASE by 3.43% on TIME and 7.67% on FEV** relative to uniform sampling, with both 95% bootstrap intervals excluding zero. A supporting study on six conventional architectures shows lower average error.

1 INTRODUCTION

The performance of large language models depends strongly on how their pretraining datasets are constructed, making data preparation a central part of model development (Penedo et al. 2024; Li et al. 2024). In natural language processing and computer vision, pretraining-data construction has become a systematic research topic encompassing data selection, preprocessing, synthetic-data generation, and sampling. DataComp and DataComp-LM provide controlled settings for comparing data-curation strategies (Gadre et al. 2023; Li et al. 2024), while Dolma and FineWeb investigate dataset construction, filtering, and deduplication (Soldaini et al. 2024; Penedo et al. 2024). Other studies examine synthetic textbook data for pretraining and optimize the proportions in which different data domains are sampled (Gunasekar et al. 2023; Xie et al. 2023). However, for time-series foundation models (TSFMs), **dataset construction is still commonly developed alongside individual models, and its systematic investigation as a separate research problem remains comparatively limited**, despite recent data-centric efforts (Goswami et al. 2024; Woo et al. 2024; Shao et al. 2025).

Existing TSFM pretraining datasets differ in their data sources, sampling policies, and synthetic generators. Recent TSFMs have emphasized large-scale pretraining on data collected from diverse domains, as exemplified by MOMENT's Time Series Pile and Moirai's LOTSA (Goswami et al. 2024; Woo et al. 2024). Moirai also adjusts sampling probabilities to limit the dominance of large constituent datasets (Woo et al. 2024). Chronos supplements real-world collections with Gaussian-process-generated series, while TimesFM combines real and synthetic data for pretraining (Ansari et al. 2024; Das et al. 2024). More recent data-centric work examines other aspects of corpus construction: BLAST balances sampling across groups defined by statistical features, and Mix, Don't Pick studies the composition of synthetic generator mixtures and their combination with real data (Shao et al. 2025; Nagpal et al. 2026). Together, these efforts have advanced corpus scale, source diversity, sampling, and synthetic-data composition. However, **increasing data volume and domain coverage does not by itself ensure broad coverage of temporal structures**. For example, windows from electricity and traffic data can both show repeated cycles, while different windows from the same time series can show gradual trends, abrupt changes, or occasional spikes. A large pretraining dataset

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.

can therefore contain many windows with similar temporal structures and few windows representing others; in our uniformly selected 18-source TTM dataset, a single archetype accounts for 45.7% of windows (Section 4.3). Under a fixed pretraining budget, repeatedly sampling common structures leaves fewer training opportunities for less represented ones, yet **how this temporal-structure composition affects downstream performance remains insufficiently understood**.

To investigate this question, we introduce **Tstar**, a temporal structure-aware method for constructing time-series pretraining datasets (Figure 1, right). Tstar **organizes training windows into an interpretable atlas of temporal structures** and characterizes their distribution across the dataset. The atlas guides which real windows enter a finite dataset (selection) and how often each window is presented during training (sampling); optional targeted synthesis replaces some scheduled presentations with windows generated for underrepresented structures. Every compared policy keeps the model architecture, number of training updates, and batch size unchanged, so **the comparisons change the pretraining data rather than the training budget**.

We pretrain MOMENT-341M and TinyTimeMixer from random initialization under matched training budgets to test whether changing the pretraining data improves forecasting. A supporting study across six conventional architectures (NLinear, DLinear, TCN, GRU, PatchTST, and Transformer) examines the effects of the data-construction method on supervised forecasting and imputation.

Our contributions are:

1. **Temporal structure-aware data construction.** We introduce Tstar, a method that uses an interpretable atlas of temporal structures to connect dataset characterization with pretraining-data construction. The same structural representation guides real-data selection, sampling, and optional targeted synthesis, making the composition being adjusted explicit.
2. **Empirical evidence on real-data composition.** We show that adjusting the selection and sampling of existing real data can improve forecasting in from-scratch pretraining without expanding the source pool or increasing the pretraining budget. Complementary studies of real-window selection and sampling demonstrate benefits beyond enlarging the corpus; in the TTM study, gains occur even with matched dataset size, source quotas, and coverage of all atlas categories.

2 Related Work

As summarized in Figure 1 (left), MOMENT's Time Series Pile and Moirai's LOTSA assemble time series from diverse domains for pretraining (Goswami et al. 2024; Woo et al. 2024). Chronos and TimesFM combine real and synthetic time series in their pretraining datasets (Ansari et al. 2024; Das et al. 2024). Across eleven synthetic generator families, Mix, Don't Pick finds that generator effectiveness depends on the model architecture and that mixing generators can improve forecasting performance (Nagpal et al. 2026). Tabby releases an open pretraining recipe combining curated real datasets with two synthetic generators (Xie et al. 2026). These efforts decide which sources and generators enter a corpus, but not how training exposure is distributed across temporal structures. BLAST balances sampling across grids in a statistical feature space computed for whole series and generates additional data by mixing existing samples (Shao et al. 2025). Its groups are grids in a reduced feature space, without explicit temporal-structure labels. Tstar differs in granularity, representation, and use. It profiles each lookback window, the input segment the model sees, rather than the whole series, so structures that change within a series can be assigned to different atlas categories. It organizes these windows by explicitly defined temporal structures rather than feature-space grids, so the composition being adjusted can be inspected and reported. The same atlas then drives dataset characterization, real-window selection, sampling during pretraining, and synthesis of examples for selected structures, all under a fixed number of training presentations.

3 Method

In plain terms, Tstar works in three steps (Figure 1, right, steps 1–5). It first cuts every source series into *windows*, the fixed-length input segments a model sees during pretraining. It then describes each window by a *fingerprint*, a vector of simple statistics such as trend slope, dominant frequency, and spike rate, and uses fixed rules to assign the window to one of 18 named temporal patterns, or *archetypes*, such as smooth trend, strong seasonality, or intermittent pulses. The archetypes, their optional finer categories, and their window counts form the *atlas*. Finally, it uses the atlas to decide

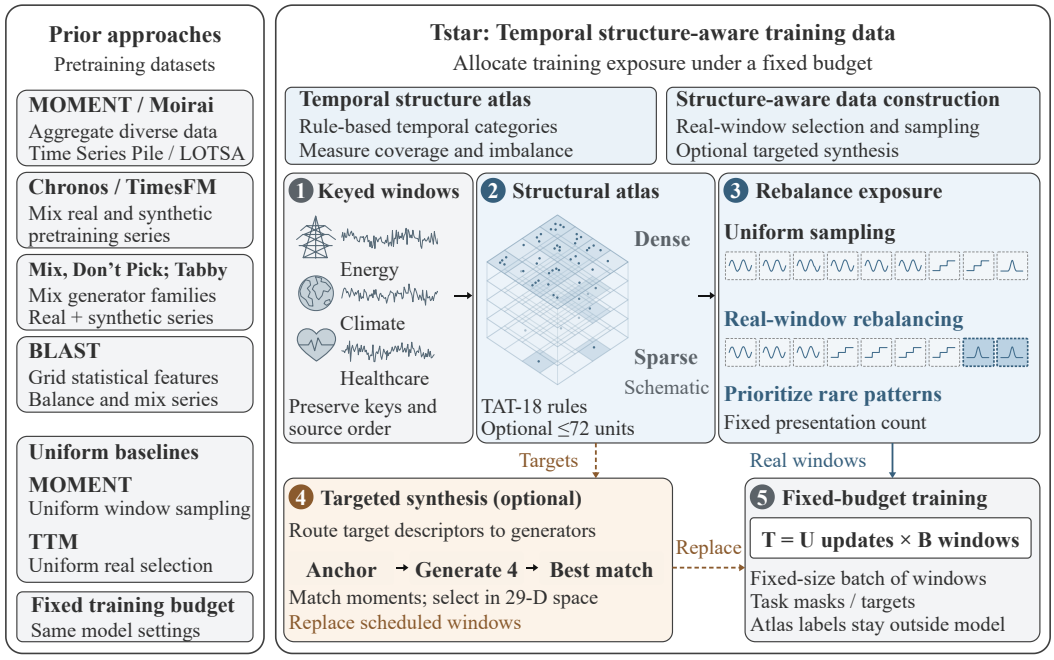


Figure 1: **Tstar rebalances exposure by temporal structure.** Source diversity alone does not ensure balanced structural coverage. Provenance-tracked windows are profiled through temporal fingerprints to build an interpretable atlas, which guides real-window sampling weights. The two rows illustrate alternative policies under the same number of training presentations: uniform window sampling versus structure-aware rebalancing. The highlighted event windows illustrate increased exposure to underrepresented patterns; the compositions and atlas point counts are schematic. Targeted synthetic replacement is optional and does not add presentations. Matched experiments find lower forecasting error with the same learner settings and fixed training budgets. Fingerprints and atlas labels are construction metadata, not model inputs.

which windows the model sees and how often. A *presentation* is one window shown to the model in one training step, and a data *policy* divides a fixed number of presentations among windows, giving underrepresented archetypes more presentations and optionally replacing some real windows with synthetic ones. The model receives only the numerical windows with task-specific masks and targets; fingerprints and archetype labels are used only to construct the pretraining data.

3.1 Problem definition and training interface

Let $\mathcal{D}$ be a collection of heterogeneous time-series sources. Each trajectory $X_\kappa = (x_{\kappa,0}, \ldots, x_{\kappa,N_\kappa-1})$ contains N_κ observations and has key $\kappa = (\text{source}, \text{dataset}, \text{entity or proxy}, \text{variable})$; a proxy is a documented sequence identifier used when no entity identifier is available. At an admissible start index t, a window of at most L observations is $w_{\kappa,t} = (x_{\kappa,t}, \ldots, x_{\kappa,\min(t+L,N_\kappa)-1})$. The candidate inventory is $\mathcal{W}(\mathcal{D}) = \bigcup_\kappa \mathcal{W}_\kappa$, where $\mathcal{W}_\kappa$ contains windows from X_κ admitted by the extraction protocol. Windows preserve trajectory boundaries and the sequence order supplied by each source adapter.

With U successful updates and global batch size B, policy π allocates $T = UB$ window presentations between real and synthetic data according to

$$Q_\pi = (1 - r_\pi)Q_{\pi,\text{real}} + r_\pi Q_{\pi,\text{syn}}. \tag{1}$$

Here $Q_{\pi,\text{real}}$ and $Q_{\pi,\text{syn}}$ are distributions over real and targeted synthetic windows, respectively, and $r_\pi \in [0, 1]$ is the synthetic presentation fraction enforced by a fixed real/synthetic schedule. For each paired comparison, we fix the architecture, initialization, and optimization settings and measure task error on the same held-out data. When a fixed-size dataset is needed, as for TTM, the finite-dataset interface exports $\mathcal{S}^\pi_n = \mathcal{S}^\pi_{\text{real}} \cup \mathcal{S}^\pi_{\text{repair}}$ for requested size n, where repair windows are synthetic windows added only for categories that real windows leave empty, with a target cap η on

their share. *Coverage-guided selection* chooses real windows by prioritizing coverage targets and using representativeness, local density, and diversity in fingerprint space (Appendix B.2).

3.2 Building a structural atlas

The fingerprint $z(w)$ measures support, distribution, trend, dependence, frequency structure, events, and observation quality (Appendix Table 8). We compute fingerprints for a capped random sample of windows, the profile $\mathcal{P} \subseteq \mathcal{W}(\mathcal{D})$ and use it to construct a sampling policy for the loader crops $\mathcal{R}$, the overlapping training segments the data loader draws from. We form $\phi(w)$ from 31 fingerprint coordinates, including standard deviation, range, and slope, using robust scaling fitted on $\mathcal{P}$ and clipping to $[-12, 12]$. The TAT-18 map g_{18} assigns each window to one of 18 temporal archetypes using ordered, first-match rules. Optional MiniBatchKMeans clustering within each archetype splits it into finer temporal-structure categories:

$$a(w) = g_{18}(z(w)), \qquad c(w) = (a(w), b_{a(w)}(\phi(w))), \qquad K = \sum_{a \in \mathcal{A}_{\text{obs}}} k_a \leq 72, \tag{2}$$

where $\mathcal{A}_{\text{obs}}$ is the observed archetype set, b_a the within-archetype cluster assignment, and $k_a \in \{1, 2, 3, 4\}$ its retained cluster count, so K is the number of finer categories. Splits are retained according to cluster separation, agreement between two fits, and minimum cluster size (Appendix A).

Algorithm 1 Tstar fixed-budget training presentations

Input: source assets $\mathcal{D}$, policy π, updates U, batch size B, synthetic share r_π
Output: trained learner and provenance log

```
Compile within-key windows W(D) and loader crops R
Measure profile P; build or load sampling weights and the optional generator catalog
Initialize the provenance log; record the profile-fitting and data-split order
for successful update s = 0, ..., U − 1 do
    Initialize an empty batch
    for batch position j = 1, ..., B do
        Draw a real window from R using the sampling weights
        if presentation sB + j is scheduled as synthetic then
            Replace it using the generator catalog and Section 3.4
        end if
        Append the window to the batch; log its real origin or synthetic target and generator choice
    end for
    Update the learner using the batch and its ordinary task masks/targets
end for
```

3.3 Turning structural imbalance into real-sampling weights

We define two real-sampling policies. The broad policy corrects frequency imbalance directly: each profile window's weight is the product of inverse-square-root frequency factors for its TAT-18 archetype and its observation regime, followed by clipping and normalization. The refined policy operates on the K finer categories and adds within-structure terms. For profile windows $w_i \in \mathcal{P}$, let $p_c = |\mathcal{P}|^{-1} \sum_{w_i \in \mathcal{P}} \mathbf{1}[c(w_i) = c]$ be the fraction of profile windows in structure c, and $d_c = \max(0, K^{-1} - p_c)/K^{-1}$ its relative shortfall from the uniform reference $1/K$. The complexity proxy h_i averages six fingerprint scores, such as change score and spike ratio, scaled and clipped to $[0, 1]$; v_i, the window's centroid distance in ϕ-space normalized within its structure and clipped to $[0, 1]$, measures how atypical the window is within its category. The refined policy combines deficit, complexity, and dispersion:

$$\begin{aligned} g_i &= w_1 d_{c(w_i)} + w_2 h_i + w_3 v_i, \\ \widetilde{\omega}_i &= \text{clip}_{[0.25,4]}[\exp(g_i/\tau)(1 + h_i)], \qquad \omega_i = \widetilde{\omega}_i / \text{mean}_{i'}\, \widetilde{\omega}_{i'}. \end{aligned} \tag{3}$$

The weights w_1, w_2, w_3, temperature τ, and clipping range are fixed heuristic settings and were not tuned; their values are listed in Appendix A. Under both policies, the normalized weights are sampling priorities with mean one that determine file masses and local crop-bin weights for sampling with replacement from $\mathcal{R}$. Broad and refined policies differ in both the partition and the weighting rule (Appendix A).

3.4 TARGET-DIRECTED SYNTHETIC REPLACEMENT

We sample a target structure with probability proportional to its mean g_i, floored at 10^{-8}. A fixed catalog maps the target's three highest-scoring descriptor groups to decomposition, structural-process, symbolic, shapelet-episode, or hybrid generator families. We sample a generator family from these options and generate four candidates using a real anchor window as a template. After matching each candidate's mean and standard deviation to the anchor, we select the candidate with the smallest root-mean-square distance to the target centroid in a separately scaled 29-dimensional descriptor space. Final processing applies moment matching, scale and shift perturbations, and clipping based on the anchor's mean and standard deviation; outputs nearly identical to the anchor receive an additional perturbation.

3.5 EXECUTING THE FIXED-BUDGET POLICY

For each of the U successful updates, Algorithm 1 draws B real windows according to the sampling policy and replaces the scheduled windows with synthetic windows. It records the real-window origins or synthetic targets and generator choices, then updates the model using the resulting batch and the task's masks and targets.

4 EXPERIMENTAL STUDY

We evaluate Tstar under fixed pretraining budgets to test whether temporal structure-aware data construction improves downstream performance. The main experiments use MOMENT-341M and TinyTimeMixer (TTM), complemented by frozen classification probes and supervised forecasting and imputation experiments on six conventional architectures. We ask three questions: (Q1) does the construction improve downstream performance under a fixed budget (Figure 2); (Q2) which part of the construction drives the gains: rebalancing real windows, finer categories, or targeted synthesis (Figure 3); and (Q3) on which architectures (linear, convolutional, recurrent, or attention-based) and tasks (forecasting or masked imputation) does the benefit appear (Figure 4). Sections 4.2–4.4 answer Q1–Q3 in turn.

4.1 EXPERIMENTAL SETUP

The MOMENT-341M study compares four pretraining-data policies built from the same real-data source pool. Original samples real windows uniformly. Tstar-Broad rebalances their sampling using the 18 archetypes (TAT-18), while Tstar-Refined uses 35 finer temporal-structure categories and the weighting rule of Equation 3. Tstar-Refined+Synthetic retains the refined real-window weights and replaces 80% of training presentations with targeted synthetic windows. Each policy uses 102.4M window presentations per seed. MOMENT-341M (Goswami et al. 2024) is pretrained from random initialization for 50,000 updates with a global batch size of 2,048 and three paired seeds. The architecture, initialization rule, objective, optimizer, seed, and evaluation split are matched within each comparison, and hardware differences are documented in Appendix A.

TTM (Ekambaram et al. 2024) is trained from random initialization for 10,000 updates with a global batch size of 256 and six paired seeds. Raw selects windows uniformly within per-source quotas, and Tstar applies coverage-guided selection within the same quotas. Both datasets contain 197,787 real windows from 18 public sources. This real-only comparison complements the four MOMENT policies by testing coverage-guided selection in a second architecture and source pool.

Evaluation covers 98 TIME tasks (Qiao et al. 2026) and 100 FEV tasks (Shchur et al. 2025) for both foundation models. The MOMENT metrics are MASE and CRPS on TIME, and MASE and SQL on FEV. The TTM metrics are the official normalized CRPS and MASE on TIME, and mean test error and WAPE on FEV. MOMENT results are percentage error reductions computed from paired geometric-mean error ratios across tasks, with 95% dataset-group-bootstrap confidence intervals. TTM results use descriptive reductions in six-seed mean benchmark errors. Because the two models use different data and metrics, we compare policies within each model rather than across models.

The supporting supervised study evaluates NLinear, DLinear, TCN, GRU, PatchTST, and Transformer on forecasting and masked imputation. Strict Tstar combines coverage-guided real-window selection with a few synthetic windows, capped at 2% of the dataset, and is compared with Original, the real-only dataset. Each architecture is trained at seven requested training-set sizes, three capacity tiers, and five seeds (105 settings per task), with actual sizes capped by the eligible pool. The evaluation reports archetype-averaged normalized root mean squared error (macro-NRMSE), which averages

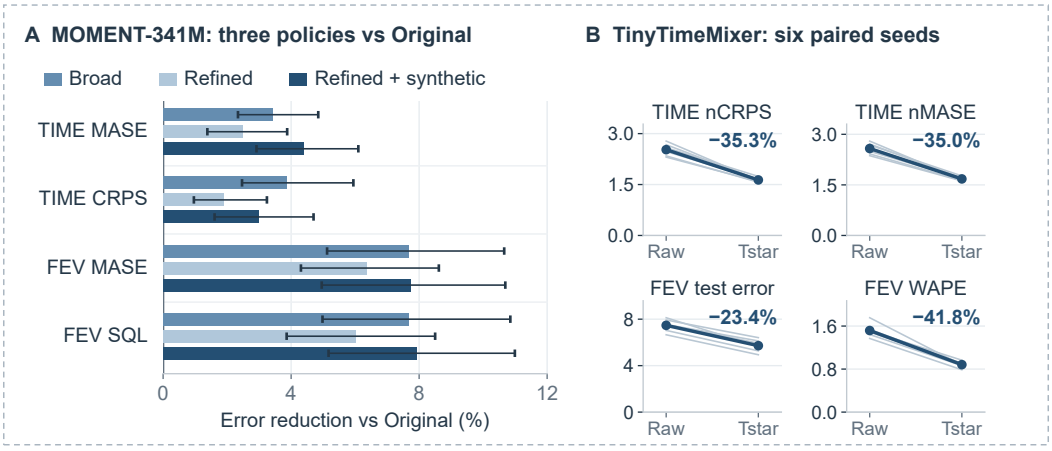


Figure 2: **Q1: forecasting gains over uniform real sampling at a fixed budget.** (A) MOMENT-341M: paired geometric error reductions relative to Original, with 95% dataset-group-bootstrap intervals over three paired seeds (98 TIME and 100 FEV tasks). (B) TTM: benchmark errors under Raw and Tstar selection; thin lines are the six paired seeds, thick lines their means, and percentages the reduction in six-seed mean error. The two panels use different estimands.

errors over archetype groups so that rare structures count equally, paired differences, and win rates on a fixed test split. Appendices C and A provide the full protocols and uncertainty calculations, and Appendix Table 10 summarizes all eight architectures.

4.2 Q1: Tstar Improves Downstream Performance under Fixed Budgets

Foundation-model forecasting. Tstar improves MOMENT-341M forecasting without adding real-data sources or increasing the pretraining budget. Relative to Original, the real-only Tstar-Broad policy reduces MASE by 3.43% on TIME and 7.67% on FEV, with 95% confidence intervals above zero on all four metrics (Figure 2A; Appendix Table 3). With refined weights and targeted synthesis, Tstar-Refined+Synthetic reduces MASE by 4.38% and 7.72%, with 95% confidence intervals of [2.92%, 6.10%] and [4.96%, 10.69%], and lowers MASE on 73 of 98 TIME tasks and 80 of 100 FEV tasks when averaged over seeds. Section 4.3 examines what refinement and synthesis add to Tstar-Broad. Because the source pool and the 102.4M presentations are fixed, these real-only gains isolate a design variable that corpus descriptions rarely report: how often each existing window is presented. Exposure allocation within an existing pool is therefore a lever distinct from source expansion, although these runs do not compare the two at matched cost.

For TTM, Tstar reduces six-seed mean TIME normalized CRPS and MASE by 35.30% and 34.97%, and FEV mean test error and WAPE by 23.41% and 41.81%, respectively, and all four aggregates improve in each of the six paired seeds (Figure 2B; Appendix A.5). Both conditions use only real windows, with dataset size, source quotas, and training budget matched. The gain therefore comes from which windows are chosen within each source: matched source quotas do not fix the temporal-structure composition (Section 4.3). TTM varies which windows enter a fixed dataset, whereas MOMENT varies how often windows are presented. The same atlas acts on both interfaces, although the two studies do not rank selection against sampling.

Frozen representations. We probe frozen MOMENT embeddings from three paired pretraining seeds on 40 classification datasets, after excluding candidates whose names or file paths match pretraining sources; this check cannot detect overlapping samples stored under other names. Using the same linear-probe protocol for every policy, Tstar-Refined+Synthetic improves accuracy, balanced accuracy, and macro-F1 over Original by 1.21, 1.59, and 1.46 percentage points, respectively, and all three 95% dataset-bootstrap intervals are above zero (Appendix Table 6). Because fingerprints and atlas labels are never model inputs, the effect of data construction reaches the pretrained representation itself, in a form that a linear probe can use without task-specific fine-tuning.

Conventional architectures. Across the six conventional architectures, weighted equally, strict Tstar lowers mean macro-NRMSE from 1.3570 to 1.3466 in forecasting and from 0.8119 to 0.8098 in

imputation relative to Original (Appendix Table 9). It has lower error in 63.0% of matched forecasting settings and 59.0% of matched imputation settings. Counting each distinct configuration once, the forecasting mean difference is not resolved, whereas the imputation difference remains negative (Appendix C.4). These averages conceal architecture- and task-specific responses, which Section 4.4 analyzes.

4.3 Q2: Reallocating Real-Window Exposure Drives the Gains

Q2 decomposes into three contrasts: whether rebalancing helps when every structure is already present, whether finer categories add value, and whether targeted synthesis improves on a strong real-only policy (Figure 3). For each contrast, and for Q3, we state the mechanism it tests and a hypothesis consistent with the result.

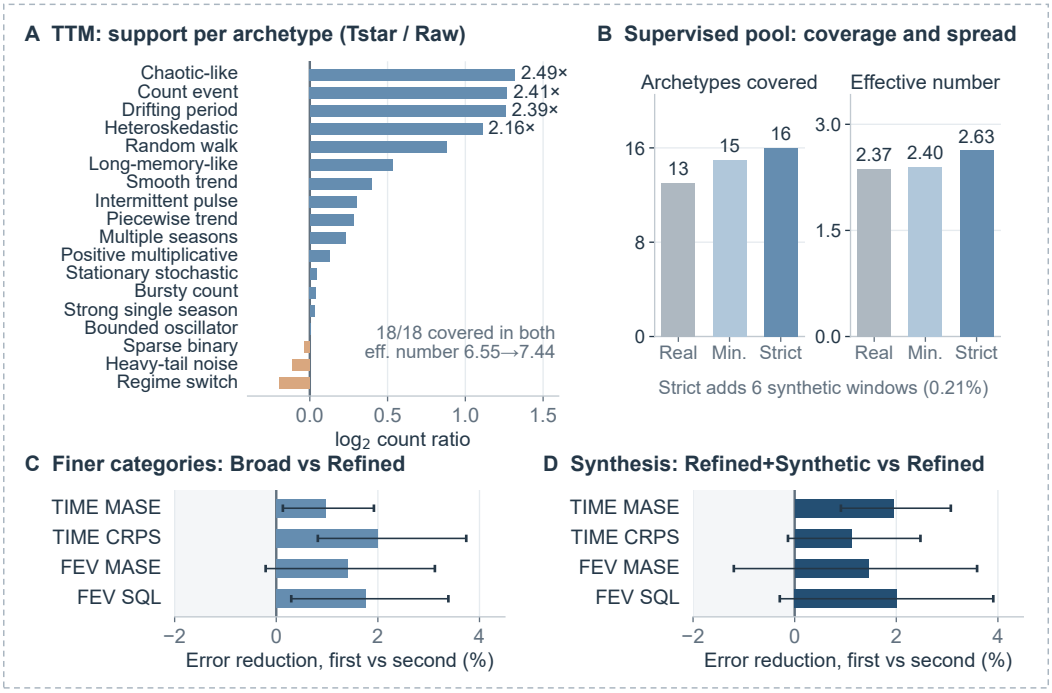


Figure 3: **Q2: which parts of the construction matter.** (A) TTM window counts per archetype under Tstar relative to Raw ($\log_2$ ratio), with dataset size and source quotas matched. (B) Supervised training sets (Real = Original, Min. = Original + minimal synthetic, Strict = strict Tstar): archetypes covered and effective number of archetypes (exponential of label entropy). (C,D) MOMENT paired geometric error reductions of the first-named policy over the second, with 95% dataset-group-bootstrap intervals over three seeds; shading marks higher error.

Balanced support improves on complete coverage. TTM benefits from selection even when all archetypes are already represented. Both Raw and Tstar cover all 18 archetypes, but selection changes the distribution of windows across them: count-event windows, for example, increase from 427 to 1,029 under matched dataset size, source quotas, and numbers of observed and missing values (Figure 3A; Appendix Table 7). Selection raises sparse structures without flattening the distribution: heavy-tail noise still accounts for 42.2% of Tstar windows (45.7% under Raw), and regime-switch windows decrease from 6,322 to 5,524. The effective number of archetypes, the exponential of label entropy, nevertheless rises from 6.55 to 7.44. No synthetic windows are added, so the gain comes from selection alone: running selection without the synthesis step reproduces the same trained model in the three seeds we checked (Appendix A.5). Beyond covering every category, rebalancing the number and composition of windows that represent each structure improves performance, so coverage can be strengthened further by raising the support of sparse structures, not only by adding missing ones. We hypothesize that part of the gain comes from the archetypes whose support more than doubles (chaotic-like, drifting-period, count-event and heteroskedastic, from 79–716 windows to

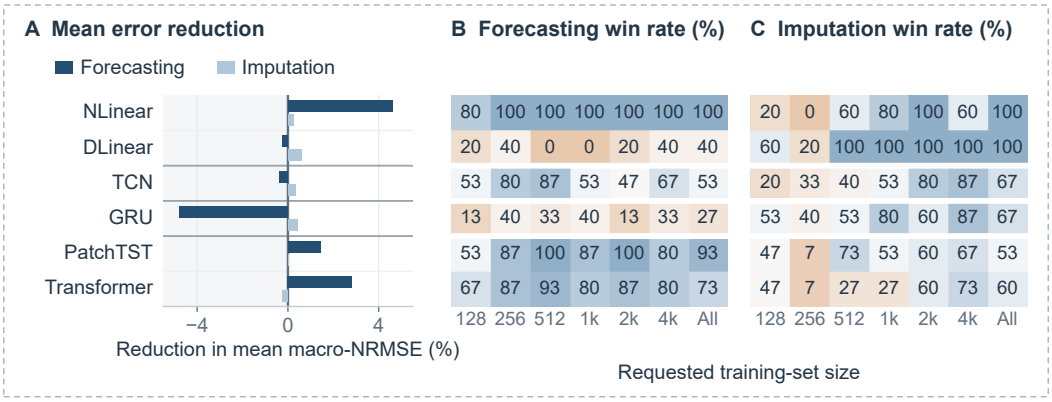


Figure 4: **Q3: which architectures and tasks benefit from strict Tstar.** Rows are grouped into linear, convolutional, recurrent, and attention-based architectures. (A) Reduction in mean macro-NRMSE relative to Original over 105 matched settings per architecture and task, $100(1 - \bar{e}_{\text{Tstar}}/\bar{e}_{\text{Original}})$. (B,C) Percentage of the 15 matched settings (three capacity labels, five seeds) in which strict Tstar has lower error, at each requested training-set size. Actual sizes are capped by the eligible pool, and linear capacity labels repeat fixed-size models.

197–1,544). This account predicts the largest error reductions on test windows of these archetypes, a per-structure analysis we leave to future work.

The supporting supervised study shows the complementary case, a pool that misses categories. Reselecting real windows and adding six synthetic windows (0.21% of the resulting 2,812-window dataset) expands coverage from 13 to 16 archetypes at nearly unchanged dataset size, and the effective number of archetypes rises from 2.37 to 2.63 (Figure 3B; Appendix Table 9). TTM is fully covered but concentrated, whereas the supervised pool both misses categories and is concentrated. Presence and support should therefore be audited separately, and which of the two cases a pool falls into determines whether reselection alone suffices or gap filling is needed. These supervised gains reflect reselection and gap filling together.

Finer categories remain effective. Tstar-Refined, which splits the 18 archetypes into 35 finer temporal-structure categories, improves on Original on all four metrics, reducing MASE by 2.48% on TIME and 6.36% on FEV with 95% confidence intervals above zero (Figure 2A; Appendix Table 3). Tstar-Broad provides the coarse real-only reference: its four paired error reductions are positive in every seed, and all four aggregate 95% confidence intervals remain above zero when retaining only the two seeds with matched hardware topology (Figure 2A; Appendix A.3). Relative to this reference, the finer categories reach comparable gains without a further increment: Broad has larger improvement point estimates on all four metrics and, in the direct paired comparison, lowers MASE by 0.98% on TIME and 1.40% on FEV (Figure 3C; Appendix Table 4). The FEV MASE interval includes zero, and only the TIME CRPS advantage retains an interval above zero in the two-seed sensitivity analysis. Because Refined changes both the partition and its weighting rule, this comparison evaluates their combined effect.

We hypothesize that the weighting rule, rather than the finer categories themselves, limits Refined. The complexity score h_i enters Equation 3 twice, in g_i and in the factor $(1 + h_i)$, shifting exposure toward windows with high change scores, spikes and outliers, and the deficit term d_c, measured against a uniform share over 35 categories, raises the weight of windows in small categories, so fewer distinct windows are repeated more often. Broad's inverse-square-root factors also upweight rare archetypes, but more mildly and without a complexity term. The hypothesis predicts that the 35 finer categories under Broad's weighting rule would match or exceed Broad, which would separate category granularity from weighting. Finer categories should thus be judged by their downstream value rather than by their number.

Targeted synthesis adds further gains. Relative to Original, Tstar-Refined+Synthetic gives the largest error reductions on three of the four metrics: 4.38% in TIME MASE, 7.72% in FEV MASE, and 7.92% in FEV SQL (Appendix Table 3). With the refined real-window weights held fixed, it

reduces MASE by 1.95% on TIME and 1.45% on FEV relative to Tstar-Refined (Figure 3D; Appendix Table 5), and the TIME MASE reduction has a 95% confidence interval of [0.91%, 3.07%]. The other three intervals include zero. With seeds 17 and 19, the two seeds with matched hardware, the TIME MASE reduction is 0.62%, and all four intervals include zero. Against Broad, all four three-seed intervals also include zero (Appendix A.3). Two features of the design shape the size of this increment. First, replacement has an opportunity cost: under a fixed budget, 80% synthetic presentations leave 20% for real windows, so a synthetic window must be worth more than the real window it displaces. Second, a strong real-only policy already supplies much of the support that synthesis targets, and TTM reaches its gains without any synthetic windows. We hypothesize that synthesis pays off mainly where the real pool lacks a structure altogether, rather than where a structure is present but under-sampled. Synthetic modules should therefore be evaluated against a strong real-only comparator, not only against uniform sampling.

4.4 Q3: Gains Depend on the Architecture and Task

The six architectures span linear (NLinear, DLinear), convolutional (TCN), recurrent (GRU), and attention-based (PatchTST, Transformer) models, each trained for forecasting and masked imputation (Figure 4). In forecasting, the gains concentrate in NLinear, PatchTST, and Transformer. NLinear reduces mean macro-NRMSE by 4.62% and wins 34 of 35 distinct requested-size–seed pairs (its three capacity labels repeat one fixed-size model), so the benefit does not require attention or recurrent modules. PatchTST and Transformer win 85.7% and 81.0% of forecasting settings and more than half at every requested size (Figure 4B). The paired forecasting effects also include losses: TCN wins 62.9% of settings despite a higher mean macro-NRMSE (Figure 5), so a policy can improve most settings while losing more in a few. In imputation, mean error decreases for five of the six architectures (Appendix Table 10). DLinear benefits most consistently, winning 29 of 35 distinct requested-size–seed pairs with a 0.59% reduction in mean macro-NRMSE, while its mean forecasting error increases by 0.25%. PatchTST and Transformer imputation win rates, by contrast, vary across sizes (Figure 4C). Being linear is therefore not sufficient to predict the response: NLinear and DLinear are both linear, yet only NLinear improves forecasting. We hypothesize that the response depends on which structures an architecture and task can exploit: forecasting must extrapolate dynamics such as trends and regime changes, whereas masked imputation can rely on the local context around each masked position.

The results of Sections 4.2–4.4 indicate how to design time-series pretraining data. First, reallocate exposure within the existing pool before expanding it. Second, audit structures by both presence and support, since complete coverage does not imply adequate support. Third, add finer categories or synthesis only when they improve on a strong real-only policy under the same budget. Finally, validate each choice per architecture and task, reporting both effect magnitude and win frequency. Tstar supplies the structural descriptors that make these checks possible, and the paired evaluations decide which intervention to adopt.

5 Conclusion and Limitations

Tstar describes time-series pretraining data with an atlas of temporal structures and uses it to select, sample, and optionally synthesize training windows. Under fixed budgets and training from random initialization, changing only which real windows are used or how often the model sees them achieves lower forecasting error. For TTM, coverage-guided selection improves all four TIME and FEV error metrics in every paired seed even though all 18 archetypes are already covered. For MOMENT, rebalancing real windows across the 18 archetypes reduces MASE by 3.43% on TIME and 7.67% on FEV. The gains also extend to frozen MOMENT representations and to six conventional architectures. These results suggest three lessons. First, temporal-structure composition should be measured and reported alongside source counts, because matched source quotas do not fix it. Second, reallocating exposure within an existing pool can precede collecting more data. Third, finer temporal-structure categories and synthesis should be evaluated against a strong structure-aware real-data policy under the same budget.

Transfer to new datasets requires refitting the atlas and sampling policy, and incomplete temporal-ordering, benchmark-overlap, and near-duplicate checks limit conclusions about unseen series. Synthetic windows were not remeasured after postprocessing, TTM results rely on seed-level aggregates, and per-structure error analyses would test the hypotheses in Section 4.3.

AI USE STATEMENT

AI tools were used for language polishing, literature searches, finding reference materials, and assisting with code. The authors are responsible for the experimental evidence, interpretation, citations, and final text.

REPRODUCIBILITY STATEMENT

Appendix A specifies the foundation-model policies, training recipes, seeds, and estimands; Appendices B and C give the atlas rules and the supervised protocol. The supplementary material contains the code, configurations, manifests, and result table of the supervised study.

REFERENCES

Ansari, Abdul Fatir, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. 2024. "Chronos: Learning the Language of Time Series." *Transactions on Machine Learning Research*. https://openreview.net/forum?id=gerNCVqqtR.

Cleveland, Robert B., William S. Cleveland, Jean E. McRae, and Irma Terpenning. 1990. "STL: A Seasonal-Trend Decomposition Procedure Based on Loess." *Journal of Official Statistics* 6 (1): 3–73.

Das, Abhimanyu, Weihao Kong, Rajat Sen, and Yichen Zhou. 2024. "A Decoder-Only Foundation Model for Time-Series Forecasting." In *Proceedings of the 41st International Conference on Machine Learning*, PMLR 235:10148–10167. https://proceedings.mlr.press/v235/das24c.html.

Ekambaram, Vijay, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. 2024. "Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series." *Advances in Neural Information Processing Systems*. https://arxiv.org/abs/2401.03955.

Fetterer, F., K. Knowles, W. N. Meier, M. Savoie, and A. K. Windnagel. 2017. "Sea Ice Index, Version 3." National Snow and Ice Data Center. https://doi.org/10.7265/N5K072F8.

Fulcher, Ben D., Max A. Little, and Nick S. Jones. 2013. "Highly Comparative Time-Series Analysis: The Empirical Structure of Time Series and Their Methods." *Journal of the Royal Society Interface* 10 (83): 20130048. https://doi.org/10.1098/rsif.2013.0048.

Gadre, Samir Yitzhak et al. 2023. "DataComp: In Search of the Next Generation of Multimodal Datasets." In *Advances in Neural Information Processing Systems* 36. https://doi.org/10.52202/075280-1179.

Goswami, Mononito, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. "MOMENT: A Family of Open Time-Series Foundation Models." In *Proceedings of the 41st International Conference on Machine Learning*, PMLR 235:16115–16152. https://proceedings.mlr.press/v235/goswami24a.html.

Gunasekar, Suriya et al. 2023. "Textbooks Are All You Need." https://arxiv.org/abs/2306.11644.

Lai, Guokun, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks." In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 95–104. https://doi.org/10.1145/3209978.3210006.

Lenssen, Nathan, Gavin A. Schmidt, Matthew Hendrickson, Peter Jacobs, Matthew Menne, and Reto Ruedy. 2024. "A GISTEMPv4 Observational Uncertainty Ensemble." *Journal of Geophysical Research: Atmospheres* 129 (17): e2023JD040179. https://doi.org/10.1029/2023JD040179.

Li, Jeffrey et al. 2024. "DataComp-LM: In Search of the Next Generation of Training Sets for Language Models." In *Advances in Neural Information Processing Systems* 37. https://doi.org/10.52202/079017-0455.

Lubba, Carl H., Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. 2019. "catch22: CAnonical Time-Series CHaracteristics." *Data Mining and Knowledge Discovery* 33: 1821–52. https://doi.org/10.1007/s10618-019-00647-x.

Nagpal, Aaryan, Debdeep Sanyal, Murari Mandal, Dhruv Kumar, and Saurabh Deshpande. 2026. "Mix, Don't Pick: Why Synthetic Corpus Composition Matters for Time Series Foundation Model Pretraining." *arXiv preprint arXiv:2606.09912*. https://arxiv.org/abs/2606.09912.

Nie, Yuqi, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. "A Time Series Is Worth 64 Words: Long-Term Forecasting with Transformers." In *International Conference on Learning Representations*. https://openreview.net/forum?id=Jbdc0vTOcol.

Penedo, Guilherme et al. 2024. "The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale." In *Advances in Neural Information Processing Systems* 37. https://arxiv.org/abs/2406.17557.

Qiao, Zhongzheng, Sheng Pan, Anni Wang, Viktoriya Zhukova, Yong Liu, Xudong Jiang, Qingsong Wen, Mingsheng Long, Ming Jin, and Chenghao Liu. 2026. "It's TIME: Towards the Next Generation of Time Series Forecasting Benchmarks." *Proceedings of the International Conference on Machine Learning*. https://arxiv.org/abs/2602.12147.

Shao, Zezhi, Yujie Li, Fei Wang, Chengqing Yu, Yisong Fu, Tangwen Qian, Bin Xu, Boyu Diao, Yongjun Xu, and Xueqi Cheng. 2025. "BLAST: Balanced Sampling Time Series Corpus for Universal Forecasting Models." In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2502–2513. https://doi.org/10.1145/3711896.3736860.

Shchur, Oleksandr, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. 2025. "fev-bench: A Realistic Benchmark for Time Series Forecasting." *arXiv preprint arXiv:2509.26468*. https://arxiv.org/abs/2509.26468.

Soldaini, Luca et al. 2024. "Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research." In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15725–15788. https://doi.org/10.18653/v1/2024.acl-long.840.

Trindade, Artur. 2015. "ElectricityLoadDiagrams20112014." UCI Machine Learning Repository. https://doi.org/10.24432/C58C86.

Varaiya, Pravin. 2009. "The Freeway Performance Measurement System (PeMS), PeMS 9.0: Final Report." California PATH, University of California, Berkeley, UCB-ITS-PRR-2009-25. https://escholarship.org/uc/item/8t01z2p8.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems* 30. https://papers.nips.cc/paper/7181-attention-is-all-you-need.

Woo, Gerald, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. "Unified Training of Universal Time Series Forecasting Transformers." In *Proceedings of the 41st International Conference on Machine Learning*, PMLR 235:53140–53164. https://proceedings.mlr.press/v235/woo24a.html.

Wu, Haixu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting." In *Advances in Neural Information Processing Systems* 34, 22419–22430. https://papers.nips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

Xie, Sang Michael, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. 2023. "DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining." In *Advances in Neural Information Processing Systems* 36. https://arxiv.org/abs/2305.10429.

Xie, Shifeng, Bahaeddine Abdessalem, Zehao Xiao, Youssef Attia El Hili, Ambroise Odonnat, Zhiwei Dong, Lei Zan, Themis Palpanas, Jianfeng Zhang, Lujia Pan, Keli Zhang, and Malik Tiomoko. 2026. "Tabby: An Open Pretraining Recipe for Time Series Foundation Models." *arXiv preprint arXiv:2609.13956*. https://arxiv.org/abs/2609.13956.

Zeng, Ailing, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. "Are Transformers Effective for Time Series Forecasting?" In *Proceedings of the AAAI Conference on Artificial Intelligence* 37 (9): 11121–11128. https://doi.org/10.1609/aaai.v37i9.26317.

Zhou, Haoyi, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting." In *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (12): 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325.

A FOUNDATION-MODEL EXPERIMENT DETAILS

This appendix documents the MOMENT-341M and TinyTimeMixer experiments of Section 4; the supporting supervised study is described in Appendix C. The two regimes use different datasets and estimands, and their scores are not pooled. All studies label windows with the same 18 TAT-18 archetypes (Table 1; named `core18` in the released code).

A.1 POLICY AND LEARNER INTERFACES

The uniform MOMENT baseline, Original, assigns equal mass to loader crop indices. Tstar-Broad multiplies inverse-square-root frequency factors computed separately for TAT-18 archetypes and for the eight observation regimes before clipping, normalization, and compilation into file and local crop sampling. The fingerprint is a construction descriptor, not an input channel. The frozen MOMENT profile contains 96,020 measured windows, distinct from 80,437,730 overlapping stride-one loader crop indices; these are not counts of independent unique examples. Source, sequence, and variable keys remain separate. The profile was fitted before the loader's 0.6/0.1/0.3 split, and chronology, benchmark-overlap, and global near-duplicate audits were not performed for it (Section 5).

A.2 FROZEN PROFILING, FIRST-MATCH LABELS AND ALLOCATION

Profiling uses length 512, start stride 512 and tails of at least 128 values. At most 16 sequence/variable proxies are retained per file, and at most 64 starts are sampled per sequence, not per file. The measured profile is a sparse inventory; it does not label every overlapping loader crop. The clustering projection has 31 coordinates and includes scale-sensitive descriptors. It uses a globally fitted RobustScaler with 10–90 quantiles and clipping to $[-12, 12]$, followed by within-archetype clustering, not within-archetype scaling or PCA. Candidate cluster counts are 2, 3 and 4; the silhouette-times-nonnegative-ARI score must exceed 0.03, with minimum cluster size 64. Rejected splits keep the archetype as one category. The frozen profile observes 17 coarse archetypes and $K = 35$ finer categories; 72 is only a maximum capacity. Table 1 lists the ordered rules used to label the frozen profile. Descriptor conventions, including raw-value slope and trend R^2, periodogram detrending and peak detection, the single-window Hurst proxy, and missing-value filling, are fixed across policies; they determine the operational labels, and scale-sensitive features are not claimed invariant.

Table 1: Frozen first-match catalog: rule order is part of the definition.

Terminal	Predicate
1. Sparse binary	$Q_b > 0.98$ or $(Q_i > 0.995 \wedge N_{\text{uniq}} \leq 2)$
2. Count node	$Q_i > 0.95 \wedge Q_+ > 0.95$; bursty if $b_{\text{burst}} > 0.55$ or $I > 0.35$ or $S > 0.04$, otherwise count event
3. Positive multiplicative	$Q_+ > 0.99 \wedge Q_{01} < 0.20 \wedge V > 1.8 \wedge D > 0.20$
4. Bounded oscillator	$Q_{01} > 0.95 \wedge Q_+ > 0.95 \wedge D > 0.40$
5. Heavy-tail noise	$t_{\text{tail}} > 6.0$ or $O > 0.03$
6. Intermittent pulse	$I > 0.60 \wedge S > 0.02$
7. Heteroskedastic	$V > 1.8 \wedge C < 1.0$
8. Smooth trend	$R^2 > 0.75 \wedge \lvert\beta\rvert > 5 \times 10^{-4} \wedge C < 1.0 \wedge D < 0.40$
9. Piecewise trend	$\lvert\beta\rvert > 5 \times 10^{-4} \wedge C \geq 1.0 \wedge D < 0.35$
10. Regime switch	$C > 1.2 \wedge \rho_1 > 0.4 \wedge R^2 < 0.95$
11. Drifting period	F finite, $F > 0.08 \wedge D > 0.15$
12. Strong single season	$D > 0.45 \wedge P \leq 1.5 \wedge H_s < 0.65$
13. Multiple seasons	$D > 0.25 \wedge P > 1.5$
14. Long-memory-like	$\rho_1 > 0.96 \wedge H > 0.58 \wedge R^2 < 0.75 \wedge \lvert\rho_4\rvert > 0.60$
15. Random walk	$\rho_1 > 0.90 \wedge R^2 < 0.80$
16. Chaotic-like	$H_s < 0.45 \wedge \rho_1 < 0.25 \wedge D < 0.20$
17. Chaotic-like	$H_s > 0.80 \wedge \rho_1 < -0.20 \wedge Z_s > 0.65 \wedge D < 0.12$
Fallback	All preceding predicates fail: stationary stochastic

Q_b, Q_i, Q_+, Q_{01} are binary-like, integer-like, nonnegative and $[0, 1]$ masses; N_{uniq} is rounded unique count. b_{burst}, I, S denote burstiness, intermittency and spike rates. $V, D, t_{\text{tail}}, O, C, F, P, H_s, H$ denote the volatility ratio, dominant power, tail strength, outlier rate, change score, frequency drift, peak count, spectral entropy and Hurst proxy. β is slope, ρ_1, ρ_4 are autocorrelations, and Z_s is difference-sign-change rate. Two chaotic rules share a terminal, while the count node has two

terminals; with the fallback there are 18 distinct labels. These are fixed heuristics, not certified physical mechanisms (in particular, not proof of chaos).

The refined weights follow Equation 3 with $w_1 = 0.45$, $w_2 = 0.35$, $w_3 = 0.20$, and $\tau = 0.7$; these are fixed heuristic settings and were not tuned. The six complexity coordinates are change score, spike ratio, outlier ratio, spectral entropy, frequency drift, and line length. Each is scaled between its 5th and 95th profile percentiles and clipped to $[0, 1]$ before averaging. Within each refined structure, centroid distances are divided by their 95th percentile (floored at 10^{-8}) and clipped to $[0, 1]$ to obtain v_i. Complexity is a feature proxy, not measured task loss. Normalization follows clipping, so final weights need not remain in $[0.25, 4]$. File mass is the mean profile weight times its weighted crop-bin count; bins are sampled by their weight times crop count, then crops uniformly within the bin. This approximates profile priorities through weighted replacement sampling, not exact presentation quotas. Broad-versus-Refined changes the weighting bundle, not only the number of labels.

Tstar-Refined and Tstar-Refined+Synthetic share these real weights. Tstar-Refined+Synthetic assigns every fifth global presentation to real data and the remaining four to synthetic data. Of the synthetic presentations, 7/20 are anchored, deriving the candidate from a real anchor window, and 13/20 are procedural, generating a parametric shape whose mean and scale are taken from the anchor. Category-average priority selects requested targets; a fixed feature-group router supplies decomposition, structural-process, symbolic, shapelet-episode and hybrid family options. Four candidates are generated per request and selected by robust-scaled 29-D centroid distance; this separately scaled projection is not the 31-D clustering projection. Final moment matching, jitter and anchor-range clipping occur after selection. We did not remeasure the categories, real-neighbor novelty, or duplicate counts of the final synthetic windows, so requested targets do not certify achieved coverage. Candidate generation cost is additional to the controlled number of learner presentations.

A.3 MOMENT PRETRAINING AND FORECASTING

MOMENT uses a randomly initialized 341,248,520-parameter architecture, length 512, patch length/stride 8/8, and 30% masked reconstruction. All four MOMENT policies use 50,000 successful optimizer updates and global batch 2,048 (102.4M presentations per seed). AdamW uses learning rate 10^{-4} to 10^{-5}, 1,000 warmup updates, cosine decay, weight decay 0.05, BF16 autocast and gradient clipping at 5.0. Seeds 17, 18, and 19 abbreviate 20260617, 20260618, and 20260619. Only final checkpoints enter primary scores. The recipe follows MOMENT's architecture and masked-reconstruction objective but is not the official MOMENT pretraining recipe. For seed 18, Original used two-GPU DDP and the other three policies used one GPU; within seeds 17 (one GPU) and 19 (two-GPU DDP), all policies share the same topology, so the two-seed analysis serves as a hardware sensitivity check.

For task t and seed s, let $e^{\text{orig}}_{t,s}$ and $e^{\pi}_{t,s}$ be the errors of Original and policy π. We average the paired log ratio within each task, $a_t = \frac{1}{3}\sum_s \log(e^{\text{orig}}_{t,s}/e^{\pi}_{t,s})$, and report geometric skill $100[1 - \exp(-\frac{1}{N_{\text{task}}}\sum_t a_t)]$. N_{task} is the benchmark task count. Twenty thousand paired bootstrap draws resample intact dataset groups, keeping their tasks and three paired seeds together; each benchmark is divided into 39 dataset groups by our evaluation wrapper. Raw scores are arithmetic task/seed means and are not the estimand of the skill column. W/T/L is descriptive over tasks after seed aggregation. Intervals are individual and not multiplicity-adjusted. TIME reports MASE and CRPS; FEV reports MASE and SQL. This bundle-level comparison does not identify which fingerprint or weighting component causes a gain.

All forecasting intervals for Tstar-Refined+Synthetic versus Tstar-Broad include zero. Against weight-matched Tstar-Refined, TIME MASE skill is 1.95% [0.91%, 3.07%], but the other three intervals include zero. Excluding seed 18 gives TIME MASE skill 0.62% [-0.58%, 1.72%], so this incremental result is not robust to that sensitivity. The Tstar-Broad-versus-Original intervals stay positive on all four metrics in the same two-seed sensitivity. These are conditional contrasts, not a monotonic law in atlas size or synthetic fraction.

MOMENT's forecasting quantiles use the same fixed evaluation wrapper: point forecasts plus Normal quantiles scaled by a robust recent-difference estimate and square-root horizon. This is not a learned probabilistic head; better CRPS/SQL under that adapter does not alone establish better calibration.

Table 2: MOMENT-341M final-checkpoint forecasting under four data policies.

Policy	TIME MASE	TIME CRPS	FEV MASE	FEV SQL
Original	1.4334	0.3245	2.8483	2.7285
Tstar-Broad	1.3778 ↓	0.3153 ↓	2.3240 ↓	2.2127 ↓
Tstar-Refined	1.3931 ↓	0.3189 ↓	2.4505 ↓	2.3512 ↓
Refined+Synthetic	1.3675 ↓	0.3232 ↓	2.3290 ↓	2.2056 ↓

Arrows show error direction versus the stated comparator (↓ lower, ↑ higher), including beside percentage reductions. Teal underlines indicate improvement; orange indicates deterioration. Marks describe point estimates, not significance. Arrows compare with Original. Arithmetic task/seed means; lower is better. Three seeds, 98 TIME and 100 FEV tasks. The paired estimators, confidence intervals and task W/T/L are in Table 3; they are not ratios of these arithmetic means.

Table 3: All MOMENT policy-versus-uniform forecasting contrasts, without selective metric omission.

Policy	Task / metric	Skill [95% CI] (%)	W/T/L
Tstar-Broad	TIME / MASE	3.43 ↓ [2.34, 4.85]	71/0/27
Tstar-Refined	TIME / MASE	2.48 ↓ [1.38, 3.88]	67/0/31
Refined+Synthetic	TIME / MASE	4.38 ↓ [2.92, 6.10]	73/0/25
Tstar-Broad	TIME / CRPS	3.85 ↓ [2.47, 5.95]	70/0/28
Tstar-Refined	TIME / CRPS	1.90 ↓ [0.96, 3.24]	61/0/37
Refined+Synthetic	TIME / CRPS	2.99 ↓ [1.61, 4.70]	70/0/28
Tstar-Broad	FEV / MASE	7.67 ↓ [5.12, 10.66]	81/0/19
Tstar-Refined	FEV / MASE	6.36 ↓ [4.31, 8.62]	78/0/22
Refined+Synthetic	FEV / MASE	7.72 ↓ [4.96, 10.69]	80/0/20
Tstar-Broad	FEV / SQL	7.68 ↓ [4.98, 10.85]	81/0/19
Tstar-Refined	FEV / SQL	6.03 ↓ [3.86, 8.50]	77/0/23
Refined+Synthetic	FEV / SQL	7.92 ↓ [5.17, 10.99]	80/0/20

Arrows show error direction versus the stated comparator (↓ lower, ↑ higher), including beside percentage reductions. Teal underlines indicate improvement; orange indicates deterioration. Marks describe point estimates, not significance. Skill uses paired geometric ratios, not the raw-score ratio. All three seeds are retained; CI resamples 39 intact dataset groups per benchmark. TIME has 98 tasks; FEV has 100. W/T/L is over tasks, not independent task-seed rows. Seed18 topology differs; intervals are not multiplicity-adjusted.

Table 4: MOMENT-341M error reductions: Tstar-Broad relative to Tstar-Refined.

Metric	All three seeds	Seeds 17 and 19
TIME / MASE	0.98 ↓ [0.13, 1.92]	0.17 ↓ [−0.37, 0.78]
TIME / CRPS	1.99 ↓ [0.82, 3.74]	0.75 ↓ [0.10, 1.63]
FEV / MASE	1.40 ↓ [−0.21, 3.12]	0.43 ↓ [−1.11, 1.98]
FEV / SQL	1.76 ↓ [0.30, 3.39]	0.95 ↓ [−0.28, 2.31]

Arrows show error direction versus the stated comparator (↓ lower, ↑ higher), including beside percentage reductions. Teal underlines indicate improvement; orange indicates deterioration. Marks describe point estimates, not significance. Entries are percentage error reductions from paired geometric error ratios [95% dataset-group-bootstrap CI]; positive values favor Tstar-Broad. The main analysis uses seeds 20260617, 20260618, and 20260619; the sensitivity retains 20260617 and 20260619. Each benchmark uses 20,000 bootstrap draws over 39 intact dataset groups. Intervals are individual and not multiplicity-adjusted.

Table 5: MOMENT-341M error reductions: Tstar-Refined+Synthetic relative to Tstar-Refined.

Metric	All three seeds	Seeds 17 and 19
TIME / MASE	1.95 ↓ [0.91, 3.07]	0.62 ↓ [−0.58, 1.72]
TIME / CRPS	1.12 ↓ [−0.13, 2.48]	−0.80 ↑ [−2.28, 0.43]
FEV / MASE	1.45 ↓ [−1.20, 3.59]	−0.72 ↑ [−3.43, 1.37]
FEV / SQL	2.02 ↓ [−0.29, 3.91]	0.11 ↓ [−1.99, 1.79]

Arrows show error direction versus the stated comparator (↓ lower, ↑ higher), including beside percentage reductions. Teal underlines indicate improvement; orange indicates deterioration. Marks describe point estimates, not significance. Entries are percentage error reductions from paired geometric error ratios [95% dataset-group-bootstrap CI]; positive values favor Tstar-Refined+Synthetic. Both policies share the same real-window weights; the synthetic policy replaces 80% of learner presentations. The main analysis uses seeds 20260617, 20260618, and 20260619; the sensitivity retains 20260617 and 20260619. Each benchmark uses 20,000 bootstrap draws over 39 intact dataset groups. Intervals are individual and not multiplicity-adjusted.

A.4 MOMENT FROZEN CLASSIFICATION PROBES

Original, Tstar-Broad and Tstar-Refined+Synthetic final checkpoints provide frozen embeddings; no Tstar-Refined classification control is available. Standardized logistic regression is selected by training-only cross-validation, then evaluated on the official test split. Three pretraining seeds are paired within each dataset. Of 66 candidate datasets, 40 remain after excluding those whose names or paths match pretraining sources; this screen does not check sample-level overlap. The estimand is the mean within-dataset paired-seed absolute score difference, with dataset bootstrap intervals, distinct from forecasting's geometric skill. Accuracy, balanced accuracy and macro-F1 are primary; log loss and ECE are secondary diagnostics. Individual intervals are not multiplicity-adjusted.

Table 6: MOMENT frozen linear probes on 40 clean-candidate classification datasets.

Metric	Original	Tstar-Broad	Refined+Synthetic	Gain [95% CI] (pp)	W/T/L
Accuracy	70.96	71.75 ↑	72.17 ↑	+1.21 ↑ [0.21, 2.43]	23/1/16
Balanced acc.	64.57	65.52 ↑	66.15 ↑	+1.59 ↑ [0.72, 2.69]	28/1/11
Macro-F1	63.63	64.44 ↑	65.09 ↑	+1.46 ↑ [0.49, 2.66]	25/1/14

Arrows show score direction versus Original (↑ higher, ↓ lower). Teal underlines indicate improvement; orange indicates deterioration. Marks describe point estimates, not significance. Macro scores in percent; three pretraining seeds and 40 candidate datasets. Gain is the paired dataset-macro difference for Refined+Synthetic versus Original, not a relative percentage. Dataset bootstrap retains paired seeds; individual intervals are not multiplicity-adjusted. Clean-candidate is not certified clean-unseen.

Tstar-Broad versus Original resolves only balanced accuracy among the three primary metrics: 0.95 pp [0.14, 1.97]. Tstar-Refined+Synthetic versus Tstar-Broad resolves balanced accuracy by 0.64 pp [0.08, 1.22]; accuracy and F1 intervals include zero. Without the Tstar-Refined classification arm, these contrasts do not isolate synthesis from the simultaneous weighting change.

A.5 TINYTIMEMIXER SCRATCH PRETRAINING AND FORECASTING

The TTM study uses the 805,280-parameter TinyTimeMixer architecture with 512-point context and 96-point prediction length. Each paired run starts from the same randomly initialized parameters. Both conditions use a pretraining dataset of 197,787 length-608 windows from 18 public sources, with matched source quotas and observed/missing budgets. Raw selects uniformly without replacement within the source quotas; Tstar uses coverage-guided real-window selection. Each dataset was constructed once (selection seed 20260726), and the six training seeds 17, 29, 42, 71, 97 and 109 vary training only; they are unrelated to the MOMENT seed labels. The selector covers all 18 reachable categories, so the residual-repair count is zero.

Raw also covers all 18 categories; Table 7 reports window counts under uniform and coverage-guided selection. Both conditions contain 120,254,496 numeric positions, including 120,146,105 observed values and 108,391 missing values. Runs that apply only the selection step, for seeds 17, 42 and 97, produce final model parameters identical to Tstar, as verified by matching model-state hashes. Together with the zero repair count, this confirms that synthetic repair is inactive in this study.

Table 7: TTM real-window counts by archetype under matched source quotas. Selection alone and the full procedure give the same counts because synthetic repair adds no windows.

Archetype	Raw	Tstar
Bounded oscillator	2,895	2,903
Bursty count	19,802	20,354
Chaotic-like	79	197
Drifting period	158	378
Count event	427	1,029
Heavy-tail noise	90,340	83,409
Heteroskedastic	716	1,544
Intermittent pulse	2,084	2,573
Long-memory-like	1,988	2,870
Multiple seasons	5,488	6,431
Piecewise trend	3,121	3,795
Positive multiplicative	6,986	7,643
Random walk	1,157	2,126
Regime switch	6,322	5,524
Strong single season	6,920	7,073
Smooth trend	1,968	2,597
Sparse binary	26,303	25,687
Stationary stochastic	21,033	21,654
Total	197,787	197,787

Each run uses 10,000 optimizer updates, global batch 256 and 2.56M window presentations. Training uses AdamW with learning rate 10^{-4}, 1,000 warmup updates, cosine decay, weight decay 0.05, BF16 and gradient clipping at 1.0. All evaluations use the final step-10,000 checkpoint. TIME evaluation covers 98 tasks and reports the official normalized CRPS and MASE; FEV covers 100 tasks and reports arithmetic mean test error and WAPE. Forecasts longer than 96 steps use recursive 96-step blocks. Requested quantiles repeat the point forecast; in this adapter, FEV test error equals SQL and MASE, and WAPE equals WQL.

The reported reductions are $100(1 - \bar{e}^{\mathrm{Tstar}}/\bar{e}^{\mathrm{Raw}})$, where each mean is computed across the six seed-level benchmark aggregates. The seed-level aggregates use TIME leaderboard values rounded to three decimal places and FEV task means. All four aggregates favor Tstar in every paired seed. A planned task-level analysis with unrounded log error ratios and a dataset-family cluster bootstrap requires task-level outputs that are not available, so we report these reductions and seed counts descriptively, without confidence intervals. A fixed benchmark crosswalk excludes the known overlapping solar tail during dataset building; full sequence-content and near-duplicate audits were not performed.

B Method Details

B.1 Windows, fingerprints, and TAT-18 archetypes

Each window is cut from a single source trajectory and carries its source, dataset, entity, and variable identifiers, window span, and split role. These fields support provenance and dataset construction; models receive only the ordered numeric values with task masks and targets. The temporal fingerprint maps a window to the descriptor groups in Table 8, following interpretable time-series features such as hctsa and catch22 and decomposition methods such as STL (Fulcher, Little, and Jones 2013; Lubba et al. 2019; Cleveland et al. 1990). The TAT-18 archetypes are operational coverage labels assigned from these descriptors by the ordered rules in Table 1; they are fixed heuristics, not validated physical mechanisms. We summarize a dataset by its number of covered archetypes and its effective number of archetypes $N_{\mathrm{eff}} = \exp(-\sum_a p_a \log p_a)$, where p_a is the share of windows in archetype a.

Table 8: Temporal fingerprint descriptor groups used for window-level coverage analysis.

Descriptor group	Examples in Tstar	Coverage role
Location and scale	mean, standard deviation, range	Basic magnitude and stability

Descriptor group	Examples in Tstar	Coverage role
Trend	trend slope, trend R^2, change score	Smooth trend, piecewise trend, drift
Autocorrelation and memory	lag-1 autocorrelation, lag-4 autocorrelation, ACF decay, Hurst proxy	Short dependence and long-memory-like behavior
Spectral structure	dominant frequency, dominant power ratio, spectral entropy, peak count, frequency drift score	Seasonality, multiple cycles, frequency drift
Volatility and regime behavior	volatility ratio, change score, outlier ratio	Heteroskedasticity, switches, heavy tails
Event and burst behavior	spike ratio, intermittent ratio, burstiness score	Pulses, bursts, sparse events
Missingness and irregularity	missing ratio, irregularity coefficient of variation	Missing or irregular observation regimes
Support and discreteness	integer-like ratio, bounded-01 ratio, binary-like ratio, positive ratio, unique ratio	Signed-real, positive-real, count, and binary-like supports

B.2 Coverage-guided selection and capped enrichment

Coverage-guided selection is real-data-first. Given a requested training-set size n, it gives selection priority to rare archetypes and unmet coverage targets and uses prototype representativeness, local density, and diversity in fingerprint space to choose representative real windows. Coverage targets are computed from the training candidate pool only; the evaluation split is never used. If real windows leave targets unmet, enrichment windows are added as capped supplements:

$$\mathcal{S}_n = \mathcal{S}_{\text{real}} \cup \mathcal{S}_{\text{repair}}, \qquad \frac{|\mathcal{S}_{\text{repair}}|}{|\mathcal{S}_n|} \leq \eta,$$

with $\eta = 2\%$ for strict Tstar in the supporting supervised study. Logged exceptions allow a small number of rows for gaps such as source families without a real anchor or missing length buckets. In the supervised study, strict Tstar exports 2,812 windows, of which 6 (0.21%) are enrichment windows. In the TTM study, real windows reach all 18 archetypes, so no enrichment windows are added (Appendix A.5).

C Supporting Supervised Study

The supporting study compares three training sets built from the same candidate pool. Original contains real windows only (2,813 windows). Original + minimal synthetic keeps the same real mixture and adds 5 enrichment windows (2,818 windows); it is an additional baseline that the main text uses only in Figure 3B, where the three training sets are labeled Real, Min., and Strict. Strict Tstar reselects real windows by coverage-guided selection and adds 6 enrichment windows (2,812 windows). Reselection also changes the source-family mix (long-horizon benchmark windows fall from 2,734 to 2,690 and public-science windows rise from 75 to 109), so the effects below concern the combined policy rather than temporal-structure composition alone.

C.1 Data and evaluation split

Training windows come from public long-horizon forecasting benchmarks, namely ETT (Zhou et al. 2021), Electricity/ECL (Trindade 2015), Traffic/PeMS (Varaiya 2009), the Jena Weather station, Exchange Rate (Lai et al. 2018), and CDC influenza-like illness surveillance, which also appear in setups such as Autoformer (Wu et al. 2021), and from public scientific observation series from NOAA, NASA GISTEMP (Lenssen et al. 2024), the USGS Earthquake Catalog, the Sea Ice Index (Fetterer et al. 2017), and WDC-SILSO sunspot numbers. Source URLs and access records are listed in `data/manifests/source_manifest.csv` of the supplementary code. All sources pass through a common windowing, task-construction, and temporal-pattern labeling pipeline before entering the candidate window pool.

All training sets use the same fixed held-out split. The validation and test splits are generated before the training sets are constructed, and the same task-specific eligible examples are used for every training set. The test split contains 1,124 exported rows, of which 1,117 are eligible for the continuous tasks. Training-set construction, coverage targets, and enrichment windows use only the training side.

C.2 Tasks, models, and protocol

Forecasting predicts the next 16 z-scored values from a 48-step context, and masked imputation reconstructs 25% randomly masked positions of a 48-step sequence, with error computed on masked positions only. The models are NLinear and DLinear (Zeng et al. 2023), TCN, GRU, PatchTST (Nie et al. 2023), and an encoder-only Transformer (Vaswani et al. 2017). TCN, GRU, PatchTST, and Transformer use three capacity tiers (tiny, small, base). NLinear and DLinear have fixed parameter counts, so their three capacity labels repeat the same model, and the uncertainty summaries in Appendix C.4 count each of their 35 distinct size–seed settings once. All runs use AdamW with learning rate 3×10^{-4}, batch size 128, at most 16 epochs with early stopping on validation macro-NRMSE, seven requested training-set sizes (128 to all), and five seeds; `configs/paper_matrix.json` in the supplementary code gives the complete settings. Each comparison fixes the task, architecture, capacity tier, requested training-set size, and seed, and varies only the training set. Requested sizes are matched; actual sizes can differ slightly because sampling is archetype-stratified and capped by the eligible pool.

C.3 Coverage and average results

Table 9 reports coverage statistics and mean macro-NRMSE for the three training sets. Table 10 reports per-architecture results for Original and strict Tstar alongside the foundation-model summaries. Strict Tstar also lowers the share of the dominant sparse-binary archetype from 82.37% to 79.77% and increases the counts of regime-switch, bursty-count, piecewise-trend, and positive-multiplicative windows.

Table 9: Coverage and mean test macro-NRMSE of the three supervised training sets. Lower macro-NRMSE is better.

Training set	Windows	Enrich.	Archetypes	N_{eff}	Forecasting	Imputation
Original	2,813	0	13	2.37	1.3570	0.8119
Original + minimal synthetic	2,818	5	15	2.40	1.3528	0.8134
Strict Tstar	2,812	6	16	2.63	**1.3466**	**0.8098**

Note. Enrich. = enrichment windows. Archetypes are covered TAT-18 archetypes; N_{eff} is the effective number of archetypes. Macro-NRMSE averages the archetype groups of the test split and weights the six architectures equally (105 settings each).

C.4 Paired uncertainty summaries

Table 11 reports strict-minus-baseline differences in test macro-NRMSE with each distinct configuration counted once: 35 settings for each linear model and 105 for each other architecture, or 490 per task. Bootstrap intervals use 10,000 resamples over configurations; win-rate intervals are Wilson intervals. Configurations still share training sets, tasks, and the held-out split, so these intervals summarize run-level variation only. In forecasting, strict Tstar wins more than half of the configurations against both baselines, but neither mean difference is resolved. In imputation, both mean differences are negative with intervals below zero. The main text reports means that weight the six architectures equally; Figure 5 shows every forecasting setting.

Table 11: Paired strict-minus-baseline differences in test macro-NRMSE over 490 distinct configurations per task. Negative differences favor strict Tstar.

Task	Baseline	Mean diff.	Bootstrap 95% CI	Win rate	Wilson 95% CI
Forecasting	Original	−0.0029	[−0.0094, 0.0038]	63.9%	[59.5, 68.0]%
Forecasting	Original + minimal synthetic	0.0019	[−0.0048, 0.0086]	61.6%	[57.3, 65.8]%
Imputation	Original	−0.0016	[−0.0028, −0.0004]	55.5%	[51.1, 59.9]%
Imputation	Original + minimal synthetic	−0.0034	[−0.0046, −0.0022]	62.0%	[57.7, 66.2]%

Table 10: Original versus Tstar training datasets under each model's matched training protocol.

A. Foundation models: model-specific error reductions

Model / policy	Benchmark / metric	Original	Tstar	Gain (%)
MOMENT / Broad	TIME MASE	1.4334	1.3778 ↓	3.43 ↓
	TIME CRPS	0.3245	0.3153 ↓	3.85 ↓
	FEV MASE	2.8483	2.3240 ↓	7.67 ↓
	FEV SQL	2.7285	2.2127 ↓	7.68 ↓
TTM	TIME norm. CRPS	2.5275	1.6353 ↓	35.30 ↓
	TIME norm. MASE	2.5788	1.6770 ↓	34.97 ↓
	FEV test error	7.4687	5.7204 ↓	23.41 ↓
	FEV WAPE	1.5172	0.8828 ↓	41.81 ↓

B. Conventional models: macro-NRMSE and matched win rates

	Forecasting		Imputation	
Model	Original → Tstar	Win (%)	Original → Tstar	Win (%)
NLinear	1.6616 → 1.5848 ↓	97.1	1.0785 → 1.0758 ↓	60.0
DLinear	1.1934 → 1.1964 ↑	22.9	0.7584 → 0.7539 ↓	82.9
TCN	1.3339 → 1.3390 ↑	62.9	0.8819 → 0.8791 ↓	54.3
GRU	1.3169 → 1.3798 ↑	28.6	0.8517 → 0.8482 ↓	62.9
PatchTST	1.2914 → 1.2726 ↓	85.7	0.6805 → 0.6802 ↓	51.4
Transformer	1.3451 → 1.3071 ↓	81.0	0.6202 → 0.6218 ↑	42.9

Teal ↓: lower error; orange ↑: higher error (point estimates). TTM gains use six-seed mean errors (6/6 wins per metric); MOMENT gains use paired geometric means, with 95% CIs in Table 3; raw scores are arithmetic means. Panel B compares real-only with strict Tstar over 105 matched settings per model/task. Linear capacity labels repeat fixed-size models.

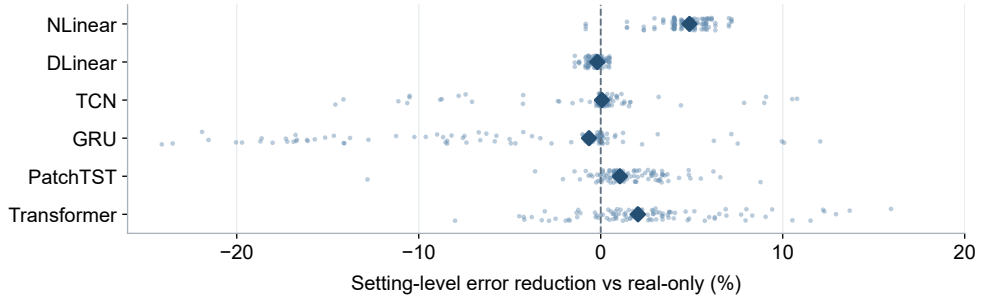


Figure 5: **Complete forecasting effect distributions.** Each dot is one of 105 matched settings per architecture for strict Tstar versus Original (for NLinear and DLinear, 35 distinct settings repeated across three capacity labels); diamonds mark medians. Positive values indicate lower macro-NRMSE. All 630 settings, including negative effects, are shown. These setting-level percentage reductions differ from the reductions in mean errors plotted in Figure 4A.