EDIT AS ACTION: REINFORCEMENT POST-TRAINING FOR TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Even a trained forecaster can mistime peaks or damp oscillations in an otherwise plausible trajectory. Correcting these features is not a local shape-matching task: an edit succeeds only if its complete forecast improves. We introduce EditCast, an outcome-guided semantic RL post-training framework built on the edit-as-action principle. It is a one-step, full-information contextual RL problem: training futures score each candidate through full-horizon mean squared error (MSE) and structural rewards, while a lightweight policy selects a semantic action from the history and reference forecast. The finite semantic action bank makes expected reward exactly computable. Across 308 checkpoint-level evaluations spanning eight datasets, four architectures, and four horizons, validation-selected EditCast reduces aggregate MSE and mean absolute error (MAE) by 1.1% and 0.8% relative to the frozen checkpoints, without updating backbone parameters. A fixed structural-reward weight raises joint accuracy–structure wins from 31 with accuracy-only editing to 59; validation selection reaches 64. These results establish outcome-guided semantic RL as a post-training stage for frozen forecasters.

1 INTRODUCTION

Forecast generation and forecast revision are different decisions. A trained predictor may track broad evolution yet mistime an event or damp its amplitude; once its trajectory exists, coordinated edits can change these properties without updating the predictor. Architectures and structure-aware losses improve the history-to-future mapping (Nie et al., 2022; Liu et al., 2023; Ma et al., 2024; Le Guen & Thome, 2019; Wang et al., 2024), while correction and adaptation show that forecasts can be revised (Zhang, 2003; Kim et al., 2025). We ask how to choose a revision of a particular forecast by the quality of its complete outcome.

The difficulty is that an apparent structural correction need not improve the prediction. A displaced peak incurs error where the target event is missed and where the forecast places it instead (Figure 1). Sharpening the late peak can better match its height while increasing squared error at the wrong time. The relevant quantity is therefore the edit's net effect over the complete trajectory, not the local feature it appears to repair (Equation 3).

Training futures make complete action outcomes observable, whereas deployment requires a decision before the future is known. This post-training asymmetry lets a policy learn from outcome feedback while acting on pre-outcome context. Language-model post-training provides the alignment paradigm (Ouyang et al., 2022; Rafailov et al., 2023); for numerical forecasts, an action transforms a continuous trajectory. We therefore use *semantic actions*, each applying a coordinated change in timing, amplitude, trend, or local shape, and learn when each is useful from its complete outcome.

EditCast instantiates this as outcome-guided semantic RL: the history and reference forecast are the context, a semantic action is the decision, and its transformed trajectory receives the realized future's reward. All actions are scored during training, so the expected reward of this one-step, full-information RL objective is computed exactly without action sampling or gradients through the scoring metrics.

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.

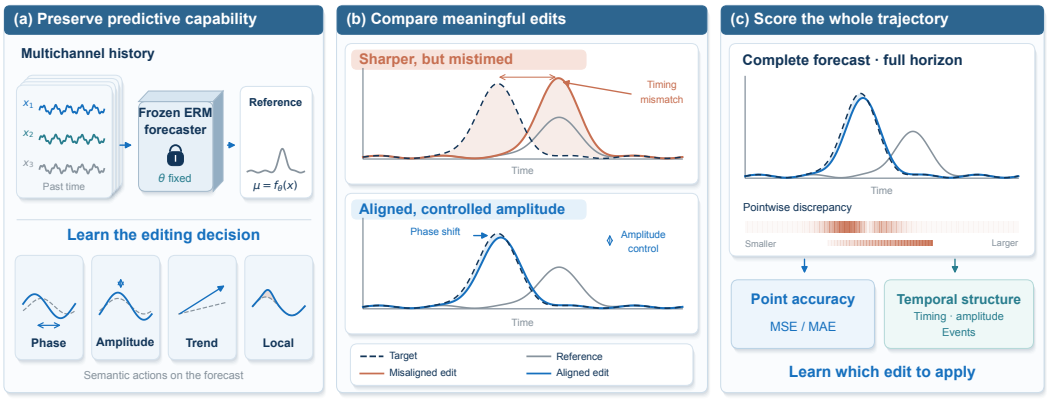

Figure 1: **A forecast becomes an editing decision.** (a) Preserve the predictor trained by empirical risk minimization (ERM) and expose temporal edits. (b) Sharpening a mistimed peak can enlarge error; coordinated timing and amplitude changes can reduce it. (c) Score the complete forecast. Schematic curves share a target (dashed navy) and reference (gray); shading shows pointwise discrepancies, and the strip displays absolute error normalized across the three schematic forecasts.

Our contributions are:

- **Forecast revision as a learning problem.** Semantic actions make a produced trajectory actionable; we learn when and how to revise it without updating its predictor.
- **Outcome-guided semantic RL.** We formulate forecast revision as one-step, full-information reinforcement post-training: a policy selects semantic actions by their complete future outcomes, and finite-bank enumeration removes action-sampling variance from the expected-reward objective.
- **Accuracy and structure beyond predictor training.** Across 308 checkpoints on eight datasets, four backbones, and four horizons, aggregate MSE and MAE decrease by 1.1% and 0.8%. A fixed structural-reward weight raises joint accuracy–structure wins from 31 to 59; validation selection reaches 64. Controlled comparisons distinguish semantic directions from generic corrections and reward-based policy learning from winner imitation.

2 Related Work

From forecast generation to revision. Decomposition and periodic representations (Wu et al., 2021; Zhou et al., 2020; Wu et al., 2022), patching and inverted attention (Nie et al., 2022; Liu et al., 2023), and linear or filtering models (Zeng et al., 2022; Xu et al., 2023; Ma et al., 2024; Yi et al., 2024) improve the history-to-future mapping. Large-scale pretraining yields reusable forecasters (Ansari et al., 2024; Das et al., 2024; Goswami et al., 2024; Rasul et al., 2023; Cohen et al., 2024); structure-aware losses supervise alignment, temporal distortion, frequency, or shape (Cuturi & Blondel, 2017; Le Guen & Thome, 2019; Wang et al., 2024; Lee et al., 2022). These advances shape the forecast-generation process. Our starting point is the trajectory it produces: its timing, amplitude, and local shape are available for a subsequent decision.

Forecast correction and adaptation. Forecast revision has appeared as learned correction and deployment-time calibration. Hybrid ARIMA–neural models learn from initial prediction errors (Zhang, 2003); output-side δ-Adapter fits a magnitude-bounded continuous residual to a reference forecast (Liang et al., 2026); and test-time adaptation addresses shifts, with TAFAS using partially observed targets and gated calibration (Kim et al., 2025). These methods parameterize revision as estimating an additive correction or calibration map. A semantic action bank exposes a different decision interface: the policy chooses among named, finite temporal transformations, such as phase and amplitude edits, whose complete trajectory outcomes are scored during training. Thus a residual adapter learns how much continuous correction to add, whereas EditCast learns which coherent transformation to apply and when its full-horizon consequence is favorable.

Learning from output outcomes. Post-training changes the behavior of an existing generator using feedback on its outputs, as in language-model alignment (Ouyang et al., 2022; Rafailov et al., 2023; DeepSeek-AI, 2025). Outcome-driven learning has also been explored for forecasts produced through language-model reasoning and numerical air-quality prediction (Zhou et al., 2025; Kang et al., 2025). We instantiate this transfer as outcome-guided semantic RL for numerical trajectories: the policy maps deployment context to a semantic action, and the realized future supplies its reward. This one-step, full-information specialization retains the policy–action–reward structure of RL, while finite actions and observable candidate outcomes make expected-reward optimization exact.

3 METHOD

EditCast is an outcome-guided semantic RL post-training procedure. It instantiates a one-step contextual RL environment over a frozen predictor: history and its reference forecast define the state, semantic trajectory transformations are actions, and the observed future supplies the reward for every action during training. The policy learns from these complete outcomes but chooses using only the context available at deployment (Figure 2).

3.1 ONE-STEP CONTEXTUAL RL OVER SEMANTIC ACTIONS

Let $x \in \mathbb{R}^{L\times C}$ be a history of L steps and C channels, and $y \in \mathbb{R}^{H\times C}$ its future over horizon H. The MSE-trained backbone f_θ is frozen and supplies the reference forecast $\mu = f_\theta(x)$. We instantiate a semantic action bank with $A = 51$ actions per channel; action a applies transformation T_a to produce candidate trajectory $z_{c,a}$:

$$z_{c,a} = T_a(x_c, \mu_c), \qquad a \in \{0, \dots, A-1\}, \qquad z_{c,0} = \mu_c. \tag{1}$$

The identity action (or no-op) $a = 0$ retains the reference forecast. Other actions alter temporal alignment (phase shifts or sampling-time warps), level and scale (amplitude rescaling, linear trends, or constant levels), or local shape (amplitude or phase changes). Local transformations use a Gaussian envelope centered at one-quarter, one-half, or three-quarters of the horizon; its smooth decay concentrates the change without truncating it. Shifts use linear interpolation with clamped endpoints, and history variability sets the scale of trend and level transformations. Appendix A.1 specifies the action bank and parameter values.

A policy $\pi_\phi(a \mid s_c)$ selects a semantic action from context $s_c = \psi(x_c, \mu_c, c)$, which summarizes history, reference-forecast shape, and channel identity. Applying the selected action produces one

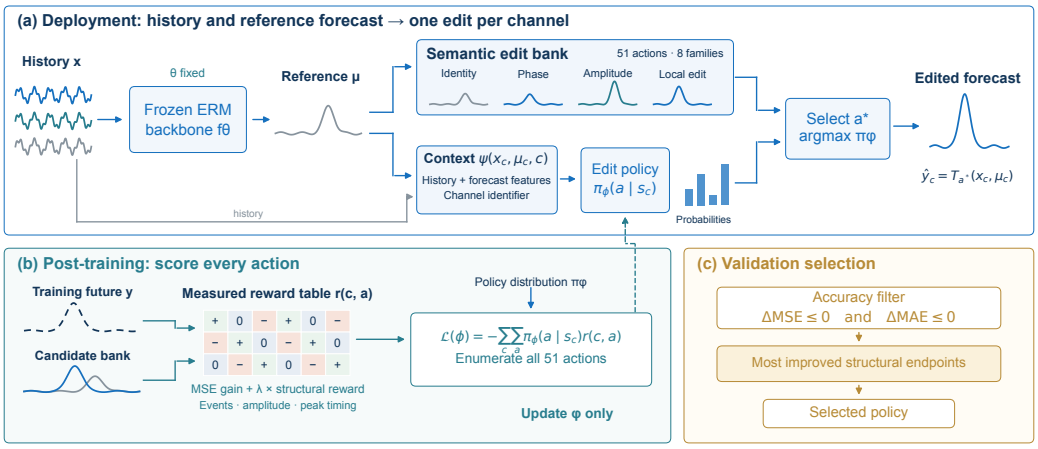

Figure 2: **Semantic-action post-training.** (a) History and the reference forecast determine one semantic action per channel. (b) Training futures score every action; expected reward updates the policy parameters ϕ. (c) Validation selects a policy using point accuracy and structural improvement. Here Δ = edited − reference. Solid arrows carry predictions or scores; the dashed arrow updates the policy.

edited forecast per channel:

$$a_c^\star = \arg\max_a \pi_\phi(a \mid s_c), \qquad \widehat{y}_c = z_{c,a_c^\star}. \tag{2}$$

One policy is trained per frozen checkpoint and dataset–horizon setting, with parameters shared across channels. Observed training futures reveal the complete reward vector for every action, making this the full-information specialization of one-step contextual RL rather than a sampled-action setting (Langford & Zhang, 2008). At deployment, the policy uses only history and the reference forecast. The semantic action bank specifies possible trajectory transformations, and the observed reward determines how the policy learns to choose among them. Because there is one action per context, each scalar reward is also the one-step reward.

3.2 Scoring the complete consequence of a semantic action

The complete horizon determines whether a candidate trajectory improves point accuracy. Let $e_c = y_c - \mu_c$ be the reference error and $\delta_{c,a} = z_{c,a} - \mu_c$ the action-induced change. Its accuracy reward is

$$r_{c,a}^{\mathrm{mse}} = \frac{\|e_c\|_2^2 - \|e_c - \delta_{c,a}\|_2^2}{HC} = \frac{2\langle e_c, \delta_{c,a}\rangle - \|\delta_{c,a}\|_2^2}{HC}. \tag{3}$$

Summing selected-action rewards over channels equals the assembled forecast's MSE reduction. The inner product measures alignment with the observed error, and the squared norm charges for the change across the horizon. A semantic action helps precisely when the alignment benefit exceeds this cost. Training computes both terms for every candidate from the observed future; deployment must choose without observing e_c. Negative rewards remain in the training feedback.

Point accuracy alone does not express which event and shape changes we value. We add four structural reward components: fewer false alarms on channels without target events, a first-order surrogate for the change in pooled event F_1, reduced dominant-frequency amplitude error, and reduced peak-timing error. Events are values above the target channel's 90th percentile, and F_1 is the harmonic mean of precision and recall. Amplitude is evaluated at the target's strongest nonzero Fourier frequency, and timing is the distance between peak locations. For signed per-action improvements $h_{m,c,a}$ in these four components,

$$g_{c,a} = \sum_{m=1}^{4} w_m h_{m,c,a}/s_m, \qquad r_{c,a}^{\lambda} = r_{c,a}^{\mathrm{mse}} + \lambda R g_{c,a}, \qquad w = (1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{4}). \tag{4}$$

Each component is divided by its training-data root mean square (RMS) s_m, and $R = \mathrm{RMS}(r^{\mathrm{mse}})/\mathrm{RMS}(g)$ matches the accuracy and structural reward scales. Both scales remain fixed during optimization, so λ controls their relative weight. The pooled-F_1 term is a first-order change around the reference forecast: it makes rewards additive across channels but does not equal the finite change in pooled F_1. We recompute all evaluation metrics on assembled forecasts. Appendix A.4 specifies the surrogate and normalization.

3.3 Learning from every candidate outcome

Each training window supplies the complete reward vector for its semantic actions. We can therefore train the policy on the expected reward of its action distribution, rather than on a sampled action or only the best-action label. Across B training windows, we minimize the negative expected reward:

$$\mathcal{L}(\phi; \lambda) = -\frac{1}{B} \sum_{n=1}^{B} \sum_{c=1}^{C} \sum_{a=0}^{A-1} \pi_\phi(a \mid s_{n,c}) r_{n,c,a}^{\lambda}. \tag{5}$$

Only policy parameters ϕ receive gradients; candidate forecasts and rewards are fixed. Because rewards add over channels and the policy factorizes, the joint action expectation uses CA terms rather than A^C combinations (Appendix A.5). For one context, let $u_{c,a}$ be the logit before probability normalization and $\bar{r}_c = \sum_b \pi_\phi(b \mid s_c) r_{c,b}^{\lambda}$ the policy's expected reward. Then

$$\frac{\partial \mathcal{L}}{\partial u_{c,a}} = -\pi_\phi(a \mid s_c)(r_{c,a}^{\lambda} - \bar{r}_c), \tag{6}$$

Algorithm 1 EditCast: from a reference forecast to a semantic action policy

Require: Frozen predictor f_θ, training/validation data, semantic transformations T_a, coefficients Λ

1: Cache reference forecasts; calibrate reward scales on the training split.
2: **for** each $\lambda \in \Lambda$ **do**
3: Initialize a policy with uniform action probabilities.
4: Each epoch: score training candidates; update ϕ_λ using Equation 5.
5: Validate hard-action predictions; retain the best eligible epoch (Section 3.4).
6: **end for**
7: Select by the count of improved validation structural endpoints; return identity if none is eligible.
8: Deploy the selected policy with Equation 2, using no future targets.

with the minibatch averaging factor omitted. Gradient descent favors actions whose observed reward exceeds $\bar{r}_c$. Enumeration computes the defined expectation exactly over actions for each training reward vector, avoiding the Monte Carlo action-sampling variance of score-function estimators (Williams, 1992). Complete feedback also permits action-value (Q) regression, which learns conditional expected rewards before choosing a semantic action; Section 4.4 compares these decision rules with winner imitation and soft targets.

The policy maps $70 + C$ context features, comprising pooled history and forecast shapes, six summary statistics, and a channel indicator, to action probabilities. Its multilayer perceptron (MLP) has two 128-unit hidden layers with hyperbolic-tangent activations. A zero-initialized output layer gives uniform initial probabilities, with identity chosen on ties. Appendix A provides the feature details, and Algorithm 1 summarizes optimization.

An action objective, not an averaged forecast. For one channel, write $p_a = \pi_\phi(a \mid s_c)$ and $\bar{z} = \sum_a p_a z_{c,a}$. Squared error obeys

$$\sum_a p_a \|y_c - z_{c,a}\|_2^2 = \|y_c - \bar{z}\|_2^2 + \sum_a p_a \|z_{c,a} - \bar{z}\|_2^2. \tag{7}$$

The decomposition distinguishes the error of an averaged trajectory from the expected error of individual edits; the latter includes their nonnegative dispersion. Equation 5 assigns the combined reward to individual semantic actions, including structural components, and deployment applies one selected transformation. Training and inference thus operate on the same action unit.

3.4 Selecting the post-trained policy

The reward coefficient $\lambda \in \{0, 0.05, 0.1, 0.25, 0.5\}$ yields five policies; validation selects the deployed one. For each coefficient, an epoch from 1–24 is eligible when its hard-action forecasts satisfy

$$\mathrm{MSE}_{\mathrm{val}}(\widehat{y}) \le \mathrm{MSE}_{\mathrm{val}}(\mu) + \epsilon, \qquad \mathrm{MAE}_{\mathrm{val}}(\widehat{y}) \le \mathrm{MAE}_{\mathrm{val}}(\mu) + \epsilon, \quad \epsilon = 10^{-9}. \tag{8}$$

For each coefficient, we retain the eligible epoch with the lowest validation MSE, breaking ties by earlier epoch. We rank these representatives by the number of defined validation structural endpoints that improve by more than 10^{-6}, then by lower validation MSE, higher λ, and earlier epoch. If none qualifies, we return the frozen predictor. MAE determines eligibility; MSE reduction remains the training accuracy reward.

4 Experiments

The experiments follow the two decisions in EditCast: what changes a forecast can usefully make, and how to choose among them before the future is observed. We first establish the accuracy and structural gains across frozen predictors, then examine the edit representation and the learning signal on matched checkpoints.

4.1 Evaluation setup

Predictors and data. Because an edit's benefit depends on the forecast it revises, we compare post-training with its corresponding frozen predictor across four MSE-trained backbones: PatchTST,

iTransformer, MixLinear, and PaiFilter, a FilterNet variant (Nie et al., 2022; Liu et al., 2023; Ma et al., 2024; Yi et al., 2024). Data comprise ETTh1, ETTh2, ETTm1, ETTm2 (Zhou et al., 2020), Electricity (ECL), Traffic (Lai et al., 2018), Weather, and Solar (Wu et al., 2021), with horizons $H \in \{96, 192, 336, 720\}$. A *cell* is a dataset–horizon–backbone–checkpoint-seed combination. The 308 cells cover 65 settings; available seeds vary by setting. Each comparison uses the same frozen checkpoint and test windows. Means weight cells equally unless stated otherwise.

Training and metrics. Policies train for 24 epochs with Adam (Kingma & Ba, 2014), learning rate 10^{-3}, cosine decay (Loshchilov & Hutter, 2017), and batch size 64; Section 3.4 defines validation selection. We report test MSE, MAE, and eight structural endpoints: peak timing, dominant-frequency amplitude and phase, cross-channel correlation discrepancy, three event-F_1 aggregations, and empty-event false alarms. Appendix A.3 gives their definitions and validity conditions.

On the test split, a *joint win* means neither MSE nor MAE increases, no defined structural metric deteriorates, and at least one structural metric strictly improves. A *strict-eight win* improves all eight structural metrics. Accuracy and structure comparisons use tolerances of 10^{-9} and 10^{-6}; undefined metrics are excluded.

4.2 Post-training improves accuracy and structure

Across 308 equally weighted paired test cells, validation-selected post-training lowers mean MSE from 0.317521 to 0.313930 (1.13%) and MAE from 0.341126 to 0.338394 (0.80%; Table 1). Every dataset-level mean improves in both errors, as do the backbone- and horizon-level averages (Appendix Table 6). Figure 6 resolves variation across dataset–backbone pairs.

Table 1: Paired test accuracy on 308 checkpoints. Cell means and checkpoint counts n are shown; s_Δ is the pooled within-setting seed SD of paired error changes (MSE/MAE), in 10^{-3} units.

		MSE ↓		MAE ↓		
Dataset	n	Frozen	Post-trained	Frozen	Post-trained	s_Δ
ECL	23	0.193858	**0.190449**	0.276697	**0.273958**	1.09/0.75
ETTh1	49	0.436154	**0.434182**	0.434298	**0.432802**	4.22/2.13
ETTh2	39	0.358394	**0.356120**	0.387737	**0.385748**	1.33/1.14
ETTm1	41	0.380030	**0.374999**	0.394994	**0.392571**	2.75/1.35
ETTm2	42	0.240443	**0.238646**	0.301488	**0.299795**	0.50/0.44
Weather	39	0.251478	**0.247305**	0.275010	**0.272947**	0.71/0.55
Traffic	9	0.523277	**0.514232**	0.332170	**0.324680**	3.82/1.64
Solar	66	0.269576	**0.264780**	0.298914	**0.294227**	1.79/2.59
All	308	0.317521	**0.313930**	0.341126	**0.338394**	2.31/1.66

Point accuracy establishes the value of revising a reference forecast; the structural-reward weight determines what else the revision preserves and improves. On the same 308 cells, fixed $\lambda = 0.1$ raises joint accuracy–structure wins from 31 under accuracy-only editing to 59, and strict-eight wins from 3 to 30, while retaining aggregate MSE and MAE improvements relative to the reference forecasts (Table 2). Validation selection reaches 64 and 35 wins. The fixed coefficient raises the joint-win count by 28, compared with a total increase of 33 after validation selection, identifying structural reward as the main source of added breadth. Figure 3 shows how these gains coexist with the paired accuracy changes.

Table 2: Accuracy and structural improvement on the same 308 cells. Joint and strict-eight wins are defined in Section 4.1. All epochs are validation-selected; s_Δ follows Table 1.

Post-training preference	MSE ↓	MAE ↓	Joint wins ↑	Strict-eight ↑	s_Δ
Accuracy only ($\lambda = 0$)	**0.313720**	**0.338134**	31	3	1.86/1.44
Fixed ($\lambda = 0.1$)	0.314466	0.338483	59	30	2.79/1.41
Validation-selected	0.313930	0.338394	**64**	**35**	2.31/1.66

The added structure is visible across several aspects of the trajectory. Amplitude and event-F_1 improve in 85–88% of cells, while peak timing and phase improve in 166 and 163. A fixed structural-reward weight improves cross-channel discrepancy in 174 cells versus 121 under accuracy-only editing; validation selection reaches 201. Overall, 293 cells improve at least one structural endpoint and 97 deteriorate on none (Appendix Table 7 and Figure 8).

The accuracy gain also persists when seeds are averaged within settings or when datasets and backbones are weighted equally (Appendix Table 8 and Section B.1).

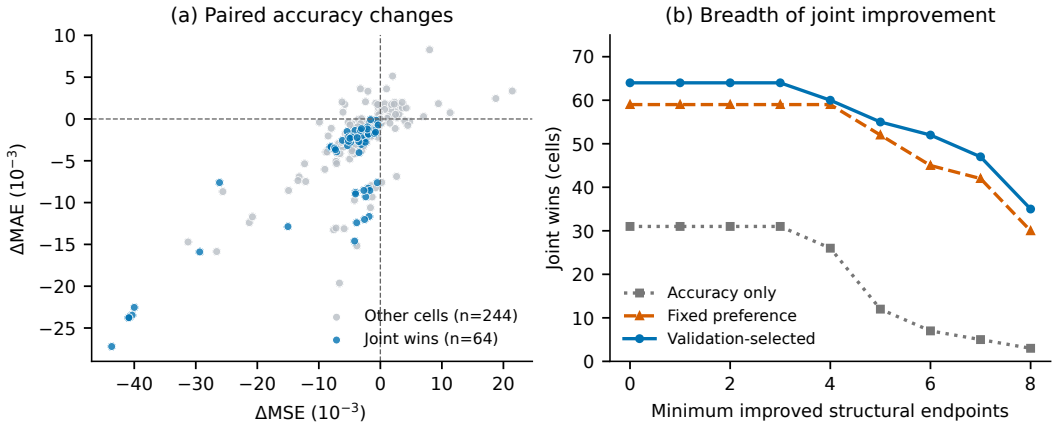

Figure 3: Joint accuracy–structure gains. (a) Each point is a checkpoint; negative changes improve accuracy, and blue marks joint wins. (b) Joint wins with at least the indicated number of structural improvements. All variants use the same 308 cells.

4.3 Semantic edits provide useful directions

The 308-cell results establish the outcome; matched controls locate the value of the action representation. We use 12 cells (Table 3): PatchTST, iTransformer, and MixLinear on ETTh1 and ETTm1 at horizon 96, with two checkpoint seeds per setting. Controls share the training split, 24 epochs, optimizer, batch size, and validation rule (Appendix B.5).

Table 3: Matched controls on 12 cells, grouped by structural-reward weight, semantic action bank, action sampling, and continuous correction. Δ reports relative error changes (%, negative is better); Imp./Deg. count improved/degraded structural metrics. Entries report mean $\pm$ pooled within-setting seed SD (two seeds per setting); see Appendix B.10 for scaling and costs.

Control	ΔMSE (%) $\downarrow$	ΔMAE (%) $\downarrow$	Imp.$\uparrow$	Deg.$\downarrow$
Full method (retrained)	-0.83 ± 0.55	-0.43 ± 0.29	4.92 ± 1.26	2.50 ± 0.91
Accuracy-only, 5 restarts	-1.28 ± 0.59	-0.72 ± 0.36	2.42 ± 1.55	4.08 ± 1.44
No local edits	-0.65 ± 0.77	-0.39 ± 0.29	5.17 ± 0.58	1.58 ± 0.76
No timing edits	-1.06 ± 0.78	-0.49 ± 0.74	5.50 ± 1.35	2.42 ± 1.19
Random library, seed 0	-0.02 ± 0.01	-0.01 ± 0.01	3.25 ± 2.99	0.92 ± 0.87
Random library, seed 1	$+0.01 \pm 0.05$	$+0.02 \pm 0.03$	2.67 ± 1.78	0.50 ± 0.58
Random library, seed 2	-0.01 ± 0.02	$+0.00 \pm 0.02$	2.08 ± 2.50	0.50 ± 0.71
Sampled PG, $K=1$	-0.27 ± 0.85	-0.20 ± 0.66	4.50 ± 2.42	1.42 ± 0.96
Sampled PG, $K=4$	-0.23 ± 0.92	-0.21 ± 0.42	4.92 ± 1.55	1.67 ± 0.58
Sampled PG, $K=16$	-0.85 ± 0.89	-0.54 ± 0.47	4.83 ± 1.73	2.50 ± 1.47
Affine calibration	-0.40 ± 0.22	-0.15 ± 0.16	1.33 ± 0.91	2.83 ± 0.91
Continuous MLP head	-0.86 ± 1.09	-0.14 ± 0.68	4.25 ± 1.50	1.75 ± 0.96
Backbone MSE fine-tuning	-0.29 ± 0.98	-0.26 ± 0.80	2.75 ± 1.50	1.83 ± 1.47

Figure 4 shows opposite outcomes for the same mistimed peak: retiming an ETTm1 channel reduces MSE by 6.3%, while amplifying that peak increases it by 24.6%.

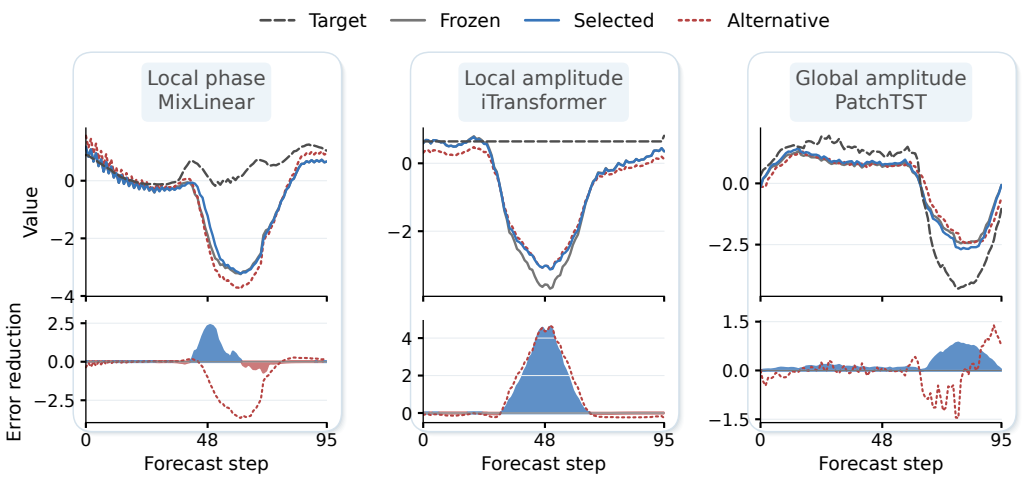

Figure 4: **Where an edit helps and where it costs.** Three ETTm1 windows ($H = 96$) compare forecasts (top) and per-step squared-error reductions (bottom). Blue/red shading marks beneficial/harmful effects of the selected edit. Dotted red shows fixed alternatives: global amplitude $+20\%$, global amplitude -20%, and phase -2 steps, respectively. Each alternative is applied to the displayed channel while leaving all other channels unchanged.

The semantic action bank supplies directions, not merely correction size. With features, policy, and reward fixed, replacing each non-identity transformation with a target-independent Gaussian direction of equal Euclidean norm leaves mean MSE changes within 10^{-4} across three random banks, whereas the semantic bank reduces MSE/MAE by 0.83%/0.43%. Direction, rather than magnitude alone, accounts for the gain; Appendix B.6 extends the control to two long-horizon cells.

The family ablation is retrained over each candidate set, so removing the 14 timing edits changes the learned policy rather than merely masking actions at inference. It improves MSE/MAE by 1.06%/0.49% versus 0.83%/0.43% for the full bank, and improves 5.50 versus 4.92 structural endpoints while degrading 2.42 versus 2.50 (Table 3). Timing edits help displaced events but can incur the pointwise double penalty when displacement is absent or misoriented, so the remaining edits form a more conservative point-error sub-bank on this cohort. The full bank instead preserves timing-specific coverage for cells where that direction is relevant: peak timing and phase improve in 166 and 163 of the 308 cells. Its role is therefore to cover coherent structural revisions, not to make every action family maximize one pooled error mean.

In another window, local amplitude scaling reduces MAE by 7.3% versus 0.9% for global scaling, despite a slightly smaller MSE gain (17.9% versus 18.8%). In a third window, global amplitude correction lowers MSE by 21.7%, whereas a phase alternative raises it by 4.8%; these paired traces expose the full-horizon cost of choosing the wrong transformation.

Under matched parameters and features, a continuous MLP reduces MSE by 0.86% versus 0.83% for semantic editing, while editing reduces MAE by 0.43% versus 0.14% and improves 4.92 versus 4.25 structural endpoints (Table 3). Thus, comparable point-error gains come with broader structural coverage when the correction is expressed as a semantic decision over the full trajectory.

4.4 Complete outcomes teach context-dependent decisions

Context determines whether an available semantic action pays off. Choosing from history and the reference forecast reduces MSE/MAE by 0.75%/0.42% on 12 cells, versus 0.21%/0.30% for a fixed action per channel (Appendix B.5). Figure 5 shows this heterogeneity across the 596,820 selected actions: an action family specifies the transformation, whereas the observed context determines its full-horizon payoff over the entire revised trajectory.

The same semantic action bank has substantial hindsight capacity: a future-aware MSE oracle reduces error by 13.89% on the 12 cells, while the evaluated deployable accuracy-only policy captures 8.27% of that gain (Appendix D). The contrast separates the transformations available after forecast generation from the information available when an action must be selected.

Training futures score every candidate trajectory, not only the hindsight winner. With the same semantic action bank, features, and network, complete-reward learning reduces MSE by 1.15% with expected-reward optimization and 1.46% with Q regression at $\lambda = 0$, versus 0.21% for best-action CE on 12 cells (Appendix C). Direct optimization exceeds CE on 11/12 checkpoints; reward differences therefore inform decisions beyond winner identity. At $\lambda = 0.1$, Q regression gives a 1.42% MSE reduction; direct optimization improves more structural endpoints (5.33 versus 3.81) and degrades fewer (2.11 versus 3.50; Table 4). Q fits conditional expected rewards, whereas Equation 5 directly updates action probabilities using each observed reward in the complete candidate-outcome vector.

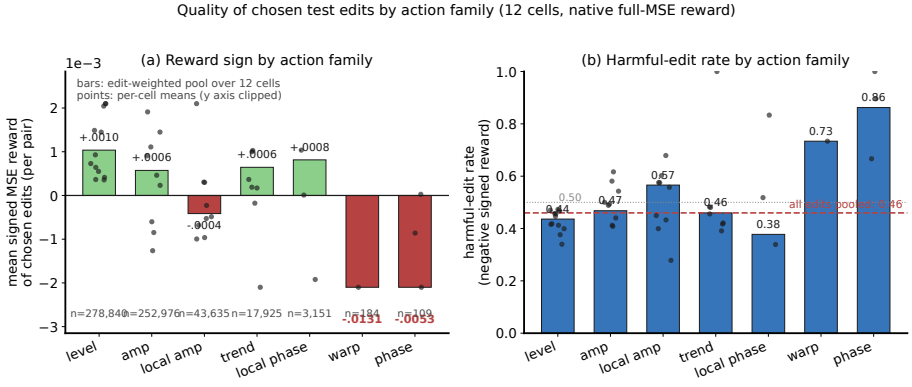

Figure 5: Action-level MSE outcomes on 12 cells: signed full-horizon reward (a) and harmful-edit fraction (b). Positive reward reduces channel MSE.

Table 4: Decision objectives at $\lambda = 0.1$ on 12 checkpoints. MSE/MAE are percentage reductions relative to the reference forecasts; Imp./Deg. count structural improvements/degradations. Mean $\pm$ SD covers three policy seeds; selection uses validation MSE/MAE. Appendix C gives the protocol and $\lambda = 0$ results.

Learning objective	MSE ↑	MAE ↑	Imp.↑	Deg.↓
Expected reward	0.62 ± 0.07	0.39 ± 0.07	5.33 ± 0.36	2.11 ± 0.46
Best-action CE	-0.33 ± 0.11	0.27 ± 0.01	4.03 ± 0.13	2.64 ± 0.13
Q regression	1.42 ± 0.05	0.33 ± 0.05	3.81 ± 0.35	3.50 ± 0.38
Soft targets, tuned τ	0.63 ± 0.17	0.41 ± 0.07	3.56 ± 0.05	3.11 ± 0.05

Tuned soft targets match MSE reduction (0.63% versus 0.62%) but improve fewer structural endpoints (3.56 versus 5.33), showing that probability matching and direct expected-reward optimization select different trajectories. Enumeration computes expected reward exactly; sampled policy gradient (PG) reduces MSE by 0.27%, 0.23%, and 0.85% at $K = 1, 4, 16$, versus 0.83% for enumeration at $K = 16$ (Table 3), preserving the exact-enumeration advantage at the largest sampling budget. With only 33,075 additional parameters, five policy runs take 212 seconds versus 460 seconds for backbone fine-tuning on 12 cells (Appendix B.10), making full-outcome learning a lightweight interface for semantic revision of a frozen forecast.

5 Conclusion

EditCast makes semantic edits a post-training decision over frozen forecasts. Across 308 paired test cells, it reduces MSE/MAE by 1.1%/0.8% and raises joint accuracy–structure wins from 31 to 59 (64 with validation selection), without backbone updates. Complete outcome vectors drive the policy: expected-reward optimization beats best-action classification on 11/12 checkpoints, improves more structural endpoints with fewer degradations than Q regression, and removes action-sampling variance through exact enumeration. The oracle–policy gap (13.89% versus 8.27%) separates action-bank capacity from context selection. Together, these findings establish semantic editing as an outcome-guided interface for aligning forecast accuracy and structure after training.

LLM USAGE STATEMENT

Generative AI was used for language polishing and writing assistance. The authors designed and conducted all experiments, verified all claims, and take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

Section 3 specifies the post-training objective and validation selection rule. Appendix A records the edit transformations, policy features, reward components, and endpoint definitions. All reported variants use the same 308 checkpoint-level cells and paired test windows. Training uses the training split, and configuration selection uses validation metrics. Each grouped table states its number of cells; these are unweighted cell means rather than pooled errors over all forecast points.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, 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 Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Ben Cohen, Haim Yedidsion, Sean Laber, Doyen Sahoo, Kami Su, Kanika Bhatia, Shuang Yang, Oleksandr Shchur, and other Datadog AI Research. Toto: Time series optimized transformer for observability, 2024. URL https://arxiv.org/abs/2412.01904.

Marco Cuturi and Mathieu Blondel. Soft-DTW: a Differentiable Loss Function for Time-Series, 2017. URL https://arxiv.org/abs/1703.01541.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models, 2024. URL https://arxiv.org/abs/2402.03885.

Inha Kang, Eunki Kim, Wonjeong Ryu, Jaeyo Shin, Seungjun Yu, Yoon-Hee Kang, Seongeun Jeong, Eunhye Kim, Soontae Kim, and Hyunjung Shim. Real-Time Long Horizon Air Quality Forecasting via Group-Relative Policy Optimization, 2025. URL https://arxiv.org/abs/2511.22169.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(17):17868–17876, 2025. doi: 10.1609/aaai.v39i17.33965. URL https://doi.org/10.1609/aaai.v39i17.33965.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 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*, 2018.

John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, 2008.

Vincent Le Guen and Nicolas Thome. Shape and Time Distortion Loss for Training Deep Time Series Forecasting Models, 2019. URL https://arxiv.org/abs/1909.09020.

Dongha Lee, Seunghyun Park, and Hwanjo Yu. Learning to embed time series patches independently, 2022. URL https://arxiv.org/abs/2212.03227.

Daojun Liang, Qi Li, Yinglong Wang, Jing Chen, Hu Zhang, Xiaoxiao Cui, Qizheng Wang, and Shuo Li. The forecast after the forecast: A post-processing shift in time series. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=syfWdclGE1.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting, 2023. URL https://arxiv.org/abs/2310.06625.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

Aitian Ma, Dongsheng Luo, and Mo Sha. MixLinear: Extreme Low Resource Multivariate Time Series Forecasting with 0.1K Parameters, 2024. URL https://arxiv.org/abs/2410.02081.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers, 2022. URL https://arxiv.org/abs/2211.14730.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, 2023. URL https://arxiv.org/abs/2305.18290.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Rishika Khorasani, Hena Ghonia, Aishwarya Bhagwatkar, George Adamopoulos, Nash Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards foundation models for probabilistic time series forecasting, 2023. URL https://arxiv.org/abs/2310.08278.

Hao Wang, Licheng Pan, Zhichao Chen, Degui Yang, Sen Zhang, Yifei Yang, Xinggao Liu, Haoxuan Li, and Dacheng Tao. FreDF: Learning to Forecast in the Frequency Domain, 2024. URL https://arxiv.org/abs/2402.02399.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting, 2021. URL https://arxiv.org/abs/2106.13008.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis, 2022. URL https://arxiv.org/abs/2210.02186.

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling Time Series with $10k$ Parameters, 2023. URL https://arxiv.org/abs/2307.03756.

Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. FilterNet: Harnessing Frequency Filters for Time Series Forecasting, 2024. URL https://arxiv.org/abs/2411.01623.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting?, 2022. URL https://arxiv.org/abs/2205.13504.

G. Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003. doi: 10.1016/S0925-2312(01)00702-0. URL https://doi.org/10.1016/S0925-2312(01)00702-0.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting, 2020. URL https://arxiv.org/abs/2012.07436.

Yitong Zhou, Yucong Luo, Mingyue Cheng, Qi Liu, Jiahao Wang, Daoyu Wang, and Enhong Chen. Time Series Forecasting via Reasoning: A Slow-Thinking Approach with Reinforcement Fine-Tuned LLMs, 2025. URL https://arxiv.org/abs/2506.10630.

A Implementation and Evaluation Details

A.1 Complete semantic action bank

For a single channel, let f_t be the reference forecast, $\bar{f} = H^{-1}\sum_t f_t$, $z_t = f_t - \bar{f}$, and σ_x the population standard deviation of the input history, clamped below at 10^{-4}. Let $S(f, g)$ denote linear interpolation of f at coordinates g, after clamping those coordinates to $[0, H-1]$. Time indices run from 0 to $H-1$. Define

$$l(t) = -1 + \frac{2t}{H-1}, \qquad w_a(t) = t + aH\sin\left(\frac{\pi t}{H-1}\right), \tag{9}$$

$$m_b(t) = \exp\left[-\frac{1}{2}\left(\frac{t-b(H-1)}{H/8}\right)^2\right]. \tag{10}$$

Table 5 specifies the complete bank. The local envelope uses centers $b \in \{0.25, 0.5, 0.75\}$. Local amplitude is defined relative to the global forecast mean. Local phase blends shifted and original values; it does not interpolate at a displacement multiplied by the envelope.

Table 5: The 51 semantic actions. Each row gives the resulting trajectory and its parameter grid. The symbol $\pm\{u, v\}$ denotes both signs of each listed value.

Family	Count	Output at time t	Parameters
Identity	1	f_t	–
Phase	6	$S(f, t+k)$	$k \in \pm\{1, 2, 4\}$
Amplitude	6	$f_t + az_t$	$a \in \pm\{0.05, 0.1, 0.2\}$
Trend	4	$f_t + a\sigma_x l(t)$	$a \in \pm\{0.05, 0.1\}$
Level	6	$f_t + a\sigma_x$	$a \in \pm\{0.02, 0.05, 0.1\}$
Warp	4	$S(f, w_a(t))$	$a \in \pm\{0.05, 0.1\}$
Local amplitude	12	$f_t + az_t m_b(t)$	$a \in \pm\{0.1, 0.2\}$; three b
Local phase	12	$f_t + [S(f, t+k) - f_t]m_b(t)$	$k \in \pm\{1, 2\}$; three b

The semantic action bank is generated from the history and reference forecast alone. No transform is fitted using the future target. The identity action is action zero, so standard first-index argmax tie-breaking returns the reference forecast when all policy logits are equal.

A.2 Policy features and optimization

Let μ_x and σ_x denote the history mean and clamped population standard deviation, and let AAP_{32} denote adaptive average pooling to 32 positions. The policy input is

$$\begin{aligned}\psi(x_c, f_c, c) = \mathrm{clip}_{[-20,20]}\big[\,&\mathrm{AAP}_{32}((x_c - \mu_x)/\sigma_x),\\ &\mathrm{AAP}_{32}((f_c - \mu_x)/\sigma_x),\\ &\mu_x, \log\sigma_x, x_{L-1,c}, \bar{f}_c, f_{0,c} - x_{L-1,c}, f_{H-1,c} - f_{0,c},\, e_c\big],\end{aligned} \tag{11}$$

where e_c is the C-dimensional channel one-hot vector. The six scalars use the input history and forecast, not their additional window-normalized versions. Feature reductions are computed in float64 and then cast back to the input dtype.

The network widths are $(70 + C) \to 128 \to 128 \to 51$, with Tanh after each hidden layer. The final weight matrix and bias are initialized to zero. Each coefficient arm is optimized for 24 epochs using Adam, initial learning rate 10^{-3}, cosine learning-rate scheduling, and minibatch size 64. Training uses every window in each epoch, in a seeded random order, and reward calibration uses the complete training split.

Inference uses hard argmax, not a probability-weighted average of candidate trajectories.

A.3 Structural endpoints

Metrics are evaluated on the final assembled prediction and the paired target. Let ε_{32} be float32 machine epsilon. A trajectory v is treated as dynamic when

$$d(v) = \mathbf{1}\Big[\max_t v_t - \min_t v_t > 32\varepsilon_{32}\max_t |v_t|\Big]. \tag{12}$$

Nonfinite predictions or targets are invalid inputs. A metric without eligible support is undefined, rather than assigned a favorable score.

Peak timing. Only dynamic target channels are eligible. There is no smoothing. For an eligible channel, the cost is

$$K(g,y) = \begin{cases} |\arg\max_t g_t - \arg\max_t y_t|, & d(g) = 1, \\ H - 1, & d(g) = 0. \end{cases} \tag{13}$$

Argmax uses the first index in a tie. Costs are averaged over eligible channels within each window, then over windows containing at least one eligible channel. The error is measured in time steps, without horizon normalization.

Dominant-frequency amplitude and phase. Let $Y = \mathrm{rFFT}(y)$ and $G = \mathrm{rFFT}(g)$. The target's dominant non-DC bin is $k^\star = \arg\max_{k \geq 1} |Y_k|$. A channel is eligible if $d(y) = 1$ and $|Y_{k^\star}| > 0$. Its amplitude cost and phase cost are

$$A(g,y) = \left| \log \max\left(\frac{|G_{k^\star}|}{|Y_{k^\star}|}, \varepsilon_{32} \right) \right|, \tag{14}$$

$$\Phi(g,y) = \begin{cases} 1 - \cos(\arg G_{k^\star} - \arg Y_{k^\star}), & |G_{k^\star}| > 0, \\ 1, & |G_{k^\star}| = 0. \end{cases} \tag{15}$$

Both use equal-weight eligible-channel means followed by eligible-window means. Phase is not amplitude-weighted and is not converted to a time lag. FFT inputs use float32.

Cross-channel discrepancy. For each window, center each channel of $v \in \{g, y\}$ and normalize its temporal vector to unit Euclidean norm. Zero a column unless both its target and the corresponding v channel are dynamic; replace a zero normalization denominator by one. Let $U_v \in \mathbb{R}^{H \times C}$ be the resulting matrix and $C_v = U_v^\top U_v$. The endpoint is

$$X(g,y) = \frac{\|C_g - C_y\|_F}{\max(\|C_y\|_F, 1)}. \tag{16}$$

It is averaged over windows with at least two dynamic target channels. Diagonal entries are included, and these operations use float64. This is a time-domain correlation discrepancy, not spectral coherence.

Event precision and recall. For each window–channel pair, let $q = \mathrm{quantile}_{0.9}(y)$, computed with linear interpolation. Define $e_t = \mathbf{1}[y_t > q]$ and $\widehat{e}_t = \mathbf{1}[g_t > q]$. The same target-derived threshold scores prediction and target. Write $P = \sum_t e_t$, $Q = \sum_t \widehat{e}_t$, and $T = \sum_t e_t \widehat{e}_t$. Matching is pointwise at the same time index; no temporal tolerance, segmentation, or minimum duration is applied. The channel score is $F_1 = 2T/(P + Q)$, with an empty/empty pair assigned one.

The window–channel macro score averages channel scores within windows, then across windows. Positive-channel macro F_1 averages over all window–channel pairs with $P > 0$. Micro F_1 pools the counts across all windows and channels before computing $2TP/(2TP + FP + FN)$, assigning one when its denominator is zero.

False alarms on empty-event channels. A pair is empty when its target has $P = 0$, not when the dataset variable is permanently constant. The false-alarm endpoint is

$$\mathrm{FA}(g,y) = \frac{\sum_{n,c:P_{nc}=0} Q_{nc}(g)}{H \,\#\{(n,c) : P_{nc} = 0\}}. \tag{17}$$

It is undefined when there are no empty target pairs. This accounts for the 217-cell denominator of Table 7. The positive-channel macro endpoint is likewise undefined without positive support.

A.4 Additive structural reward and training calibration

The structural reward uses four endpoints: empty-event false alarms, micro event F_1, dominant-frequency amplitude, and peak timing. The other structural endpoints are used for evaluation, not

as reward components. All calibration below uses the complete training split and is fixed before optimization of the coefficient arms. Let f denote the baseline and $g = z_{n,c,a}$ the candidate at a particular window and channel. Counts and channel costs follow Appendix A.3.

For false alarms, let $E = H \sum_{n,c} \mathbf{1}[P_{nc} = 0]$. The gain is

$$h_{\mathrm{FA}}(n, c, a) = \begin{cases} \mathbf{1}[P_{nc} = 0]\,[Q_{nc}(f) - Q_{nc}(g)]/E, & E > 0, \\ 0, & E = 0. \end{cases} \tag{18}$$

For micro F_1, let T_0 be the split-global number of baseline true positives and $D_0 = 2TP_0 + FP_0 + FN_0$, replacing $D_0 = 0$ by $\widetilde{D}_0 = 1$ and otherwise setting $\widetilde{D}_0 = D_0$. A channel edit changes the true-positive count by ΔT and the predicted-positive count by ΔQ. Since the target is fixed, its change to the F_1 denominator is $\Delta D = \Delta Q$. The additive differential is

$$h_{\mathrm{micro}}(n, c, a) = \frac{2\Delta T\, \widetilde{D}_0 - 2T_0 \Delta Q}{\widetilde{D}_0^2}. \tag{19}$$

This linearizes the ratio around the frozen baseline. Summing such differentials does not equal the finite change in joint micro F_1, which is why that endpoint is recomputed after assembling the selected edits.

For $m \in \{\mathrm{amp}, \mathrm{peak}\}$, let v_{nc}^m be the target-validity mask, $V_n^m = \max(1, \sum_c v_{nc}^m)$, and N_m the number of training windows with at least one valid channel. The corresponding channel costs are $K_{\mathrm{amp}} = A$ and $K_{\mathrm{peak}} = K$ as defined above. Their gains are

$$h_m(n, c, a) = \frac{v_{nc}^m\,[K_m(f_{nc}, y_{nc}) - K_m(g_{nc}, y_{nc})]}{V_n^m N_m}. \tag{20}$$

This uses the same channel-then-window averaging as the endpoint and requires eligible training support for that component.

RMS normalization includes every training window, channel, and action, including the identity:

$$s_m = \sqrt{\frac{1}{NCA} \sum_{n,c,a} h_m(n, c, a)^2}, \qquad g_{n,c,a} = \sum_m w_m h_m(n, c, a)/s_m. \tag{21}$$

A zero or nonfinite s_m is replaced by one. We set $R = \mathrm{RMS}(r^{\mathrm{mse}})/\,\mathrm{RMS}(g)$, taking $R = 0$ when the denominator is zero or nonfinite. The resulting $r^\lambda = r^{\mathrm{mse}} + \lambda R g$ is the reward in Equation 5. Neither the per-component normalization nor R is recomputed per minibatch. Only training rewards enter the optimization loss.

A.5 Why channel-wise enumeration is sufficient

Let $\mathbf{a} = (a_1, \ldots, a_C)$ and define its additive training reward as $r(\mathbf{a}) = \sum_c r_{c,a_c}$. For the factorized policy,

$$\mathbb{E}_{\mathbf{a}}\, r(\mathbf{a}) = \sum_{\mathbf{a}} \left[\prod_c \pi_\phi(a_c \mid s_c)\right] \sum_c r_{c,a_c} \tag{22}$$

$$= \sum_c \sum_{a_c} \pi_\phi(a_c \mid s_c) r_{c,a_c}. \tag{23}$$

All other channel probabilities sum to one. This reduces the action summation from A^C joint configurations to CA terms. For the MSE component, Equation 3 additionally proves that the summed reward is the actual whole-forecast reduction. For structure, the equality concerns the specified additive training surrogate.

The objective updates probabilities over fixed candidate trajectories. It neither backpropagates through the frozen backbone nor regresses to an oracle-selected action label. Every candidate's signed reward contributes, including harmful edits; predicting which choices benefit a new history remains the policy's generalization task.

Table 6: Paired accuracy changes by backbone and horizon. Δ denotes post-trained minus frozen; negative is better. The same 308 cells are regrouped in each block. All backbone and horizon averages improve.

Group	n	ΔMSE $\downarrow$	ΔMAE $\downarrow$
PatchTST	72	-0.001046	-0.004401
iTransformer	142	-0.003558	-0.001721
MixLinear	35	-0.012648	-0.006951
PaiFilter	59	-0.001405	-0.000627
$H = 96$	135	-0.004532	-0.003313
$H = 192$	66	-0.002465	-0.001811
$H = 336$	61	-0.002002	-0.002270
$H = 720$	46	-0.004554	-0.002962

A.6 Data processing and selection details

All settings use context length $L = 96$, multivariate inputs, stride-one windows, and channel-wise standardization fitted on training rows only. The standardized space is used for backbone outputs, edits, rewards, and reported metrics; values are not inverse-transformed before scoring. Window-level policy normalization is an additional feature transformation.

ETTh uses chronological target boundaries at rows 8640, 11520, and 14400; ETTm uses 34560, 46080, and 57600. ECL, Traffic, Weather, and Solar allocate $\lfloor 0.7T \rfloor$ rows to training and $\lfloor 0.2T \rfloor$ to test, with the remainder for validation. Validation and test splits include the preceding L observations as context, while their prediction targets remain in the assigned split. A span of S rows yields $S - L - H + 1$ windows.

When a backbone requires decoder input, it receives the last 48 observed values followed by H zeros; future values are masked before backbone inference. Calendar features are supplied through the backbone's native interface and are not additional edit-policy inputs.

Within each coefficient arm, validation eligibility is determined by both point errors relative to the paired baseline, with tolerance 10^{-9}. The representative is the eligible learned epoch (1–24) with minimum validation MSE, breaking ties by earlier epoch. Epoch zero is the untrained reference and is excluded. Representatives are ranked by the number of strictly improved validation structural endpoints, lower validation MSE, higher coefficient, then earlier epoch. The identity is returned if no arm has an eligible representative. The fixed-0.1 and zero-coefficient variants apply the same epoch eligibility rule within their single available arm.

Validation eligibility and within-arm MSE minimization use unrounded float64 accuracy values; Tables 1 and 2 report accuracy from the same evaluation. Across arms, structural improvement counts and MSE tie-breaking use six-decimal validation values. Paired test comparisons likewise use six-decimal metric values. Differences are computed before applying 10^{-9} tolerance for accuracy and 10^{-6} for structural endpoints.

Tables 1 and 6 give the dataset, backbone, horizon, and checkpoint coverage. All three variants use the identical 308-cell set.

B Additional Evidence and Evaluation Analyses

All analyses in this section use the same validation-selected configurations as the main results. A setting combines a dataset, backbone, and horizon; multiple checkpoints within it differ in seed. Figure 6 displays the dataset–backbone means with their checkpoint counts, complementing the marginal results in Table 6.

B.1 Sensitivity to aggregation weights

Figure 7 and Table 8 compare four aggregation rules. Checkpoint weighting gives each of the 308 cells equal weight. Setting weighting first averages seeds within each of the 65 settings, then averages settings. Dataset weighting averages those setting means within each dataset and then weights the

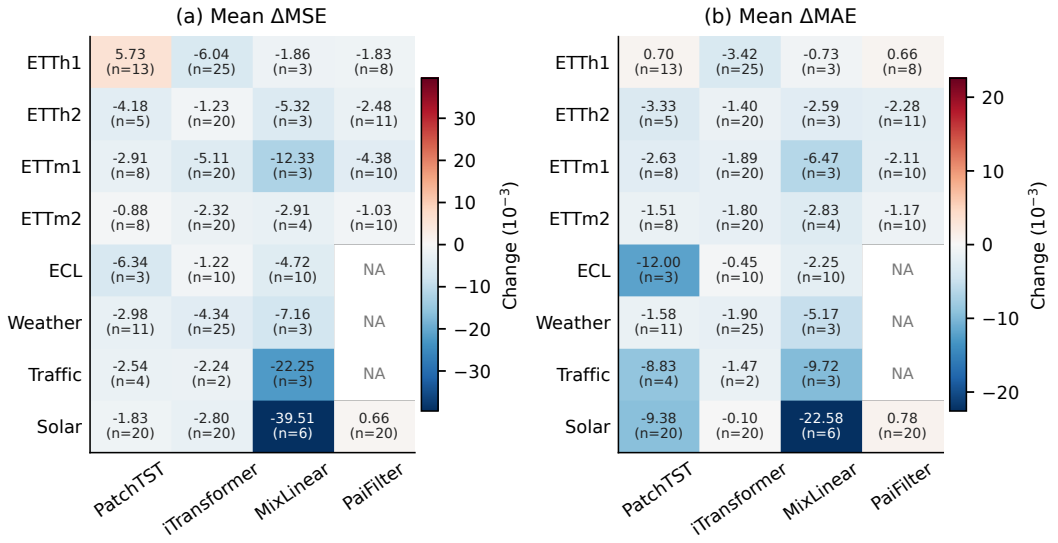

Figure 6: Accuracy changes by dataset and backbone, averaging the available checkpoints and horizons within each entry. Negative is better. Each entry reports its value in units of 10^{-3} and its checkpoint count. NA indicates no evaluated cell in this matched set. The two panels use separate symmetric color scales.

Table 7: Structural changes for validation-selected post-training. Counts use paired test forecasts and the evaluation tolerance. Empty-event false alarms are defined on the 217 cells with an empty-event support.

Endpoint	Valid cells	Improved	Tied	Worsened
Peak timing ↓	308	166	51	91
Dominant-frequency amplitude ↓	308	262	21	25
Phase error ↓	308	163	60	85
Cross-channel discrepancy ↓	308	201	53	54
Event F_1 ↑	308	272	13	23
Micro event F_1 ↑	308	265	13	30
Positive-channel macro F_1 ↑	308	270	13	25
Empty-event false alarms ↓	217	137	3	77

eight datasets equally. Backbone weighting instead averages setting means within each backbone and then weights the four backbones equally. These estimates summarize the evaluated coverage; each rule re-aggregates the observed settings without imputing unavailable cells.

Table 8: Aggregation sensitivity of validation-selected post-training. Negative changes favor the edited forecasts; values are relative changes in percent against the frozen baseline under the same weighting. The aggregation unit changes, while the underlying evaluated cells remain fixed.

Weighting rule	ΔMSE (%)	ΔMAE (%)
Equal checkpoint weight	−1.13	−0.80
Equal setting weight	−1.26	−0.88
Equal dataset weight	−1.31	−0.94
Equal backbone weight	−1.50	−1.01

B.2 Structural gain distributions

Figure 8 shows how cell-level gains vary across the eight structural endpoints counted in Table 7.

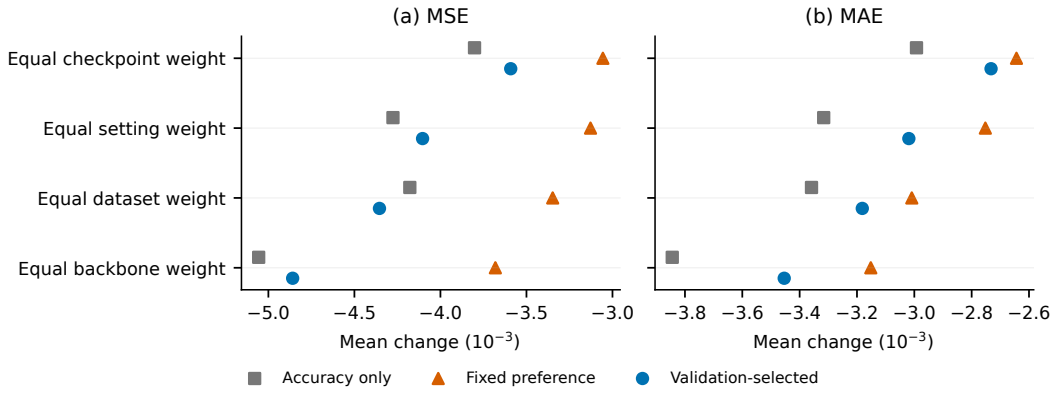

Figure 7: Point-accuracy changes remain favorable under four aggregation rules. Points show the exact paired means under the stated weighting; within-setting checkpoint dispersion is summarized in Appendix B.10. Color and marker shape identify the post-training variants. All estimates use the same paired test results.

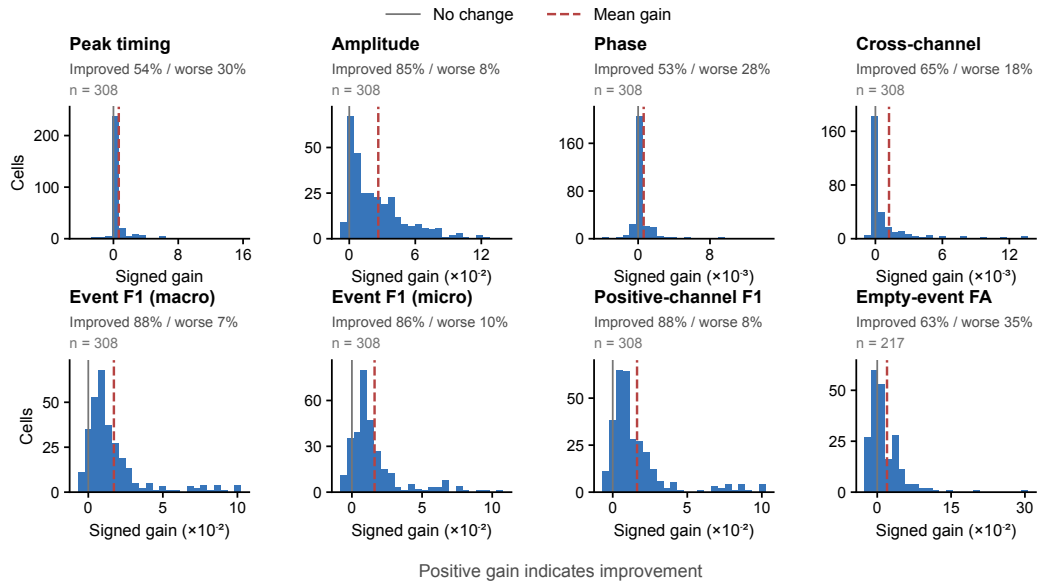

Figure 8: Structural gain distributions across checkpoints; positive is better. Dashed lines mark full-set means. Histograms show the 1st–99th percentiles. Rates use 308 cells, except the false-alarm metric, which has 217 eligible checkpoints.

B.3 Paired joint-win transitions

The increase in total joint wins can be resolved into retained, gained, and lost wins on the same checkpoints (Table 9). Relative to accuracy-only post-training, the full method retains 19 joint wins and adds 45, while 12 accuracy-only wins no longer satisfy the full criterion. This gives the net increase of 33. Relative to a fixed structural preference, the net increase is five. These paired transitions distinguish the increase in overall joint success from uniform improvement of every individual configuration.

B.4 Geometry of structural gains

Figure 9 projects the eight-dimensional gain vectors for the 217 cells with all endpoints defined. The projected positions show dataset-associated grouping, while joint wins appear in several regions rather than a single isolated cluster. This descriptive view complements the endpoint counts without assigning a cause to the geometry of the embedding.

Table 9: Joint-win transitions when replacing each reference variant with validation-selected post-training. Retained: both are joint wins; gained: only the full method wins; lost: only the reference wins; neither: neither wins. Each row covers all 308 matched cells.

Reference	Retained	Gained	Lost	Neither
Accuracy only	19	45	12	232
Fixed $\lambda = 0.1$	39	25	20	224

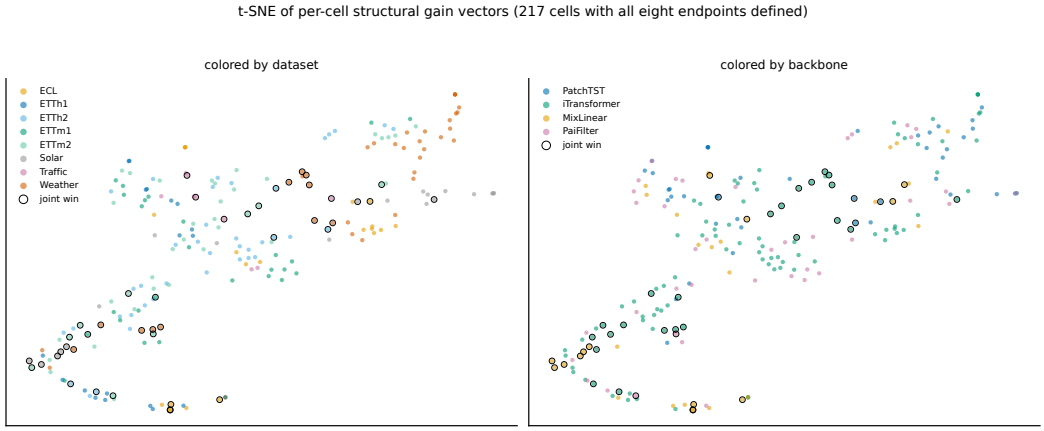

Figure 9: t-SNE embedding of per-cell structural gain vectors (217 cells with all eight endpoints defined; perplexity 30, gains clipped at the 1st/99th percentile for embedding geometry only). Left: colored by dataset. Right: colored by backbone. Open circles mark the 43 joint wins within this subset. Clusters organize primarily by dataset rather than backbone, and joint wins concentrate in regions of broad positive gain rather than forming a separate island.

B.5 Mechanism-control protocol

The mechanism controls compare alternative trajectory transformations, selection rules, and optimization procedures on the same reference forecasts. We use a fixed 12-cell cohort spanning PatchTST, iTransformer, and MixLinear on ETTh1 and ETTm1 at horizon 96, with checkpoint seeds 1 and 2.

Shared protocol. All controls retrain for 24 epochs on the complete training split with Adam, cosine scheduling, batch size 64, and the same validation protocol. Within each candidate arm, the representative is the learned epoch with minimum validation MSE among epochs passing both validation MSE and MAE gates (10^{-9} tolerance against the paired reference baseline). Across candidate arms, selection maximizes the number of validation structural improvements ($> 10^{-6}$), then minimizes validation MSE. The reference forecast serves as the identity fallback when no learned epoch meets the accuracy gates.

Control definitions.

- **Retrained full method.** The complete semantic action bank, features, expected-reward objective, and five structural coefficients, trained under the shared control protocol. Structural gain scales are calibrated on the training split and are not refitted per variant.
- **Accuracy-only restarts.** Five independently seeded runs with coefficient 0, selected with the same validation rule. This isolates the structural term in the training reward under the shared validation selector.
- **Action-family ablations.** *No local* removes the 24 local amplitude/phase edits; *no timing* removes the 14 phase, warp, and local-phase edits. Features, policy input dimension, and calibration are unchanged.
- **Random libraries.** All 50 non-identity edits are replaced by target-independent Gaussian directions, with per-window, per-channel, per-action ℓ_2 norm matched exactly to the native edit they replace. Three libraries with fixed seeds 0, 1, 2 are trained and reported separately.

Table 10: Structure-aware training on eight matched from-scratch cells: PatchTST and iTransformer on ETTh1/ETTm1, H=96, two seeds each. Values are means; relative changes compare each loss against the MSE-trained reference. Appendix B.7 specifies the loss settings.

Training approach	MSE ↓	MAE ↓	ΔMSE	ΔMAE	Event F_1 ↑
ERM (MSE, from scratch)	0.366246	0.391358	–	–	0.124
Soft-DTW (from scratch)	0.494684	0.463528	+35.1	+18.4	0.180
Frequency-domain (from scratch)	0.899241	0.631336	+145.5	+61.3	0.117

- **Sampled policy gradient.** The exact expected-reward loss is replaced by a K-sample score-function estimator per window and channel with the exact state value (from the same reward vectors) as a detached baseline. The estimator is unbiased; the comparison reports final outcomes at $K \in \{1, 4, 16\}$ against enumeration of all 51 actions. Reward computation cost is counted identically in both.
- **Generic post-training.** *Affine* applies per-channel scale and shift (14 parameters). *Continuous MLP* adds a two-layer Tanh head on the identical native policy features with zero-initialized output (33,042 parameters vs. 33,075 for the edit policy). *Backbone fine-tuning* unfreezes the native checkpoint and continues MSE training in native train mode (about 1.8×10^6 parameters). Each uses five learning rates $\{10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}\}$ as its five candidates, trained with the same float64 full-horizon MSE loss.
- **Static edit controls.** Global-fixed and per-channel-fixed choose actions by maximizing training reward over the same five coefficients with the same validation gates, requiring no policy network. They are evaluated with the identical metric pipeline.

Wall-clock times in Table 3 include training and validation of all five candidate runs; backbone fine-tuning additionally performs live data loading. Parameter counts are exact module counts.

B.6 Long-horizon mechanism cohort

The semantic-versus-random comparison extends to two H=720 cells chosen by forecast availability before control evaluation: iTransformer on ETTh1 (seed 1) and PatchTST on Weather (seed 5). Each cell uses the same 24-epoch, validation-selected protocol to compare the full method, one magnitude-matched random library (seed 0), and accuracy-only editing. The full method reduces MSE/MAE by 4.1%/2.5% and 1.2%/0.3%, with six improved structural endpoints and one and two degraded endpoints, respectively. Accuracy-only editing reduces MSE by 2.9% and 1.3%, with four and five improved endpoints and three and one degraded endpoints, respectively. The random library has no validation-eligible learned epoch in either cell and therefore returns the reference forecast. These outcomes show the semantic-direction contrast at a longer horizon while retaining the same selection rule.

B.7 Structure-aware training as an alternative intervention

Changing the backbone's training loss asks whether structural objectives can be incorporated at forecast generation rather than through outcome-based editing. Table 10 compares PatchTST and iTransformer trained from scratch on ETTh1 and ETTm1 (H=96, seeds 1–2, eight cells). All arms use the same model configurations, splits, train-only normalization, Adam at learning rate 10^{-3}, cosine decay to zero, batch size 64, and 24 epochs; epochs are selected by validation MSE. The losses are plain MSE (the ERM reference), Soft-DTW (Cuturi & Blondel, 2017) with γ=1, and $\mathrm{MSE} + \alpha\,\mathrm{MSE}(\mathcal{F}(\widehat{y}), \mathcal{F}(y))$ with α=1 over the real FFT, following FreDF (Wang et al., 2024). Test evaluation uses the same MSE, MAE, and structural endpoints as the post-training experiments.

At these loss settings, Soft-DTW raises event F_1 by 0.056 and improves amplitude error by 0.081 on average, while MSE and MAE increase by 35.1% and 18.4% relative to MSE training. The frequency-domain loss raises MSE by 145.5% and has mixed structural effects. The comparison shows how changing the training objective alters the accuracy–structure balance; post-training instead changes decisions over the outputs of existing frozen checkpoints.

B.8 Action usage and individual edit outcomes

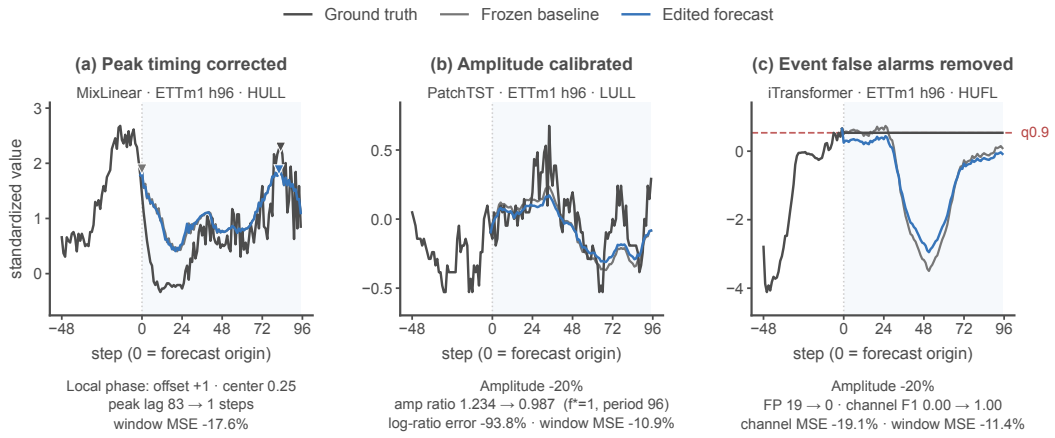

Figure 10: Real edits on ETTm1 test windows (H=96). (a) Peak-timing error falls from 83 steps to one. (b) Dominant-frequency amplitude overshoot falls from 23% to below 2%. (c) All 19 false alarms on an empty-event channel disappear (F_1: $0 \rightarrow 1$). The displayed windows illustrate trajectory-level effects of semantic edits.

The selected policies use the same semantic action bank differently across backbones and windows. Figures 10 and 11 show, respectively, the trajectory-level effect of individual actions and the distribution of chosen action families.

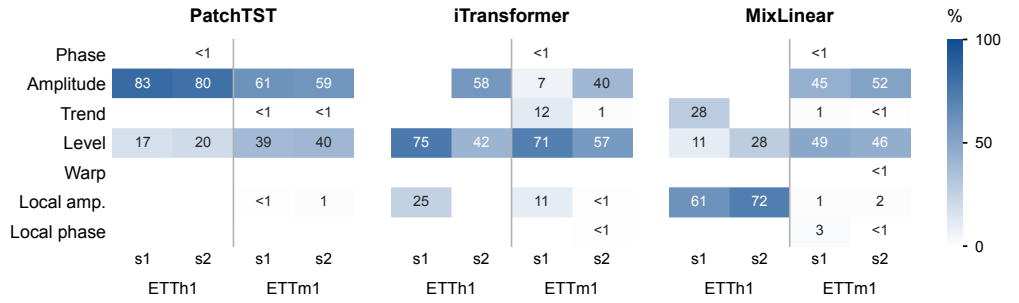

Figure 11: Action-family usage (%) across 12 validation-selected policies: three backbones, two datasets, and two seeds. Shares are rounded; blanks denote zero.

Figure 11 shows how the learned decisions distribute across the semantic action bank. PatchTST policies favor amplitude corrections (60–81% of edits), iTransformer policies favor level shifts (59–64%), MixLinear on ETTh1 relies on local amplitude (66%), and MixLinear on ETTm1 splits almost evenly between amplitude and level. The contextual-versus-fixed comparison in Section 4.4 measures the benefit of this input-dependent selection; Figure 5 reports action-level outcomes.

The frequency of an edit family is not its contribution to accuracy: Figure 5 reports the signed MSE outcomes by family. These outcome distributions, together with the trajectory cases in Figure 10, connect action use to the endpoint improvements in Section 4.

B.9 Endpoint-wise structural profiles

The profile in Figure 12 compares how often each variant improves each endpoint; Table 7 gives the corresponding improved, tied, and worsened counts for the validation-selected variant.

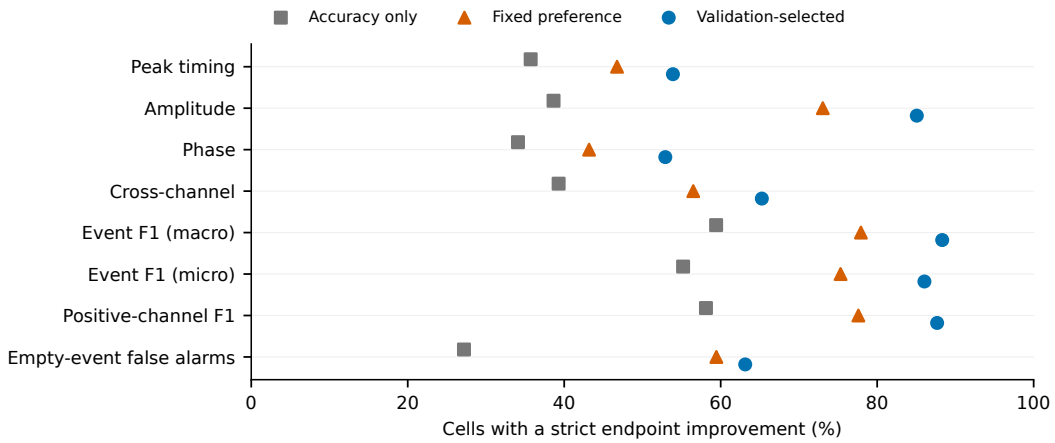

Figure 12: Frequency of strict structural improvement across the matched cells. Shape and color distinguish the three post-training variants. The denominator is 308 for the first seven endpoints and 217 for empty-event false alarms. Counts and ties are given in Appendix Table 7.

B.10 Seed dispersion and control costs

For each fixed dataset–backbone–horizon setting g, let d_{gs} be the paired error change for checkpoint seed s. Tables 1 and 2 report

$$s_\Delta = \sqrt{\frac{\sum_g \sum_s (d_{gs} - \bar{d}_g)^2}{\sum_g (n_g - 1)}}.$$

Only settings with at least two seeds contribute to the pooled within-setting SD. The main cohort has 65 settings with 1–10 checkpoint seeds each, yielding 243 residual degrees of freedom after between-setting differences are removed. The reported means retain their original checkpoint weights, while the pooled SD summarizes within-setting checkpoint variation. Structural counts in Table 3 use the same pooling formula; percentage-error means and SDs are normalized by the cohort mean frozen error.

Table 11: Control costs corresponding to Table 3. Means over 12 cells; time covers five candidate runs. Policy timing excludes cache preparation.

Control	Trainable parameters	Time (s)
Full method (retrained)	33,075	212
Accuracy-only, 5 restarts	33,075	185
No local edits	29,979	177
No timing edits	30,237	216
Random library, seed 0	33,075	199
Random library, seed 1	33,075	194
Random library, seed 2	33,075	216
Sampled PG, K=1	33,075	241
Sampled PG, K=4	33,075	215
Sampled PG, K=16	33,075	209
Affine calibration	14	63
Continuous MLP head	33,042	138
Backbone MSE fine-tuning	1.8M	460

C Learning decisions from complete action feedback

From the semantic action interface to its learning rule. Each training window–channel pair supplies the full reward vector for all 51 semantic actions, combining full-horizon MSE with the specified additive structural components. This one-step, full-information RL interface can train different decision rules: imitate the hindsight-best action, fit every action's reward, maximize

expected reward through policy probabilities, or match a soft reward distribution. At deployment, each rule selects from the same semantic actions using history and the reference forecast, without future outcomes.

Matched protocol. The 12 cells from Section 4.3 each use three paired initialization/order seeds. Objectives share features, the 51-output network, the semantic action bank, full splits, 24 epochs, Adam, cosine scheduling, and batch size 64. Separately fixing $\lambda = 0$ and 0.1, we search learning rates $\{10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}\}$. A training-only global reward RMS shared across objectives conditions optimization without changing action ordering or the expected-reward optimum. This matched search is distinct from Table 3.

Learning objectives. Write $\widetilde{r}_a$ for the scaled reward at one window and channel. Alongside Equation 5, we train

$$\mathcal{L}_{\mathrm{CE}} = -\log \pi_\phi(a^* \mid s), \quad a^* = \arg\max_a \widetilde{r}_a, \tag{24}$$

$$\mathcal{L}_Q = A^{-1} \sum_a (q_\phi(s,a) - \widetilde{r}_a)^2, \tag{25}$$

$$\mathcal{L}_{\mathrm{soft}} = -\sum_a p_a^* \log \pi_\phi(a \mid s), \quad p_a^* = \frac{\exp(\widetilde{r}_a/\tau)}{\sum_b \exp(\widetilde{r}_b/\tau)}. \tag{26}$$

Losses sum over channels and average over windows. Hard-label ties use the first action index, giving identity on all-zero ties. Soft-target CE has the same policy gradients as forward KL from p^* to π_ϕ. Deployment takes the highest policy score, or the highest predicted value for Q regression. The population target of squared value regression is $Q^*(s,a) = \mathbb{E}[\widetilde{r}_a \mid s]$. Its greedy decision maximizes estimated conditional expected reward. By contrast, hard-label CE targets the conditional distribution of hindsight winners. The most probable winner and the highest-expected-reward action need not coincide. Direct policy optimization uses the same realized rewards as value learning but learns action probabilities without an explicit value-fitting step.

Selection and dispersion. All objectives use the same validation selector: epochs and learning rates minimize validation MSE subject to both accuracy gates, while structural metrics provide the reported endpoint profile. Ties favor earlier epochs and configuration order; the identity action is returned when no candidate meets the accuracy gates. Fixed $\tau = 1$ uses five configurations, while tuned $\tau \in \{1, 0.1, 0.5\}$ uses 15 under the same selector. Table 12 reports all 12 checkpoints with equal weight, with SD summarizing initialization and minibatch-order variation across three policy seeds.

Table 12: Learning-rule controls within the fixed semantic action bank, on 12 checkpoints with three policy seeds each. MSE/MAE are percentage reductions in cohort-mean error (positive is better); Imp./Deg. count improved/degraded structural metrics. Mean ± SD is over three policy-seed cohort aggregates. Tuned soft targets receive 15 configurations; other rows receive five.

λ	Objective	MSE ↑	MAE ↑	Imp.↑	Deg.↓
0	Expected reward	1.15 ± 0.04	0.66 ± 0.03	2.81 ± 0.17	3.58 ± 0.08
	Best-action CE	0.21 ± 0.04	0.56 ± 0.02	3.50 ± 0.46	3.03 ± 0.59
	Q regression	1.46 ± 0.01	0.40 ± 0.02	2.08 ± 0.25	4.50 ± 0.22
	Soft, $\tau = 1$	0.61 ± 0.14	0.17 ± 0.06	2.28 ± 0.17	3.89 ± 0.38
	Soft, tuned τ	0.79 ± 0.13	0.44 ± 0.03	2.50 ± 0.14	4.11 ± 0.13
0.1	Expected reward	0.62 ± 0.07	0.39 ± 0.07	5.33 ± 0.36	2.11 ± 0.46
	Best-action CE	-0.33 ± 0.11	0.27 ± 0.01	4.03 ± 0.13	2.64 ± 0.13
	Q regression	1.42 ± 0.05	0.33 ± 0.05	3.81 ± 0.35	3.50 ± 0.38
	Soft, $\tau = 1$	0.52 ± 0.13	0.12 ± 0.05	3.39 ± 0.13	2.83 ± 0.30
	Soft, tuned τ	0.63 ± 0.17	0.41 ± 0.07	3.56 ± 0.05	3.11 ± 0.05

Reward information beyond winner identity. Winner labels preserve the identity of each hindsight-best action but discard the reward differences between actions. Expected-reward optimization exceeds CE in MSE on 30/36 paired runs at $\lambda = 0$ and 32/36 at $\lambda = 0.1$, or 11/12 checkpoints after averaging policy seeds at either coefficient. The corresponding MAE counts are

21/36 and 23/36 runs, or 7/12 and 8/12 checkpoints. Q regression also exceeds CE's aggregate MSE reduction at both coefficients. Thus both full-reward objectives obtain point-accuracy information that winner imitation leaves unused.

How the shared feedback is used. Retaining the full reward vector does not make the objectives interchangeable. Q regression yields the larger MSE reductions, whereas direct expected-reward optimization at $\lambda = 0.1$ improves more structural endpoints and degrades fewer. Tuned soft targets attain similar point accuracy but improve fewer endpoints. The comparison separates the value of full outcome feedback from the effect of fitting values, matching preferences, or optimizing the policy's expected reward directly.

Reward effects without structural selection. The structural reward changes what the direct policy selects under a shared accuracy-only validation selector. Moving from $\lambda = 0$ to 0.1 changes its mean improved/degraded endpoint counts from 2.81/3.58 to 5.33/2.11 while retaining MSE and MAE reductions. The comparison attributes the structural shift to the reward rather than cross-coefficient structural ranking; it complements Table 2.

D ACTION CAPACITY AND THE HINDSIGHT ORACLE GAP

Two distinct bounds. The oracle comparison separates the capacity of the fixed semantic action bank from the performance of history-conditioned decisions. After policy selection, the MSE oracle maximizes realized full-horizon MSE reduction per window/channel; identity ensures nonnegative gain. The combined-reward oracle maximizes r^{λ}. Both choose from the same 51 semantic actions but use future outcomes, so neither is deployable. Training-fitted component scales, support denominators, and event-F_1 linearization constants remain fixed. All eight endpoints are recomputed on assembled forecasts. The combined-reward oracle bounds additive reward, not nonlinear metrics such as pooled F_1.

For mean realized combined reward R_π, regret is $R_{\text{oracle}} - R_\pi$. MSE regret is defined analogously using MSE reduction. Gain capture divides the aggregate policy MSE reduction by the aggregate oracle reduction, rather than averaging window-level ratios. Table 13 uses $\lambda = 0.1$. Fixed global and per-channel actions maximize training reward; uniform random performance is the exact expectation over candidate losses, not the loss of an averaged forecast.

Table 13: Oracle and decision comparisons at $\lambda = 0.1$. Error reductions are percentages relative to the frozen cohort mean. G_{MSE} and G_r are MSE and combined-reward regrets in 10^{-3} units. Learned rows show mean $\pm$ SD across three policy-seed aggregates; fixed references have no policy-seed variation. The two oracle rows use future outcomes.

Decision rule	MSE ↑	MAE ↑	G_{MSE} ↓	G_r ↓
Identity	0.00	0.00	50.62	56.07
Fixed global	−0.93	−0.30	54.00	59.05
Fixed per channel	−0.46	0.02	52.30	57.33
Uniform random (exact)	−12.17	−5.00	95.00	101.36
Expected reward	0.62 ± 0.07	0.39 ± 0.07	48.36 ± 0.27	53.38 ± 0.42
Best-action CE	-0.33 ± 0.11	0.27 ± 0.01	51.82 ± 0.40	57.41 ± 0.28
Q regression	1.42 ± 0.05	0.33 ± 0.05	45.44 ± 0.18	50.45 ± 0.15
Soft, tuned τ	0.63 ± 0.17	0.41 ± 0.07	48.34 ± 0.63	54.41 ± 0.73
MSE oracle	13.89	7.23	0.00	4.08
Reward oracle	13.35	6.62	1.96	0.00

Capacity versus predictability. The frozen cohort means are MSE 0.364563 and MAE 0.386628. The MSE oracle attains 0.313941 and 0.358666, a 13.89% MSE reduction. With accuracy-only reward, direct policy optimization captures 8.27% of this hindsight gain, compared with 10.50% for Q regression, 1.51% for CE, and 5.72% for tuned soft targets. The semantic action bank's realized capacity therefore exceeds the gains captured by history-conditioned policies, separating action-bank capacity from deployable policy capture.

E SETTING-LEVEL ACCURACY AND SEED VARIATION

The aggregate result covers settings with different numbers of available checkpoints. Table 14 therefore reports paired changes for all 65 dataset–backbone–horizon settings, each averaged over its available seeds. For settings with multiple checkpoints, the table also reports sample standard deviation across checkpoint seeds; single-seed settings report their paired mean directly. Comparisons follow the six-decimal convention in Appendix A.6.

Table 14: All evaluated dataset–backbone–horizon settings. Negative mean changes favor post-training. n counts checkpoint seeds; $\pm$ denotes sample standard deviation.

Dataset	Backbone	H	n	ΔMSE	ΔMAE
ECL	MixLinear	96	5	-0.007677 ± 0.000550	-0.003626 ± 0.000364
ECL	MixLinear	192	5	-0.001761 ± 0.000896	-0.000865 ± 0.000239
ECL	PatchTST	96	3	-0.006337 ± 0.001851	-0.012002 ± 0.001987
ECL	iTransformer	96	5	-0.000061 ± 0.000011	0.000057 ± 0.000045
ECL	iTransformer	720	5	-0.002380 ± 0.001592	-0.000961 ± 0.000612
ETTh1	MixLinear	96	3	-0.001863 ± 0.002924	-0.000729 ± 0.001194
ETTh1	PaiFilter	96	3	0.002493 ± 0.004941	0.003525 ± 0.004791
ETTh1	PaiFilter	192	5	-0.004422 ± 0.003279	-0.001052 ± 0.001485
ETTh1	PatchTST	96	3	0.003608 ± 0.000581	0.001081 ± 0.000421
ETTh1	PatchTST	192	5	-0.000249 ± 0.002757	-0.000462 ± 0.000848
ETTh1	PatchTST	336	5	0.012991 ± 0.007100	0.001632 ± 0.001401
ETTh1	iTransformer	96	10	0.001376 ± 0.001357	0.000816 ± 0.001029
ETTh1	iTransformer	192	5	-0.003289 ± 0.001668	-0.001746 ± 0.000893
ETTh1	iTransformer	336	5	-0.008233 ± 0.004526	-0.004613 ± 0.002778
ETTh1	iTransformer	720	5	-0.021417 ± 0.007812	-0.012376 ± 0.004041
ETTh2	MixLinear	96	3	-0.005325 ± 0.000707	-0.002590 ± 0.000322
ETTh2	PaiFilter	96	5	-0.003282 ± 0.001461	-0.002672 ± 0.001751
ETTh2	PaiFilter	192	5	-0.001728 ± 0.000970	-0.001806 ± 0.000491
ETTh2	PaiFilter	336	1	-0.002219	-0.002740
ETTh2	PatchTST	96	5	-0.004185 ± 0.000806	-0.003335 ± 0.000805
ETTh2	iTransformer	96	5	-0.000930 ± 0.000560	-0.000948 ± 0.000461
ETTh2	iTransformer	192	5	-0.000524 ± 0.001350	-0.001129 ± 0.000620
ETTh2	iTransformer	336	5	-0.001796 ± 0.001420	-0.002216 ± 0.001331
ETTh2	iTransformer	720	5	-0.001651 ± 0.002286	-0.001301 ± 0.001822
ETTm1	MixLinear	96	3	-0.012328 ± 0.003124	-0.006474 ± 0.002335
ETTm1	PaiFilter	96	10	-0.004378 ± 0.002130	-0.002114 ± 0.001032
ETTm1	PatchTST	96	3	-0.003791 ± 0.001323	-0.003310 ± 0.000661
ETTm1	PatchTST	336	5	-0.002388 ± 0.004378	-0.002214 ± 0.001633
ETTm1	iTransformer	96	5	-0.003287 ± 0.002472	-0.001567 ± 0.001319
ETTm1	iTransformer	192	5	-0.002725 ± 0.002015	-0.000683 ± 0.000933
ETTm1	iTransformer	336	5	-0.007454 ± 0.003313	-0.003160 ± 0.001695
ETTm1	iTransformer	720	5	-0.006975 ± 0.002463	-0.002150 ± 0.001205
ETTm2	MixLinear	96	4	-0.002913 ± 0.000148	-0.002827 ± 0.000172
ETTm2	PaiFilter	96	5	-0.001203 ± 0.000593	-0.001424 ± 0.000494
ETTm2	PaiFilter	192	5	-0.000859 ± 0.000590	-0.000917 ± 0.000532
ETTm2	PatchTST	96	8	-0.000880 ± 0.000357	-0.001508 ± 0.000397
ETTm2	iTransformer	96	5	-0.001927 ± 0.000478	-0.001872 ± 0.000404
ETTm2	iTransformer	192	5	-0.001811 ± 0.000535	-0.001445 ± 0.000511
ETTm2	iTransformer	336	5	-0.002510 ± 0.000280	-0.001777 ± 0.000177
ETTm2	iTransformer	720	5	-0.003048 ± 0.000775	-0.002118 ± 0.000614
Solar	MixLinear	96	5	-0.041159 ± 0.001455	-0.024150 ± 0.001785
Solar	MixLinear	192	1	-0.031258	-0.014711
Solar	PaiFilter	96	5	-0.000157 ± 0.001602	0.002009 ± 0.002021
Solar	PaiFilter	192	5	0.002297 ± 0.001708	0.000418 ± 0.001106
Solar	PaiFilter	336	5	0.000482 ± 0.001477	0.000708 ± 0.000733
Solar	PaiFilter	720	5	0.000000 ± 0.000000	0.000000 ± 0.000000
Solar	PatchTST	96	5	-0.004591 ± 0.001613	-0.012718 ± 0.005991
Solar	PatchTST	192	5	-0.001772 ± 0.001737	-0.010698 ± 0.003046
Solar	PatchTST	336	5	-0.000962 ± 0.002025	-0.008539 ± 0.001337
Solar	PatchTST	720	5	0.000015 ± 0.004018	-0.005546 ± 0.004312

Table 14 continued

Dataset	Backbone	H	n	ΔMSE	ΔMAE
Solar	iTransformer	96	5	-0.003890 ± 0.001537	0.000299 ± 0.002406
Solar	iTransformer	192	5	-0.005493 ± 0.001274	0.001125 ± 0.002293
Solar	iTransformer	336	5	-0.001177 ± 0.001046	-0.001141 ± 0.000960
Solar	iTransformer	720	5	-0.000654 ± 0.000886	-0.000700 ± 0.000709
Traffic	MixLinear	96	3	-0.022252 ± 0.006268	-0.009721 ± 0.002781
Traffic	PatchTST	96	4	-0.002542 ± 0.001732	-0.008827 ± 0.000453
Traffic	iTransformer	96	2	-0.002241 ± 0.000171	-0.001471 ± 0.000201
Weather	MixLinear	96	3	-0.007165 ± 0.000051	-0.005168 ± 0.000148
Weather	PatchTST	96	5	-0.002394 ± 0.000546	-0.001709 ± 0.000295
Weather	PatchTST	336	5	-0.003305 ± 0.000868	-0.001537 ± 0.000327
Weather	PatchTST	720	1	-0.004321	-0.001180
Weather	iTransformer	96	5	-0.003175 ± 0.000807	-0.001654 ± 0.000723
Weather	iTransformer	192	5	-0.003954 ± 0.000771	-0.001709 ± 0.000474
Weather	iTransformer	336	10	-0.004816 ± 0.000678	-0.002144 ± 0.000685
Weather	iTransformer	720	5	-0.004922 ± 0.000736	-0.001860 ± 0.000586