ONART: ONLINE ADAPTATION OF RECTIFIED FLOW FOR PROBABILISTIC TIME-SERIES FORECASTING WITH FEEDBACK DELAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Probabilistic time-series forecasting is essential for decisions under uncertainty, yet existing generative forecasters are predominantly fixed after offline training, while online methods largely adapt point predictors without updating predictive uncertainty. We address the resulting problem of adapting an entire predictive distribution from sparse, delayed feedback without destabilizing its learned transport. We introduce **OnART**, an online adaptation framework that turns Rectified Flow into an efficient delayed-feedback probabilistic forecaster. Offline re-flow straightens the conditional transport and enables one-step generation, limiting the trajectory interference induced by online updates. At deployment, a newly observed residual provides a new endpoint but no source from which to update the learned transport. OnART selects, for each variable, the Gaussian source paired with the nearest forecast-time residual endpoint. This choice exactly minimizes the change from the cached displacement target to the new re-flow displacement target among the stored candidates; online training then intentionally moves the selected source toward the new endpoint. Generated replay anchors the cached forecast-time mappings while this update is applied. The point forecaster learns from all causally available values, and the rectified velocity model learns from the assigned pair and generated replay after the feedback delay. Across seven multivariate benchmarks, OnART consistently improves probabilistic forecasting performance while preserving competitive point accuracy. These results establish minimal-shift source assignment and replay as effective mechanisms for online adaptation of probabilistic forecasts. Our code is available at https://anonymous.4open.science/r/OnART-ICLRSubmission-C277

1 INTRODUCTION

Probabilistic time-series forecasting aims to characterize the range of plausible future outcomes rather than predict only a conditional mean (Gneiting & Katzfuss, 2014). This capability is essential for decision-making under uncertainty: inventory planning must account for demand variability, energy systems must anticipate fluctuating renewable generation, and operations systems must quantify the risk of extreme deviations. In many such applications, however, the data-generating process is nonstationary. Seasonal patterns, user behavior, operating conditions, and external environments may change over time, causing both point forecasts and predictive distributions learned from historical data to become misaligned with the current regime (Gama et al., 2014).

This setting requires an online forecaster to update both its central prediction and the shape of its predictive uncertainty when a new target becomes available. At the same time, feedback is delayed by the forecasting horizon, and each update may be based on only one newly observed future. For example, consider short-term renewable-energy forecasting. After weather regime shifts, wind volatility rises while mean output stays stable. Online point forecasters may adapt the mean but ignore rising uncertainty, issuing narrow intervals that underestimate shortfall/surplus risks. Practical online probabilistic forecasting therefore requires a mechanism that updates the full predictive distribution efficiently under delayed feedback while preserving useful previously learned structure.

Adapting a transport-based generative forecaster introduces a fundamental difficulty absent from ordinary point-model updates. Flow Matching (Lipman et al., 2023) learns a velocity field that transports samples from a simple source distribution to the target residual distribution. As illustrated in

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.

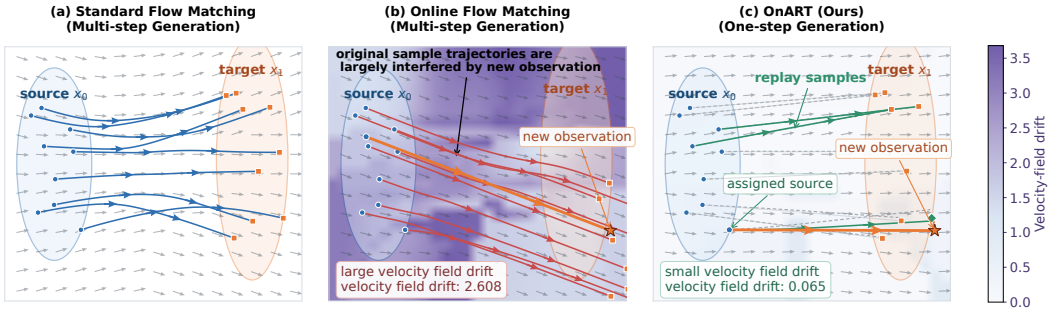


Figure 1: Toy illustration of the motivation. A newly observed residual provides a new endpoint but no prescribed source for updating the transport. Random pairing can require a large change from the existing displacement target; Online Re-Flow instead reuses the source paired with the nearest generated endpoint, minimizing this target shift among the cached candidates. The background color visualizes the pointwise change of the toy velocity field after one update.

Fig. 1(a), standard flow matching trains the velocity field along interpolation paths between targets and randomly sampled Gaussian sources. During deployment, however, a newly observed residual reveals only its target endpoint, while the transport must be updated from some Gaussian source. As Fig. 1(b) illustrates, pairing that endpoint with an independent source can impose a displacement target unrelated to the model's forecast-time mappings, perturb overlapping trajectories, and drive harmful velocity-field drift. The defining challenge is therefore not to recover a source that the current transport already maps to the new endpoint; online learning is intended to change that mapping. Instead, the update should reuse a forecast-time anchor that requires the smallest correction toward the new residual.

We solve this endpoint-only adaptation problem with an online rectified-flow mechanism. As Fig. 1(c) shows, OnART retrieves, for each variable, the generated residual trajectory closest to the newly observed residual and inherits its paired Gaussian trajectory. For any cached pair, the change between its old displacement target and the displacement target induced by the new residual is exactly the distance between the cached and new endpoints. Nearest-endpoint assignment therefore selects the forecast-time anchor requiring the smallest supervision shift. Replaying the remaining generated pairs constrains changes to the other forecast-time mappings, so the model can move the selected source toward the new endpoint without discarding its learned local structure.

This mechanism forms **OnART**, a unified framework for delayed-feedback probabilistic forecasting. OnART decomposes each forecast into a deterministic point prediction and a stochastic residual, following D3U (Li et al., 2025), and models the residual with a codebook-conditioned Rectified Flow (Liu et al., 2023). At each timestep, OnART issues the current probabilistic forecast and then evaluates the latest $H + 1$ overlapping windows with a causal mask using the same pre-update PatchTST state. This forward snapshot supplies the point loss, the newly observed residual, and its condition; PatchTST and the velocity model are subsequently updated in the same post-prediction phase. The indexing is strictly causal: no forecast accesses its own target, and both updated models are first deployed at the next timestep.

Limited feedback also makes a continuously evolving condition representation prone to drift and overfitting. OnART therefore maps PatchTST representations to a finite vector-quantized codebook that shares condition prototypes across related forecasting contexts. The codebook remains fixed online, and each forecast caches its quantized condition with its source–endpoint pairs.

Our contributions are:

- We identify and formalize online probabilistic forecasting as endpoint-only transport adaptation under delayed feedback: each observed future supplies a new residual endpoint, and the model must choose a source anchor from which to update the transport. **OnART** jointly adapts the point forecast and residual predictive distribution using strictly causal feedback.
- We introduce Online Re-Flow with channel-wise proximity-based source assignment. It selects the cached source whose displacement supervision requires the smallest correction toward each newly observed residual, while generated replay anchors the cached forecast-time mappings. Offline re-flow makes both forecasting and adaptation compatible with one-step generation.
- We prove that channel-wise proximity assignment minimizes the displacement-target shift over all channel-wise combinations of replay candidates and minimizes a uniform upper

bound on the resulting gradient perturbation. We also provide a PAC–Bayes generalization bound for codebook-conditioned prediction. Component-wise ablations verify that both mechanisms materially improve probabilistic accuracy.

- On seven multivariate benchmarks, OnART achieves the best $\mathrm{CRPS}_{\mathrm{sum}}$ in every dataset, improves macro-averaged $\mathrm{CRPS}_{\mathrm{sum}}$ by 10.6% over the strongest offline probabilistic baseline, and retains competitive point accuracy with one-step probabilistic generation.

We provide a detailed review of probabilistic forecasting, online forecasting, and generative time-series models in Appendix A.

2 METHODOLOGY

2.1 PROBLEM FORMULATION

We formulate delayed-feedback probabilistic forecasting for a multivariate time-series stream with C variables. Each forecasting sample is written as

$$\mathbf{s}_t = [\mathbf{l}_t; \mathbf{y}_t] \in \mathbb{R}^{(L+H)\times C}, \tag{1}$$

where $\mathbf{l}_t \in \mathbb{R}^{L\times C}$, $\mathbf{y}_t \in \mathbb{R}^{H\times C}$ denote the look-back sequence and its subsequent ground-truth forecasting window, respectively. Here, L is the look-back length and H is the forecasting horizon.

The complete target window $\mathbf{y}_t$ becomes available at time $t+H$. Thus, at time t, feedback includes $\mathbf{y}_{t-H}$ and causally available prefixes of newer overlapping windows, but not $\mathbf{y}_t$. OnART follows an explicit predict–then–update protocol: it first issues P_t without accessing $\mathbf{y}_t$, then processes the available delayed feedback, and deploys the updated point and velocity models from timestep $t+1$ onward. We denote the predictive ensemble size by K and the number of stored pairs used for source assignment and replay by $R \le K$.

2.2 SOLUTION FRAMEWORK

Figure 2 presents OnART's two-stage solution: *offline pretraining* establishes an efficient conditional transport, and *online adaptation* updates both location and uncertainty from delayed observations. Point forecasting, conditional flow matching, and offline re-flow build the pretrained model. During deployment, the PatchTST point forecaster and rectified velocity model learn exclusively from causally available observations; the target of the current probabilistic forecast remains unavailable by construction.

The point forecaster contains a time-series backbone b_ϕ and a decoder d_ψ. Given a look-back sequence $\mathbf{l}_t$, the backbone produces a channel-wise condition representation $\mathbf{c}_t = b_\phi(\mathbf{l}_t) \in \mathbb{R}^{C\times D}$, where D denotes the condition embedding dimension. The decoder produces the point forecast $\mathbf{y}_{\mathrm{Point},t} = d_\psi(\mathbf{c}_t) \in \mathbb{R}^{H\times C}$.

The uncertainty remaining after point forecasting is represented by the residual $\mathbf{x}_{1,t} = \mathbf{y}_t - \mathbf{y}_{\mathrm{Point},t} \in \mathbb{R}^{H\times C}$. An online conditional rectified flow model generates residual forecast samples from Gaussian starting points. For the k-th sample, $\mathbf{y}^{(k)}_{\mathrm{Res},t} = \hat{\mathbf{x}}^{(k)}_t$, and the final probabilistic forecast is

$$\hat{\mathbf{y}}^{(k)}_t = \mathbf{y}_{\mathrm{Point},t} + \mathbf{y}^{(k)}_{\mathrm{Res},t}, \tag{2}$$

where $k = 1, \ldots, K$.

This decomposition assigns complementary roles to the two components: the point forecaster captures central structure, and the rectified velocity model explicitly represents the remaining conditional variation. It also isolates the distribution that must be adapted online from changes already absorbed by the point forecast.

2.3 OFFLINE PRETRAINING

Offline pretraining consists of point-forecaster training, conditional flow matching, and offline re-flow, as illustrated in the upper part of Fig. 2.

Step 1: Point-forecasting. Given a historical sample $\mathbf{s} = [\mathbf{l}; \mathbf{y}] \in \mathbb{R}^{(L+H)\times C}$, the time-series backbone and decoder produce a condition representation $\mathbf{c} \in \mathbb{R}^{C\times D}$ and a point forecast $\mathbf{y}_{\mathrm{Point}} \in \mathbb{R}^{H\times C}$. They are jointly optimized using

$$\mathcal{L}_{\mathrm{Point}} = \|\mathbf{y}_{\mathrm{Point}} - \mathbf{y}\|^2. \tag{3}$$

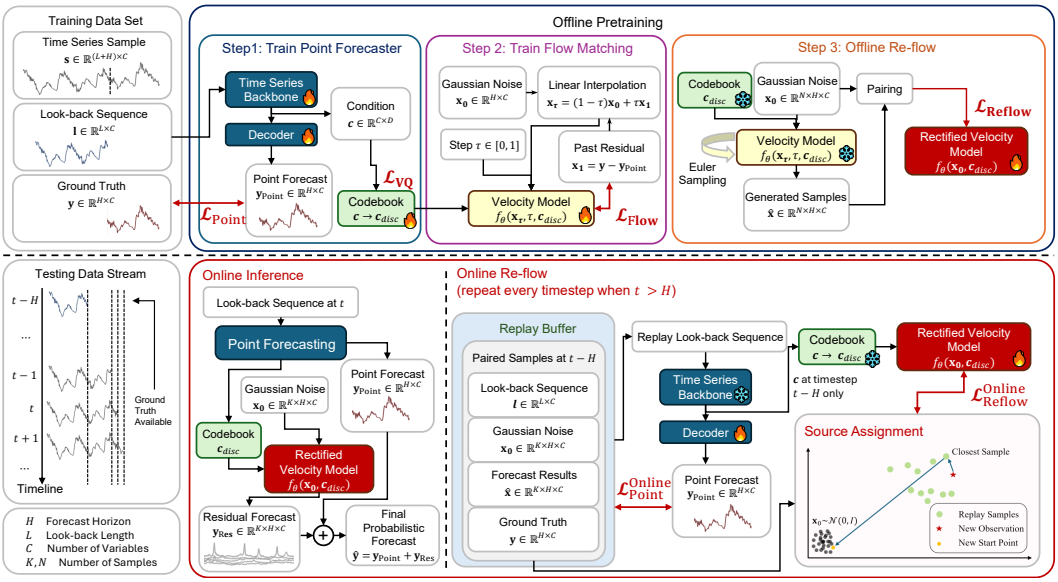


Figure 2: OnART jointly adapts point predictions and residual uncertainty under delayed feedback. Point forecasting, conditional flow matching, and offline re-flow establish the pretrained model; strictly causal online updates then adapt the point forecaster and rectified velocity model without exposing the current forecast target.

The remaining forecasting uncertainty is represented by the residual $\mathbf{x}_1 = \mathbf{y} - \mathbf{y}_{\text{Point}}$, which serves as the target distribution of the subsequent generative model.

Step 2: Conditional flow matching. The continuous condition $\mathbf{c}$ is mapped by a learnable codebook to the discrete condition $\mathbf{c}_{\text{disc}}$, which is used to condition the velocity model. Given Gaussian noise $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and residual target $\mathbf{x}_1$, we construct

$$\mathbf{x}_\tau = (1-\tau)\mathbf{x}_0 + \tau\mathbf{x}_1, \qquad \tau \sim \mathcal{U}[0,1], \tag{4}$$

and train the conditional velocity model by

$$\mathcal{L}_{\text{Flow}} = \mathbb{E}\left[\left\| f_\theta(\mathbf{x}_\tau, \tau, \mathbf{c}_{\text{disc}}) - (\mathbf{x}_1 - \mathbf{x}_0)\right\|^2\right] \tag{5}$$

The codebook is jointly learned using the standard vector-quantization loss $\mathcal{L}_{\text{VQ}}$ (van den Oord et al., 2017), while the pretrained point forecaster remains fixed.

We theoretically characterize the learned codebook with a **PAC–Bayes generalization bound** for a bounded, parameter-Lipschitz codebook-conditioned surrogate in Appendix C. The bound makes the regularization tradeoff explicit by relating test risk to training risk, codebook complexity, and derandomization terms.

Step 3: Offline Re-Flow. To convert the learned flow into a transport suitable for efficient online use, we draw N Gaussian starting points for each historical condition and use the pretrained flow-matching model to generate their corresponding residual endpoints $\hat{\mathbf{x}}$. The generated pairs $(\mathbf{x}_0, \hat{\mathbf{x}})$ are then used to retrain the velocity model:

$$\mathcal{L}_{\text{Reflow}} = \mathbb{E}\left[\left\| f_\theta((1-\tau)\mathbf{x}_0 + \tau\hat{\mathbf{x}}, \tau, \mathbf{c}_{\text{disc}}) - (\hat{\mathbf{x}} - \mathbf{x}_0)\right\|^2\right]. \tag{6}$$

Offline re-flow rectifies the conditional transport and yields one-step Euler generation. This step is essential to OnART's deployment regime: it replaces iterative sampling trajectories with direct source–endpoint supervision that can be stored, assigned, and replayed online.

2.4 Online Inference

At timestep t, the model first processes the current look-back sequence:

$$\mathbf{c}_t = b_\phi(\mathbf{l}_t), \qquad \mathbf{c}_{\text{disc},t} = Q_{\mathcal{E}}(\mathbf{c}_t), \tag{7}$$

and produces the point forecast $\mathbf{y}_{\text{Point},t} = d_\psi(\mathbf{c}_t) \in \mathbb{R}^{H\times C}$.

We then draw K Gaussian starting points $\mathbf{x}_{0,t}^{(k)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $k = 1, \ldots, K$, and generate residual forecasts using the one-step rectified velocity model:

$$\mathbf{y}_{\mathrm{Res},t}^{(k)} = \hat{\mathbf{x}}_t^{(k)} = \mathbf{x}_{0,t}^{(k)} + f_\theta \left(\mathbf{x}_{0,t}^{(k)}, 0, \mathbf{c}_{\mathrm{disc},t} \right). \tag{8}$$

The final probabilistic samples are obtained through Eq. 2. OnART stores the forecast-time source–endpoint anchors generated at timestep t in a replay buffer:

$$\mathcal{B}_t = \left\{ \left(\mathbf{c}_{\mathrm{disc},i}, \{\mathbf{x}_{0,i}^{(k)}, \hat{\mathbf{x}}_i^{(k)}\}_{k=1}^{K} \right) \right\}_{i \in \mathcal{I}_t}, \tag{9}$$

where $\mathcal{I}_t$ contains the most recent $H+1$ forecast indices. The implementation stores the K Gaussian sources, their generated residual endpoints, and the quantized condition used for each forecast. Once the target for index $t - H$ is complete, this record supplies both the source-assignment candidates and the replay pairs, ensuring that the update is grounded in the transport that produced the original probabilistic forecast.

2.5 Online Re-Flow

Online Re-Flow is executed at every timestep $t \geq H$ after the probabilistic forecast for timestep t has been issued. The point forecaster and velocity model are updated in the same post-prediction phase, and both updates take effect at timestep $t + 1$.

In the PatchTST implementation, OnART optimizes the complete point forecaster and rectified velocity model online while freezing the codebook. This division preserves stable condition prototypes without preventing the forecasting and uncertainty models from adapting. The complete algorithm appears in Appendix E.

2.5.1 Causally Masked Point-Forecaster Update

Because the forecasting windows issued at recent timesteps are only partially observed, the point forecaster is trained using a causal mask. The update batch contains the $H + 1$ stride-one windows $\mathcal{P}_t = \{t - H, \ldots, t\}$. Let the h-th future position of the window issued at timestep i correspond to absolute timestep $i + h$. At current timestep t, define

$$m_{i,h}^{(t)} = \mathbb{I}[i + h \leq t], \qquad h = 1, \ldots, H. \tag{10}$$

The online point loss is

$$\mathcal{L}_{\mathrm{Point},t}^{\mathrm{online}} = \frac{1}{C(H+1)H} \sum_{i \in \mathcal{P}_t} \sum_{h=1}^{H} \sum_{j=1}^{C} m_{i,h}^{(t)} \left(y_{i,h,j} - y_{\mathrm{Point},i,h,j} \right)^2. \tag{11}$$

Here $y_{\mathrm{Point},i,h,j}$ is produced by the current point forecaster for every retained window. The implementation multiplies the elementwise squared error by the triangular mask and then takes the mean over the complete $(H + 1) \times H \times C$ tensor. With stride-one windows, an observed timestamp contributes through each overlapping window in which it appears, while masked future entries contribute zero.

Equation 11 updates the PatchTST parameters using every available overlapping observation while excluding all unobserved future entries from the gradient.

2.5.2 Proximity-based Source Assignment

Online flow adaptation observes a new residual endpoint without its source. OnART therefore selects the cached source–endpoint anchor whose displacement target requires the smallest correction, without assuming that its source remains exactly compatible with the current transport. After issuing the forecast at timestep t, OnART performs one masked forward pass on the feedback windows before either optimizer step and converts the complete target for the oldest window to the residual

$$\mathbf{x}_{1,t-H}^{\mathrm{new}} = \mathbf{y}_{t-H} - \mathbf{y}_{\mathrm{Point},t-H}^{(t)}, \tag{12}$$

where $\mathbf{y}_{\mathrm{Point},t-H}^{(t)}$ is retained from this pre-update pass; its representation is quantized to obtain $\mathbf{c}_{\mathrm{disc},t-H}^{(t)}$. Using the current output rather than the cached prediction makes the accumulated point-model change since timestep $t - H$ part of the endpoint correction. Both online objectives are therefore defined in the same pre-update point-forecaster coordinates; the subsequent point-forecaster step need not preserve them. The point forecaster and velocity model are then updated together and

first deployed at timestep $t+1$, with no post-step re-forward. We retrieve the generated item stored when this window was forecast:

$$\left(\mathbf{c}^{\text{stored}}_{\text{disc},t-H}, \left\{\mathbf{x}^{(k)}_{0,t-H}, \hat{\mathbf{x}}^{(k)}_{1,t-H}\right\}_{k=1}^{K}\right) \in \mathcal{B}_t. \tag{13}$$

The implementation performs source assignment independently for each variable. For variable c, it compares the length-H residual trajectories and selects

$$k_c^\star \in \underset{k\in\{1,\dots,R\}}{\arg\min} \left\|\mathbf{x}^{\text{new}}_{1,t-H,:,c} - \hat{\mathbf{x}}^{(k)}_{1,t-H,:,c}\right\|_2. \tag{14}$$

The assigned source is assembled channel-wise as

$$\tilde{\mathbf{x}}_{0,t-H,:,c} = \mathbf{x}^{(k_c^\star)}_{0,t-H,:,c}, \qquad c = 1,\dots,C. \tag{15}$$

The rectified velocity model used in OnART is channel-separable. For a residual state $\mathbf{x} \in \mathbb{R}^{H\times C}$ and channel-wise condition $\mathbf{c}_{\text{disc}}$, its output satisfies

$$\left[f_\theta\left(\mathbf{x}, \tau, \mathbf{c}_{\text{disc}}\right)\right]_{:,c} = v_{\theta,c}\left(\mathbf{x}_{:,c}, \tau, \mathbf{c}_{\text{disc},c}\right), \tag{16}$$

so channel c is invariant to the sources assigned to other channels, and the assembled source need not coincide with a cached multivariate Monte-Carlo sample. Because Online Re-Flow intentionally changes the old mapping, assignment instead seeks the forecast-time anchor requiring the smallest change in displacement supervision. The following result establishes this property for the complete $H \times C$ residual target; its proof is provided in Appendix B.

Theorem 1 (Optimality of channel-wise source assignment)**.** *Assume the channel-separable velocity model in Eq. 16. For $\boldsymbol{\kappa} = (\kappa_1,\dots,\kappa_C) \in \mathcal{K} := \{1,\dots,R\}^C$, define $D(\boldsymbol{\kappa}) := [\sum_c \|\mathbf{x}^{\text{new}}_{1,:,c} - \hat{\mathbf{x}}^{(\kappa_c)}_{1,:,c}\|_2^2]^{1/2}$. Because the assembled source is shared, D equals the Frobenius change between the cached and new displacement targets. The nearest-endpoint indices $\boldsymbol{\kappa}^\star = (k_1^\star,\dots,k_C^\star)$ from Eq. 14 globally minimize this shift:*

$$\boldsymbol{\kappa}^\star \in \underset{\boldsymbol{\kappa}\in\mathcal{K}}{\arg\min}\, D(\boldsymbol{\kappa}), \qquad D(\boldsymbol{\kappa}^\star) = \left[\sum_{c=1}^{C} \epsilon_c^2\right]^{1/2}, \quad \epsilon_c := \min_k \left\|\mathbf{x}^{\text{new}}_{1,:,c} - \hat{\mathbf{x}}^{(k)}_{1,:,c}\right\|_2. \tag{17}$$

Let $\ell^{\boldsymbol{\kappa}}_{\text{new}}$ and $\ell^{\boldsymbol{\kappa}}_{\text{anchor}}$ be the mean-squared re-flow losses at the same assembled source and current condition, using the new and cached displacement targets, respectively. If the parameter-Jacobian operator norm is uniformly bounded by B_J at these sources, then

$$\left\|\nabla_\theta\left(\ell^{\boldsymbol{\kappa}^\star}_{\text{new}} - \ell^{\boldsymbol{\kappa}^\star}_{\text{anchor}}\right)\right\|_2 \le \frac{2B_J}{HC}\left[\sum_{c=1}^{C}\epsilon_c^2\right]^{1/2}. \tag{18}$$

Proximity assignment minimizes this uniform bound; averaging with the same R replay losses scales it by $1/(R+1)$.

Theorem 1 compares losses at the current pre-update velocity parameters and retained condition. Its counterfactual anchor loss holds the assembled source and condition fixed, isolating replacement of the forecast-time displacement target by the new target without assuming that the forecast-time model or condition is unchanged. Because $\mathbf{x}_1^{\text{new}}$ is defined relative to the current pre-update point snapshot, each ϵ_c includes the residual-coordinate shift since the original forecast. The theorem thus justifies the source choice for the actual endpoint correction without implying that the correction vanishes or that the selected source already maps to the new endpoint.

2.5.3 Online Re-Flow Loss

The newly assigned pair is $\left(\tilde{\mathbf{x}}_{0,t-H}, \mathbf{x}^{\text{new}}_{1,t-H}, \mathbf{c}^{(t)}_{\text{disc},t-H}\right)$. For compact notation, define the re-flow loss of a paired source and target as

$$\ell_{\text{RF}}(\theta; \mathbf{a}, \mathbf{b}, \mathbf{c}_{\text{disc}}) = \frac{1}{HC}\left\|f_\theta(\mathbf{a}, 0, \mathbf{c}_{\text{disc}}) - (\mathbf{b}-\mathbf{a})\right\|^2. \tag{19}$$

Because online generation uses one Euler step, OnART aligns adaptation with inference by fixing the interpolation time to $\tau = 0$ for every online re-flow update.

To prevent a single delayed observation from overriding the learned residual generator, OnART combines the newly assigned pair with R generated pairs stored for the same forecast. The new pair supplies plasticity: it uses the condition and point output retained from the current pre-update masked forward snapshot and intentionally changes the selected source's displacement target toward

the resulting residual. The generated pairs supply stability: they retain the conditions cached when their endpoints were produced and continue to supervise their forecast-time displacement targets. Thus, the two condition versions serve different roles rather than being assumed identical. The online re-flow objective is the mean over the $R+1$ pairs:

$$\mathcal{L}^{\text{online}}_{\text{Reflow},t} = \frac{1}{R+1}\Bigg[\ell_{\text{RF}}\Big(\theta; \tilde{\mathbf{x}}_{0,t-H}, \mathbf{x}^{\text{new}}_{1,t-H}, \mathbf{c}^{(t)}_{\text{disc},t-H}\Big) + \sum_{k=1}^{R} \ell_{\text{RF}}\Big(\theta; \mathbf{x}^{(k)}_{0,t-H}, \hat{\mathbf{x}}^{(k)}_{1,t-H}, \mathbf{c}^{\text{stored}}_{\text{disc},t-H}\Big)\Bigg]. \tag{20}$$

Equation 20 updates only the rectified velocity parameters θ. In the experiments, $K = R = 100$, so every update changes one selected source toward the newly observed residual while replaying 100 forecast-time displacement targets as anchors.

3 Experiments

3.1 Experimental Setup

Datasets and baselines We conduct a broad evaluation on seven multivariate time-series benchmarks: ETTh2 and ETTm1 (Zhou et al., 2021), Weather (Max Planck Institute for Biogeochemistry, 2020), Exchange, Traffic, and Solar (Lai et al., 2018), and ECL (Trindade, 2015).

We compare with offline probabilistic methods D3U (Li et al., 2025), TMDM (Li et al., 2024), and NsDiff (Ye et al., 2025), as well as online deterministic methods Proceed (Zhao & Shen, 2025), DSOF (Lau et al., 2025), and OneNet (Zhang et al., 2023). We additionally include *Naive Flow Matching* as an online probabilistic baseline. It follows the same delayed-feedback schedule but pairs each newly observed residual with an independently sampled Gaussian source during online flow updates.

Metrics and implementation details. We report MSE, CRPS (Gneiting & Raftery, 2007), and CRPS_{sum} (Rasul et al., 2021a), where the latter evaluates the predictive distribution after aggregating all variables. Lower values indicate better performance. We use PatchTST (Nie et al., 2023) as the backbone. We unify the look-back length of our model and all baselines to 96. The prediction horizons are $H \in \{24, 48, 96\}$ and we report their average performance. The split ratio of every dataset follows Proceed (Zhao & Shen, 2025). The data loader standardizes every series using statistics from its training partition, and metrics are computed in the standardized space. We use $K = 100$ predictive trajectories. For all published baselines, we use the implementations released in their official repositories and verify the dataset partitions, look-back and forecast-horizon configurations, and dataset-level normalization and de-normalization procedures. With these settings aligned, the order-of-magnitude differences for TMDM and NsDiff shown in Table 1 persist when running their official implementations. Full hyperparameter settings are provided in Appendix D.

3.2 Main Results

Table 1 reports results averaged over the three prediction horizons. OnART delivers the strongest probabilistic forecasts in the comparison: it ranks first in CRPS_{sum} on every one of the seven datasets. Against D3U, the strongest offline probabilistic baseline, OnART reduces macro-averaged MSE by 16.0% and CRPS_{sum} by 10.6%. The CRPS_{sum} gains remain substantial across heterogeneous domains, reaching 20.5% on Traffic, 14.7% on ETTm1, 15.2% on Weather, and 14.3% on Exchange. This uniform advantage across datasets provides direct evidence that adapting the residual distribution to the chronological stream is more effective than leaving a pretrained probabilistic model fixed.

The probabilistic gains do not require sacrificing the central forecast. OnART obtains the best MSE on ETTh2, ETTm1, and Weather and the second-best MSE on Exchange and ECL. It outperforms the online deterministic method Proceed on three datasets while additionally producing a full sample-based predictive distribution. Together, these results establish OnART as a strong probabilistic forecaster with competitive point accuracy, rather than an uncertainty model purchased at the expense of mean prediction.

Table 1: OnART achieves the best CRPS_{sum} on all seven datasets. Average forecasting performance over prediction horizons $\{24, 48, 96\}$. The best result is highlighted in bold, and the second-best result is underlined. Full results are listed in Appendix F. The best CRPS_{sum} does not take deterministic methods into consideration.

Dataset	OnART (Ours)		Offline Probabilistic Methods						Online Deterministic Methods		
			D3U		TMDM		NsDiff		Proceed	DSOF	OneNet
	MSE	CRPS_{sum}	MSE	CRPS_{sum}	MSE	CRPS_{sum}	MSE	CRPS_{sum}	MSE	MSE	MSE
ETTh2	**3.6332**	**2.9461**	4.1902	3.1516	10.5700	4.0154	4.8744	3.2786	3.9113	14.0341	4.087
ETTm1	**0.6652**	**1.7030**	1.0069	1.9961	1.2619	2.6319	1.1668	2.4248	0.7795	0.8386	0.9933
Weather	**1.2109**	**4.2545**	1.4161	5.0155	1.9857	6.4540	2.0349	6.4134	1.2212	1.5831	1.7012
Exchange	0.2217	**1.3714**	0.2979	1.6007	1.6377	2.4428	0.4731	1.9114	**0.2080**	0.8797	0.2230
Traffic	0.4656	**113.2022**	0.5053	142.4310	450.7319	1559.9105	0.6659	155.8423	**0.4324**	0.5070	0.4589
ECL	5.0315	**42.5782**	5.4703	43.0061	465.0539	937.6668	69.7845	93.0751	**4.9714**	8.8382	5.1357
Solar	0.2899	**33.0233**	0.3168	33.6705	259.8266	602.5870	0.6904	56.6606	**0.2768**	0.3370	4.5273

Table 2: Component-wise ablation isolates the contributions of online adaptation, source assignment, codebook conditioning, and residual-distribution updates. The best result for each dataset and metric is highlighted in bold. Lower values indicate better performance. "Excessive Runtime" means it takes more than 3 days to evaluate the whole dataset because of the low degree of parallelism due to the online setting.

Dataset	OnART (Ours)		Offline		w/o Source Assignment		w/o Codebook		Point Update Only		Naive Online Flow Matching	
	CRPS	CRPS_{sum}	CRPS	CRPS_{sum}	CRPS	CRPS_{sum}	CRPS	CRPS_{sum}	CRPS	CRPS_{sum}	CRPS	CRPS_{sum}
ETTh2	**0.588**	**2.946**	0.600	3.129	0.627	3.137	0.601	2.990	0.598	3.005	0.795	4.169
ETTm1	**0.395**	**1.703**	0.621	2.898	0.448	2.017	0.401	1.760	0.402	1.770	0.492	2.328
Weather	**0.455**	**4.254**	0.496	4.903	0.508	5.044	0.464	4.553	0.475	4.611	0.537	6.424
Exchange	**0.225**	**1.371**	0.227	1.381	0.232	1.457	0.265	1.638	0.231	1.430	0.258	1.744
ECL	**0.293**	**42.578**	0.298	43.953	0.305	44.876	0.351	48.642	**0.293**	43.492	Excessive Runtime	
Solar	0.242	**33.023**	**0.240**	33.100	0.261	33.091	0.274	36.414	0.276	36.614	Excessive Runtime	

3.3 Ablation Study

Table 2 shows that the complete model achieves the best CRPS_{sum} on every evaluated dataset. The consistency of this result demonstrates that the components act together to improve distributional accuracy rather than solving only isolated cases.

Online adaptation. Compared with the Offline variant, OnART reduces CRPS_{sum} by 10.7% on average. The largest gain occurs on ETTm1, where the score drops from 2.898 to 1.703. These gains demonstrate that recent residual feedback contains actionable distributional information and that OnART successfully incorporates it.

Source assignment. Proximity assignment improves CRPS and CRPS_{sum} by 7.1% and 8.1% on average over independent random pairing. This decisive gap validates the paper's central design choice: a Gaussian trajectory paired with a nearby generated residual provides substantially better online supervision than an unrelated source.

Codebook conditioning. Compared with continuous conditioning, the codebook improves CRPS and CRPS_{sum} by 8.1% and 8.2% on average. The improvement verifies that shared condition prototypes provide effective regularization for the conditional velocity model under the evaluated streaming shifts.

Residual-distribution adaptation. Updating both the point forecaster and velocity model improves CRPS and CRPS_{sum} by 3.8% and 4.9%, respectively, over updating the point forecaster alone. This ablation isolates the source of the gain: explicitly adapting the residual distribution contributes beyond any improvement obtained from point-model adaptation.

Online probabilistic baseline. Table 2 reports Naive Flow Matching under the same delayed-feedback protocol. Its evaluation on ECL and Solar exceeds the three-day runtime limit and is marked accordingly. Compared with the offline model, naive online flow matching degrades performance, increasing CRPS and CRPS_{sum} by an average of 8.4% and 17.7%, respectively, across the datasets for which evaluation is completed. In contrast, OnART reduces these two metrics by an average of 18.5% and 27.8%, respectively, compared with naive online flow matching.

This comparison exposes the failure mode that OnART is designed to solve. Naive random pairing can impose an arbitrarily large change from existing displacement supervision, whereas proximity assignment selects the cached anchor requiring the smallest endpoint correction and replay constrains the cached mappings.

3.4 Visualization

Figure 4 illustrates that OnART captures both central structure and predictive dispersion on ETTm1: the generated trajectories track the observed evolution while maintaining diverse samples around the central trend.

Table 3: Per-timestep forecasting and online-update latency on ETTh2 under the same hardware configuration.

Method	Forecast Latency (ms)	Update Latency (ms)	Total Latency (ms)
OnART	7.15	33.57	40.72
Proceed	2.36	8.30	10.66
OneNet	4.22	21.91	26.13
D3U	171.07	–	171.07
TMDM	761.22	–	761.22

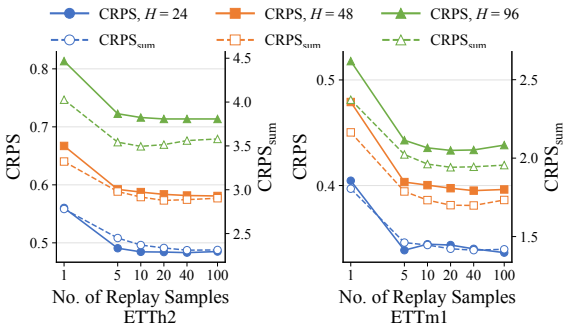


Figure 3: Sensitivity to the number R of generated pairs used for source assignment and replay. The x-axis is logarithmic.

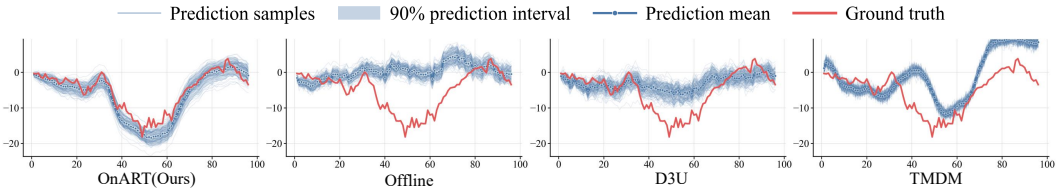


Figure 4: Forecast visualization on ETTm1, where each timestep is aggregated across all variables.

3.5 Efficiency Analysis

Table 3 reports per-timestep latency on ETTh2. OnART requires 7.15 ms for forecasting and 33.57 ms for adaptation, for a total of 40.72 ms. This modest premium over deterministic online methods buys explicit probabilistic sampling and velocity-model adaptation. OnART is decisively faster than the evaluated iterative diffusion baselines. Its forecasting latency is only 4.18% and 0.94% of D3U's and TMDM's, respectively. Even after including online updates, its total latency remains only 23.80% of D3U's and 5.35% of TMDM's forecasting latency. These measurements establish that offline re-flow and one-step generation remove the principal latency barrier to online probabilistic forecasting in this comparison.

3.6 Sensitivity to Generated Replay Size

Figure 3 varies R, the number of stored source–endpoint pairs used both as assignment candidates and as generated replay in each online update. Increasing R from 1 to 10 produces a clear probabilistic improvement, demonstrating the value of broader anchor coverage and stronger replay. Beyond 10 pairs, the gains saturate and become non-monotonic. We use $R = 100$ in the main experiments while fixing the predictive ensemble size at $K = 100$.

4 Conclusion and Future Work

We introduced **OnART**, a transport-aware framework that jointly adapts point predictions and residual uncertainty under delayed feedback with strictly causal access to observations. Its central contribution is a controlled transport update for endpoint-only feedback: channel-wise proximity assignment selects the forecast-time anchor requiring the smallest change in displacement supervision, generated replay anchors cached mappings, and codebook conditioning stabilizes the condition space. Theory characterizes the optimality and gradient stability of source assignment together with the generalization tradeoff of codebook conditioning. Experiments confirm reliable probabilistic adaptation while retaining competitive point accuracy.

Beyond time-series forecasting, OnART suggests a general principle for continual generative learning: when new observations provide endpoints but no prescribed sources, adaptation can reuse the source–endpoint pair requiring the smallest supervision correction and replay prior mappings. Extending this principle to domains that must absorb limited new data without forgetting is a direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to provide key ingredients for proving mathematical claims and to assist in writing proofs, to assist with translation and language editing, and to help write non-core research code. We did not use generative AI tools to design the core method or implement the core algorithm. The remaining required-disclosure tasks were either performed without generative AI or were not applicable to this work. The authors reviewed all AI-assisted material. In particular, we independently checked AI-assisted mathematical arguments against the stated assumptions and verified the resulting proofs, reviewed and tested AI-assisted non-core code for correctness, and reviewed all translated or edited text for technical accuracy and fidelity to the intended claims. We take responsibility for the final content of this work, including all text, claims, proofs, code, and other artifacts produced with the aid of generative AI.

REFERENCES

João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014. doi: 10.1145/2523813.

Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014. doi: 10.1146/annurev-statistics-062713-085831.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1232.

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 & Development in Information Retrieval*, pp. 95–104. Association for Computing Machinery, 2018. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

Ying-Yee Ava Lau, Zhiwen Shao, and Dit-Yan Yeung. Fast and slow streams for online time series forecasting without information leakage. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/46e624c244cff669223d488defd4e835-Abstract-Conference.html.

Qi Li, Zhenyu Zhang, Lei Yao, Zhaoxia Li, Tianyi Zhong, and Yong Zhang. Diffusion-based decoupled deterministic and uncertain framework for probabilistic multivariate time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a877dcfbf01491ef9e6dbc31cea4bba2-Abstract-Conference.html.

Yuxin Li, Wenchao Chen, Xinyue Hu, Bo Chen, Baolin Sun, and Mingyuan Zhou. Transformer-modulated diffusion models for probabilistic multivariate time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/516a9317af9d89e9f2251bd7fde49b8f-Abstract-Conference.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Max Planck Institute for Biogeochemistry. Weather station data. Max Planck Institute for Biogeochemistry, Jena, 2020. URL https://www.bgc-jena.mpg.de/wetter/. Dataset.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. doi: 10.1109/ICCV51070.2023.00387.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8857–8868. PMLR, 18–24 Jul 2021a. URL https://proceedings.mlr.press/v139/rasul21a.html.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2021b.

Matthias Seeger. PAC-bayesian generalisation error bounds for gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31016–31029. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/shen23d.html.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Artur Trindade. Electricityloaddiagrams20112014. UCI Machine Learning Repository, 2015. URL https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014. Dataset.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Tijin Yan, Hongwei Zhang, Tong Zhou, Yufeng Zhan, and Yuanqing Xia. ScoreGrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models, 2021. URL https://arxiv.org/abs/2106.10121.

Weiwei Ye, Zhuopeng Xu, and Ning Gui. Non-stationary diffusion for probabilistic time series forecasting. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 72112–72130. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/ye25i.html.

Yi-Fan Zhang, Qingsong Wen, Xue Wang, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 69949–69980. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/dd6a47bc0aad6f34aa5e77706d90cdc4-Paper-Conference.pdf.

Lifan Zhao and Yanyan Shen. Proactive model adaptation against concept drift for online time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 2020–2031, New York, NY, USA, August 2025. Association for Computing Machinery. ISBN 979-8-4007-1245-6. doi: 10.1145/3690624.3709210. URL https://doi.org/10.1145/3690624.3709210.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/17325.

A Related Works

A.1 Probabilistic Time-Series Forecasting

Probabilistic forecasting estimates a predictive distribution rather than only a conditional mean. Recent approaches, including D3U (Li et al., 2025), TMDM (Li et al., 2024), and NsDiff (Ye et al., 2025), use diffusion-based generators to model complex forecast uncertainty. Conditional normalizing flows have also been used for multivariate probabilistic forecasting (Rasul et al., 2021b). These methods establish strong offline probabilistic forecasts, but they do not adapt the predictive distribution to streaming feedback; iterative diffusion sampling also makes frequent deployment-time updates costly.

A.2 Online Time-Series Forecasting

Online forecasting updates a model as new observations arrive. OneNet (Zhang et al., 2023) combines complementary predictors, DSOF (Lau et al., 2025) employs fast and slow adaptation streams, and Proceed (Zhao & Shen, 2025) addresses discrepancies caused by delayed labels. These methods effectively adapt deterministic point forecasts but leave the predictive distribution fixed or implicit. OnART targets the complementary and harder objective of explicitly adapting uncertainty itself.

A.3 Generative Models for Time Series

Diffusion-based forecasters generate future trajectories through iterative denoising (Ho et al., 2020; Tashiro et al., 2021; Rasul et al., 2021a; Yan et al., 2021; Shen & Kwok, 2023; Kollovieh et al., 2023), which increases inference and adaptation costs in online settings. Flow Matching (Lipman et al., 2023) learns a transport field between a simple source and a target distribution, while Rectified Flow (Liu et al., 2023) produces straighter trajectories and supports few-step or one-step sampling after re-flow. Unlike offline conditional-flow forecasting, OnART addresses endpoint-only supervision after deployment: delayed feedback supplies a new residual endpoint, but online re-flow still requires a source from which to revise the transport. Its novelty lies in selecting a low-shift forecast-time anchor through channel-wise proximity assignment and constraining the resulting update through generated replay, rather than in conditional flow modeling alone.

OnART consequently advances beyond offline generative forecasting by updating a sample-based residual distribution from sparse, delayed feedback. Proximity-based assignment supplies a source anchor that minimizes the change in re-flow displacement supervision among the cached candidates.

B Proof of Theorem

Theorem 1 (Optimality of channel-wise source assignment)**.** *Assume that the residual velocity model is channel-separable:*

$$[f_\theta(\mathbf{X}, t, \mathbf{C})]_{:,c} = v_{\theta,c}(\mathbf{X}_{:,c}, t, \mathbf{C}_c), \qquad c = 1, \dots, C. \tag{21}$$

For any tuple of replay indices $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_C) \in \{1, \dots, R\}^C$, assemble

$$\mathbf{A}^{\boldsymbol{\kappa}}_{:,c} = \mathbf{X}^{(\kappa_c)}_{0,:,c}, \qquad \mathbf{B}^{\boldsymbol{\kappa}}_{:,c} = \hat{\mathbf{X}}^{(\kappa_c)}_{1,:,c}. \tag{22}$$

Here $\mathbf{A}^{\boldsymbol{\kappa}}$ is the assembled source and $\mathbf{B}^{\boldsymbol{\kappa}}$ is its assembled forecast-time anchor endpoint. Define the corresponding forecast-time and new displacement targets:

$$\mathbf{U}^{\boldsymbol{\kappa}}_{\text{anchor}} = \mathbf{B}^{\boldsymbol{\kappa}} - \mathbf{A}^{\boldsymbol{\kappa}}, \qquad \mathbf{U}^{\boldsymbol{\kappa}}_{\text{new}} = \mathbf{X}^{\text{new}}_1 - \mathbf{A}^{\boldsymbol{\kappa}}. \tag{23}$$

For each channel, let

$$k^\star_c \in \arg\min_{1 \le k \le R} \left\| \mathbf{X}^{\text{new}}_{1,:,c} - \hat{\mathbf{X}}^{(k)}_{1,:,c} \right\|_2, \qquad \epsilon_c = \left\| \mathbf{X}^{\text{new}}_{1,:,c} - \hat{\mathbf{X}}^{(k^\star_c)}_{1,:,c} \right\|_2. \tag{24}$$

Then $\boldsymbol{\kappa}^\star = (k^\star_1, \dots, k^\star_C)$ globally minimizes the displacement-target shift over all channel-wise combinations:

$$\begin{aligned} \boldsymbol{\kappa}^\star &\in \arg\min_{\boldsymbol{\kappa} \in \{1,\dots,R\}^C} \left\| \mathbf{U}^{\boldsymbol{\kappa}}_{\text{new}} - \mathbf{U}^{\boldsymbol{\kappa}}_{\text{anchor}} \right\|_F, \\ \left\| \mathbf{U}^{\boldsymbol{\kappa}^\star}_{\text{new}} - \mathbf{U}^{\boldsymbol{\kappa}^\star}_{\text{anchor}} \right\|_F &= \left[\sum_{c=1}^{C} \epsilon_c^2 \right]^{1/2}. \end{aligned} \tag{25}$$

Let $n = HC$ and define

$$\ell(\theta; \mathbf{A}, \mathbf{U}, \mathbf{C}) = \frac{1}{n} \left\| f_\theta(\mathbf{A}, 0, \mathbf{C}) - \mathbf{U} \right\|_F^2. \tag{26}$$

Assume that, for all candidate tuples, the parameter Jacobian

$$\mathbf{J}_\theta^{\boldsymbol{\kappa}} = \frac{\partial \operatorname{vec} f_\theta\left(\mathbf{A}^{\boldsymbol{\kappa}}, 0, \mathbf{C}^{(t)}\right)}{\partial \theta} \tag{27}$$

has operator norm at most B_J. Let $\ell_{\text{new}}^{\boldsymbol{\kappa}}$ and $\ell_{\text{anchor}}^{\boldsymbol{\kappa}}$ use the same assembled source and current condition $\mathbf{C}^{(t)}$, but $\mathbf{U}_{\text{new}}^{\boldsymbol{\kappa}}$ and $\mathbf{U}_{\text{anchor}}^{\boldsymbol{\kappa}}$, respectively. Then

$$\left\|\nabla_\theta \ell_{\text{new}}^{\boldsymbol{\kappa}^\star} - \nabla_\theta \ell_{\text{anchor}}^{\boldsymbol{\kappa}^\star}\right\|_2 \le \frac{2B_J}{HC}\left[\sum_{c=1}^{C} \epsilon_c^2\right]^{1/2}. \tag{28}$$

Thus, proximity assignment minimizes the uniform upper bound induced by endpoint reassignment among the candidate tuples.

Finally, let $\mathcal{R}_t(\theta)$ denote the common sum of the R stored replay losses and define

$$\begin{aligned} \mathcal{L}_{\text{new}}^{\boldsymbol{\kappa}} &= \frac{\ell_{\text{new}}^{\boldsymbol{\kappa}} + \mathcal{R}_t(\theta)}{R+1}, \\ \mathcal{L}_{\text{anchor}}^{\boldsymbol{\kappa}} &= \frac{\ell_{\text{anchor}}^{\boldsymbol{\kappa}} + \mathcal{R}_t(\theta)}{R+1}. \end{aligned} \tag{29}$$

Then

$$\left\|\nabla_\theta \mathcal{L}_{\text{new}}^{\boldsymbol{\kappa}^\star} - \nabla_\theta \mathcal{L}_{\text{anchor}}^{\boldsymbol{\kappa}^\star}\right\|_2 \le \frac{2B_J}{(R+1)HC}\left[\sum_{c=1}^{C} \epsilon_c^2\right]^{1/2}. \tag{30}$$

All quantities in the gradient comparison are evaluated at the current pre-update velocity parameters θ and the condition $\mathbf{C}^{(t)}$ retained from the same pre-update PatchTST snapshot. The anchor loss is a counterfactual reference used to isolate the target change; the statement does not require the forecast-time and current velocity models or conditions to coincide. Because $\mathbf{X}_1^{\text{new}}$ is constructed from that current pre-update point prediction, the ϵ_c terms also retain, rather than discard, the residual-coordinate shift accumulated since the original forecast.

Proof. For any fixed $\boldsymbol{\kappa}$, the assembled source cancels when the two displacement targets are subtracted:

$$\begin{aligned} \mathbf{U}_{\text{new}}^{\boldsymbol{\kappa}} - \mathbf{U}_{\text{anchor}}^{\boldsymbol{\kappa}} &= (\mathbf{X}_1^{\text{new}} - \mathbf{A}^{\boldsymbol{\kappa}}) - (\mathbf{B}^{\boldsymbol{\kappa}} - \mathbf{A}^{\boldsymbol{\kappa}}) \\ &= \mathbf{X}_1^{\text{new}} - \mathbf{B}^{\boldsymbol{\kappa}}. \end{aligned} \tag{31}$$

Therefore,

$$\begin{aligned} \|\mathbf{U}_{\text{new}}^{\boldsymbol{\kappa}} - \mathbf{U}_{\text{anchor}}^{\boldsymbol{\kappa}}\|_F^2 &= \|\mathbf{X}_1^{\text{new}} - \mathbf{B}^{\boldsymbol{\kappa}}\|_F^2 \\ &= \sum_{c=1}^{C} \left\|\mathbf{X}_{1,:,c}^{\text{new}} - \hat{\mathbf{X}}_{1,:,c}^{(\kappa_c)}\right\|_2^2. \end{aligned} \tag{32}$$

Every term in the final sum depends on only one index κ_c. Hence the sum is globally minimized by selecting each $k_c^\star$ independently. At this choice, the c-th term equals ϵ_c^2, proving Eq. 25. Channel separability ensures that assembling channels selected from different replay members does not introduce cross-channel state interactions in the velocity model.

It remains to prove the gradient statements. For a fixed $\boldsymbol{\kappa}$, write

$$\begin{aligned} \mathbf{F}^{\boldsymbol{\kappa}} &= \operatorname{vec} f_\theta\left(\mathbf{A}^{\boldsymbol{\kappa}}, 0, \mathbf{C}^{(t)}\right), \\ \mathbf{u}_{\text{new}}^{\boldsymbol{\kappa}} &= \operatorname{vec} \mathbf{U}_{\text{new}}^{\boldsymbol{\kappa}}, \qquad \mathbf{u}_{\text{anchor}}^{\boldsymbol{\kappa}} = \operatorname{vec} \mathbf{U}_{\text{anchor}}^{\boldsymbol{\kappa}}. \end{aligned} \tag{33}$$

Because the source, condition, and parameters are identical in the two losses, their parameter Jacobian is the same. Differentiating the squared losses gives

$$\begin{aligned} &\nabla_\theta \ell_{\text{new}}^{\boldsymbol{\kappa}} - \nabla_\theta \ell_{\text{anchor}}^{\boldsymbol{\kappa}} \\ &= \frac{2}{n}\left(\mathbf{J}_\theta^{\boldsymbol{\kappa}}\right)^\top \left[(\mathbf{F}^{\boldsymbol{\kappa}} - \mathbf{u}_{\text{new}}^{\boldsymbol{\kappa}}) - (\mathbf{F}^{\boldsymbol{\kappa}} - \mathbf{u}_{\text{anchor}}^{\boldsymbol{\kappa}})\right] \\ &= \frac{2}{n}\left(\mathbf{J}_\theta^{\boldsymbol{\kappa}}\right)^\top \left(\mathbf{u}_{\text{anchor}}^{\boldsymbol{\kappa}} - \mathbf{u}_{\text{new}}^{\boldsymbol{\kappa}}\right). \end{aligned} \tag{34}$$

Taking Euclidean norms, applying the operator-norm bound, and using invariance of the Frobenius norm under vectorization yields

$$\begin{aligned} &\|\nabla_\theta \ell_{\text{new}}^{\boldsymbol{\kappa}} - \nabla_\theta \ell_{\text{anchor}}^{\boldsymbol{\kappa}}\|_2 \\ &\le \frac{2B_J}{n}\|\mathbf{U}_{\text{new}}^{\boldsymbol{\kappa}} - \mathbf{U}_{\text{anchor}}^{\boldsymbol{\kappa}}\|_F. \end{aligned} \tag{35}$$

Substituting $\boldsymbol{\kappa}^\star$, $n = HC$, and Eq. 25 proves Eq. 28. Because B_J is a uniform bound over the candidate tuples, minimizing the displacement-target shift also minimizes this uniform gradient-perturbation upper bound.

For the replay-averaged objectives, the common replay term cancels exactly:

$$\nabla_\theta \mathcal{L}^{\boldsymbol{\kappa}}_{\text{new}} - \nabla_\theta \mathcal{L}^{\boldsymbol{\kappa}}_{\text{anchor}} = \frac{1}{R+1} \left(\nabla_\theta \ell^{\boldsymbol{\kappa}}_{\text{new}} - \nabla_\theta \ell^{\boldsymbol{\kappa}}_{\text{anchor}} \right). \tag{36}$$

Applying Eq. 28 proves Eq. 30.

□

C Analysis on Codebook Quantization

Theorem 2 (Generalization bound for codebook quantization)**.** *Let $\widehat{\mathcal{C}} \in \mathbb{R}^{K\times d}$ be a codebook learned from an i.i.d. training sample $S \sim \mathcal{D}^n$, where $n \geq 8$. Assume that:*

1. *the loss satisfies $\ell(h_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) \in [0,1]$;*
2. *the learned codebook is bounded as $\|\widehat{\mathcal{C}}\|_F \leq B$;*
3. *the prior and posterior over the flattened codebook parameters are*

$$P = \mathcal{N}\left(\mathbf{0}, \sigma_0^2 \mathbf{I}_{Kd}\right), \qquad Q_{\widehat{\mathcal{C}}} = \mathcal{N}\left(\widehat{\mathcal{C}}, \sigma_q^2 \mathbf{I}_{Kd}\right); \tag{37}$$

4. *for*

$$\mathcal{B}_r = \{\boldsymbol{\theta} : \|\boldsymbol{\theta}\|_F \leq B + r\}, \tag{38}$$

the surrogate loss is uniformly Lipschitz in the codebook parameters:

$$\sup_{(\mathbf{x},\mathbf{y})\in \text{supp}(\mathcal{D})} \sup_{\boldsymbol{\theta}\in\mathcal{B}_r} \|\nabla_{\boldsymbol{\theta}} \ell\left(h_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}\right)\|_F \leq L_\theta(r). \tag{39}$$

Define

$$\mu_{Kd} = \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}} \left[\left\| \boldsymbol{\theta} - \widehat{\mathcal{C}} \right\|_F \right]. \tag{40}$$

Then, for any $\delta \in (0,1)$ and any $r > \mu_{Kd}$, with probability at least $1-\delta$ over $S \sim \mathcal{D}^n$,

$$\begin{aligned} R_{\text{test}}(\widehat{\mathcal{C}}) \leq R_{\text{train}}(\widehat{\mathcal{C}}) &+ \sqrt{\frac{\text{KL}\left(Q_{\widehat{\mathcal{C}}} \,\|\, P\right) + \ln\left(2\sqrt{n}/\delta\right)}{2n}} \\ &+ 2L_\theta(r)\sigma_q\sqrt{Kd} + 2\exp\left(-\frac{(r-\mu_{Kd})^2}{2\sigma_q^2}\right). \end{aligned} \tag{41}$$

Proof. By the one-sided PAC–Bayes inequality (Seeger, 2002), with probability at least $1-\delta$ over the training sample,

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[R_{\text{test}}(\boldsymbol{\theta})\right] \leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[R_{\text{train}}(\boldsymbol{\theta})\right] \\ + \sqrt{\frac{\text{KL}\left(Q_{\widehat{\mathcal{C}}} \,\|\, P\right) + \ln\left(2\sqrt{n}/\delta\right)}{2n}}. \end{aligned} \tag{42}$$

For

$$F \in \{R_{\text{train}}, R_{\text{test}}\}, \tag{43}$$

Jensen's inequality gives

$$\left| F(\widehat{\mathcal{C}}) - \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[F(\boldsymbol{\theta})\right] \right| \leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[\left| F(\boldsymbol{\theta}) - F(\widehat{\mathcal{C}}) \right| \right]. \tag{44}$$

Define

$$\mathcal{A}_r = \left\{ \left\| \boldsymbol{\theta} - \widehat{\mathcal{C}} \right\|_F \leq r \right\}. \tag{45}$$

On $\mathcal{A}_r$, the boundedness of $\widehat{\mathcal{C}}$ implies

$$\|\boldsymbol{\theta}\|_F \leq \left\| \boldsymbol{\theta} - \widehat{\mathcal{C}} \right\|_F + \|\widehat{\mathcal{C}}\|_F \leq B + r. \tag{46}$$

Therefore, the Lipschitz assumption gives

$$\left| F(\boldsymbol{\theta}) - F(\widehat{\mathcal{C}}) \right| \leq L_\theta(r) \left\| \boldsymbol{\theta} - \widehat{\mathcal{C}} \right\|_F. \tag{47}$$

Table 4: Dataset-specific experimental configurations. The train/validation/test ratios shown for the non-ETT datasets follow the chronological partitions implemented by the data loader. All datasets are evaluated under the multivariate-to-multivariate setting.

Dataset	Variables	Split	Input length	Forecast horizons	Train batch
ETTh2	7	4/1/16 months	96	24, 48, 96	32
ETTm1	7	4/1/16 months	96	24, 48, 96	32
ECL	321	20%/5%/75%	96	24, 48, 96	8
Exchange	8	20%/5%/75%	96	24, 48, 96	32
Solar	137	20%/5%/75%	96	24, 48, 96	32
Traffic	862	20%/5%/75%	96	24, 48, 96	4
Weather	21	20%/5%/75%	96	24, 48, 96	32

On $\mathcal{A}_r^c$, the boundedness of the loss implies

$$\left|F(\boldsymbol{\theta}) - F(\widehat{\mathcal{C}})\right| \leq 1. \tag{48}$$

Splitting the expectation over $\mathcal{A}_r$ and $\mathcal{A}_r^c$ yields

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[\left|F(\boldsymbol{\theta}) - F(\widehat{\mathcal{C}})\right|\right] \\ &\quad\leq L_\theta(r)\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[\left\|\boldsymbol{\theta} - \widehat{\mathcal{C}}\right\|_F\right] + \Pr_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left(\mathcal{A}_r^c\right).\end{aligned} \tag{49}$$

Since

$$\boldsymbol{\theta} - \widehat{\mathcal{C}} \sim \mathcal{N}\left(\mathbf{0}, \sigma_q^2 \mathbf{I}_{Kd}\right), \tag{50}$$

Jensen's inequality gives

$$\begin{aligned}\mathbb{E}\left[\left\|\boldsymbol{\theta} - \widehat{\mathcal{C}}\right\|_F\right] &\leq \sqrt{\mathbb{E}\left[\left\|\boldsymbol{\theta} - \widehat{\mathcal{C}}\right\|_F^2\right]} \\ &= \sigma_q\sqrt{Kd}.\end{aligned} \tag{51}$$

Moreover, Gaussian concentration for the Euclidean norm implies

$$\Pr\left(\left\|\boldsymbol{\theta} - \widehat{\mathcal{C}}\right\|_F > r\right) \leq \exp\left(-\frac{(r-\mu_{Kd})^2}{2\sigma_q^2}\right). \tag{52}$$

Combining equation 49, equation 51, and equation 52, we obtain

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\widehat{\mathcal{C}}}}\left[\left|F(\boldsymbol{\theta}) - F(\widehat{\mathcal{C}})\right|\right] \\ &\quad\leq L_\theta(r)\sigma_q\sqrt{Kd} + \exp\left(-\frac{(r-\mu_{Kd})^2}{2\sigma_q^2}\right).\end{aligned} \tag{53}$$

Applying equation 53 to both R_{test} and R_{train}, and combining the resulting inequalities with equation 42, gives

$$\begin{aligned}R_{\text{test}}(\widehat{\mathcal{C}}) \leq R_{\text{train}}(\widehat{\mathcal{C}}) &+ \sqrt{\frac{\mathrm{KL}\left(Q_{\widehat{\mathcal{C}}} \,\|\, P\right) + \ln\left(2\sqrt{n}/\delta\right)}{2n}} \\ &+ 2L_\theta(r)\sigma_q\sqrt{Kd} + 2\exp\left(-\frac{(r-\mu_{Kd})^2}{2\sigma_q^2}\right),\end{aligned} \tag{54}$$

which proves the result. □

D Detailed Experiment Configurations

We evaluate OnART across seven widely used multivariate time-series datasets, three forecast horizons, and diverse dimensionalities. Unless otherwise specified, all experiments use an input context of 96 time steps and forecast horizons of $H \in \{24, 48, 96\}$. The multivariate series are standardized using statistics computed exclusively from the training partition. The data windows advance by one time step, and no training samples are discarded at the end of an epoch. Table 4 summarizes the dataset-specific configurations.

For the ETT datasets, the first four months are used for training, the following month for validation, and the next sixteen months for evaluation, following the data-loader boundaries. Each partition is windowed independently.

Table 5: Architecture configuration shared across datasets. Dataset-specific deviations are stated explicitly.

Component	Hyperparameter	Value
Point forecaster	Backbone	PatchTST (Nie et al., 2023)
Point forecaster	Patch length / stride / end padding	16 / 8 / 8
Point forecaster	Encoder layers / attention heads	1 / 2
Point forecaster	Hidden dimension / feed-forward dimension	512 / 2048
Point forecaster	Traffic feed-forward dimension	1024
Point forecaster	Dropout / activation / attention factor	0.1 / GELU / 3
Point forecaster	Normalization	Per-window mean–variance normalization and encoder BatchNorm
Conditional flow	Velocity backbone	PatchDN (Li et al., 2025) with adaLN-Zero conditioning (Peebles & Xie, 2023)
Conditional flow	Patch length / stride / padding mode	16 / 8 / end
Conditional flow	Blocks / attention heads / hidden dimension	2 / 8 / 128
Conditional flow	MLP ratio / timestep embedding dimension	4 / 256
Conditional flow	Interpolation path / time distribution	Linear / $t \sim \mathcal{U}(0,1)$
Conditional flow	Training objective	Mean-squared velocity matching
Discrete condition	Codebook entries / entry dimension	256 / 512
Discrete condition	Assignment	Hard nearest-neighbor assignment in Euclidean distance
Discrete condition	Commitment coefficient	0.25
Discrete condition	Online behavior	Frozen

Table 6: Offline optimization and re-flow configuration used whenever checkpoint training is enabled.

Stage	Hyperparameter	Value
Point forecaster	Optimizer / initial learning rate	AdamW (Loshchilov & Hutter, 2019) / 1×10^{-4}
Point forecaster	Maximum epochs / early-stopping patience	100 / 5
Point forecaster	Loss	MSE
Flow and codebook	Optimizer / initial learning rate	Adam (Kingma & Ba, 2015) / 1×10^{-4}
Flow and codebook	Maximum epochs / early-stopping patience	100 / 5
Flow and codebook	Loss	Velocity MSE + VQ loss
Learning-rate schedule	Schedule	Cosine decay to zero
Re-flow data generation	Samples per condition / Euler steps	10 / 20
Re-flow optimization	Epochs / optimizer / initial learning rate	10 / AdamW / 1×10^{-4}

For the other datasets, the first 20% of observations form the training set, the following 5% form the validation set, and the final 75% form the evaluation stream. The validation and evaluation windows additionally retain the preceding observations required to construct the look-back sequence, while all prediction targets remain strictly within their corresponding chronological partitions. The online data loader preserves temporal order and uses a batch size of one, whereas the offline training loader shuffles the training windows.

All series are standardized using statistics computed from the training partition, and the reported metrics are computed in the standardized space.

The point forecaster first produces

$$\mathbf{y}_{\mathrm{Point}} = d_\psi\left(b_\phi(\mathbf{l})\right). \tag{55}$$

The residual target used for conditional flow training is

$$\mathbf{x}_1 = \mathbf{y} - \mathbf{y}_{\mathrm{Point}}. \tag{56}$$

The backbone representation is mapped to the discrete condition through the shared codebook:

$$\mathbf{c} = b_\phi(\mathbf{l}), \qquad \mathbf{c}_{\mathrm{disc}} = Q_E(\mathbf{c}). \tag{57}$$

The codebook objective consists of the codebook reconstruction loss and a commitment loss weighted by 0.25.

The deterministic forecaster is selected according to validation MSE. During subsequent offline conditional-flow training, the point forecaster remains frozen. Offline re-flow then generates coupled source–endpoint pairs using the pretrained flow model and fine-tunes the velocity model for ten additional epochs. All learning rates in the offline stages follow cosine decay.

At each online timestep t, OnART first constructs and outputs the current probabilistic forecast without accessing its target. The data loader then appends the current stride-one window to two deques of maximum length $H+1$. Once the deques are full, the complete target of the oldest window and the causally available prefixes of the newer windows provide feedback for one post-prediction online update. The updated point and velocity models are first used for the forecast at timestep $t+1$.

The point forecaster is evaluated on all $H+1$ windows. A triangular mask retains entries satisfying $i+h \le t$ and sets the remaining loss entries to zero. The implementation then averages the masked squared-error tensor over all $(H+1)HC$ entries. Consequently, a realized timestamp participates in each overlapping window in which it appears.

Before either online optimizer step, the masked forward pass used to construct the point loss also produces the point output and representation for the oldest window. Its residual target is

$$\mathbf{x}^{\mathrm{new}}_{1,t-H} = \mathbf{y}_{t-H} - \mathbf{y}^{(t)}_{\mathrm{Point},t-H}. \tag{58}$$

Table 7: Dataset-specific online learning rates.

Dataset	Flow learning rate	Point-forecaster learning rate
ETTh2	1×10^{-6}	1×10^{-6}
ETTm1	1×10^{-6}	1×10^{-6}
ECL	1×10^{-7}	1×10^{-7}
Exchange	3×10^{-7}	1×10^{-6}
Solar	1×10^{-7}	1×10^{-6}
Traffic	1×10^{-7}	1×10^{-7}
Weather	1×10^{-6}	1×10^{-6}

Table 8: Online adaptation, sampling, and evaluation configuration shared across datasets and horizons.

Category	Hyperparameter	Value
Feedback	Online batch size / window stride	1 / 1
Feedback	Effective feedback delay	H time steps
Point update	Parameters / optimizer / objective	PatchTST / AdamW / masked MSE
Flow update	Optimizer	AdamW
Flow update	Interpolation time	Fixed at $\tau = 0$
Flow replay	Generated pairs / new residuals per update	100 / 1
Flow replay	Noise coupling	Channel-wise proximity assignment
Online buffer	Retained forecast items	$H + 1$
Sampling	Predictive samples per input	100
Sampling	Solver / number of function evaluations	Explicit Euler / 1
Point metrics	MSE and MAE	Computed from the ensemble mean
Probabilistic metrics	CRPS and $\mathrm{CRPS}_{\mathrm{sum}}$	Computed from all 100 samples

The corresponding representation is quantized by the frozen codebook to obtain the condition for the new pair. The point output and condition therefore come from the same pre-update PatchTST parameter state that issued the current forecast. After the point and re-flow objectives have both been constructed, one PatchTST step and one velocity-model step are taken in the same update phase. No post-step forward pass is used: the retained state is the common coordinate for this synchronous update, not an assertion that the representation is invariant to the PatchTST step. At the next timestep, the updated point and velocity models are deployed together using only the new look-back sequence.

The online buffer retains the $K = 100$ Gaussian sources, generated residual endpoints, and quantized condition produced for each of the latest $H + 1$ forecasts. The first R pairs are used for assignment and replay; in the main experiments $R = 100$. For each variable c, the implementation selects

$$k_c^\star = \underset{k\in\{1,\dots,R\}}{\arg\min} \left\| \mathbf{x}^{\mathrm{new}}_{1,t-H,:,c} - \hat{\mathbf{x}}^{(k)}_{1,t-H,:,c} \right\|_2 . \tag{59}$$

The assigned source is assembled channel-wise:

$$\tilde{\mathbf{x}}_{0,t-H,:,c} = \mathbf{x}^{(k_c^\star)}_{0,t-H,:,c}. \tag{60}$$

The assigned pair is concatenated with the R generated pairs stored for the same delayed forecast. The implementation fixes $\tau = 0$ and applies a single mean-squared velocity loss over all $R+1$ pairs and all trajectory entries. The PatchTST and velocity-model optimizer steps conclude the timestep. At timestep $t + 1$, their updated parameters produce the next point forecast, condition, and K one-step residual samples without accessing that forecast's target.

Naive Flow Matching follows the same delayed-feedback schedule but independently samples the Gaussian source paired with each new residual. Its ECL and Solar runs exceed the three-day runtime limit and are marked "Excessive Runtime" in the tables.

The reported point-estimation metrics are computed from the empirical mean of the $K = 100$ final predictive trajectories. CRPS (Gneiting & Raftery, 2007) and $\mathrm{CRPS}_{\mathrm{sum}}$ (Rasul et al., 2021a) are computed directly from the same predictive ensemble.

Given predictive samples $\{\hat{y}^{(k)}_{t,c}\}_{k=1}^K$ and the corresponding observation $y_{t,c}$, the empirical continuous ranked probability score is

$$\begin{aligned} \mathrm{CRPS}_{t,c} = \frac{1}{K}\sum_{k=1}^{K} \left| \hat{y}^{(k)}_{t,c} - y_{t,c} \right| \\ - \frac{1}{2K^2}\sum_{k=1}^{K}\sum_{r=1}^{K} \left| \hat{y}^{(k)}_{t,c} - \hat{y}^{(r)}_{t,c} \right| . \end{aligned} \tag{61}$$

The first term measures the distance between the predictive ensemble and the observation, whereas the second term accounts for the dispersion of the predictive distribution. The reported CRPS is obtained by averaging Eq. 61 over forecast positions, variables, and evaluation windows.

To compute $\mathrm{CRPS}_{\mathrm{sum}}$, each generated multivariate prediction and its corresponding observation are first aggregated over the variable dimension:

$$\hat{y}^{(k)}_{t,\mathrm{sum}} = \sum_{c=1}^{C} \hat{y}^{(k)}_{t,c}, \qquad y_{t,\mathrm{sum}} = \sum_{c=1}^{C} y_{t,c}. \tag{62}$$

The same empirical CRPS estimator is then applied to $\hat{y}^{(k)}_{t,\mathrm{sum}}$ and $y_{t,\mathrm{sum}}$, and the resulting scores are averaged over forecast positions and evaluation windows. CRPS evaluates marginal predictive quality, whereas $\mathrm{CRPS}_{\mathrm{sum}}$ evaluates the probabilistic accuracy of the aggregate signal. Because OnART's residual velocity field is channel-separable, $\mathrm{CRPS}_{\mathrm{sum}}$ should be interpreted as an aggregate-distribution metric rather than as evidence that the residual transport models cross-channel interactions.

All experiments are conducted on a single NVIDIA V100 GPU with 32 GB memory. The complete source code will be made publicly available upon publication of the paper.

E Algorithm of OnART

Algorithm 1 specifies the causal deployment protocol of OnART, from offline transport construction through predict–then–update adaptation and one-step probabilistic generation.

Algorithm 1: OnART for Probabilistic Streaming Time-Series Forecasting

Train the backbone b_ϕ and decoder d_ψ using $\mathcal{L}_{\mathrm{Point}}$.
Freeze the point forecaster and train the codebook $Q_{\mathcal{E}}$ and velocity model f_θ using $\mathcal{L}_{\mathrm{Flow}} + \lambda_{\mathrm{VQ}}\mathcal{L}_{\mathrm{VQ}}$.
Generate N source–target pairs $(\mathbf{x}_0, \hat{\mathbf{x}}_1)$ and optimize $\mathcal{L}_{\mathrm{Reflow}}$.
Freeze the codebook and initialize feedback and generated-pair buffers of length $H+1$.
for each online timestep t **do**
  Without accessing $\mathbf{y}_t$, compute $\mathbf{y}_{\mathrm{Point},t}$ and $\mathbf{c}_{\mathrm{disc},t}$ with the current point forecaster and frozen codebook.
  Draw K Gaussian sources, generate K residual endpoints using one-step Euler sampling, and output the K probabilistic forecasts.
  Store their sources, residual endpoints, and quantized condition.
  Append the current stride-one window to the feedback buffer.
  if $t \geq H$ **then**
    Before either optimizer step, evaluate PatchTST on the latest $H+1$ windows, retain its point outputs and conditions, and construct $\mathcal{L}^{\mathrm{online}}_{\mathrm{Point},t}$.
    From the same pre-update masked-forward snapshot, compute the newly observed residual target $\mathbf{x}^{\mathrm{new}}_{1,t-H} = \mathbf{y}_{t-H} - \mathbf{y}^{(t)}_{\mathrm{Point},t-H}$ and its quantized condition.
    Retrieve the first R source–endpoint pairs and stored condition for timestep $t-H$.
    For each variable c, select

$$k^\star_c = \operatorname*{arg\,min}_{k\in\{1,\dots,R\}} \left\| \mathbf{x}^{\mathrm{new}}_{1,t-H,:,c} - \hat{\mathbf{x}}^{(k)}_{1,t-H,:,c} \right\|_2 .$$

    Assemble the assigned source channel-wise: $\tilde{\mathbf{x}}_{0,t-H,:,c} = \mathbf{x}^{(k^\star_c)}_{0,t-H,:,c}$.
    Construct $\mathcal{L}^{\mathrm{online}}_{\mathrm{Reflow},t}$ at $\tau = 0$ from the assigned pair and R stored generated pairs.
    In the same post-prediction update phase, take one PatchTST step on $\mathcal{L}^{\mathrm{online}}_{\mathrm{Point},t}$ and one f_θ step on $\mathcal{L}^{\mathrm{online}}_{\mathrm{Reflow},t}$.
  end if
end for

F Full Table of Results

The main text reports averaged results and the most representative comparisons. Table 9 provides the complete evidence across every dataset and prediction horizon, including both probabilistic and deterministic baselines.

Table 9: Detailed forecasting results under different prediction horizons. Probabilistic methods are evaluated using MSE and CRPS_{sum}, whereas deterministic methods are evaluated using MSE only. Lower values indicate better performance.

Dataset	Horizon	Online Probabilistic Methods		Offline Probabilistic Methods						Online Deterministic Methods		
		Ours		D3U		TMDM		NsDiff		Proceed	DSOF	OneNet
		MSE	CRPS_{sum}	MSE	CRPS_{sum}	MSE	CRPS_{sum}	MSE	CRPS_{sum}	MSE	MSE	MSE
ETTh2	24	1.9841	2.3564	2.1235	2.6305	5.5729	3.5981	4.4417	3.1443	1.7518	10.2663	2.5827
	48	3.2842	2.8945	3.6662	3.0085	5.7820	3.7330	4.7353	3.2088	3.6310	10.5112	3.7087
	96	5.6313	3.5873	6.7810	3.8158	20.3551	4.7150	5.4464	3.4825	6.3510	21.3248	5.9689
ETTm1	24	0.5044	1.4184	0.7413	1.6899	1.2254	2.4450	1.0209	2.1375	0.5611	0.6933	0.7625
	48	0.6858	1.7309	1.0932	2.0363	1.3048	2.7568	1.0425	2.1877	0.8958	0.8565	1.1723
	96	0.8056	1.9598	1.1861	2.2620	1.2556	2.6938	1.4370	2.9493	0.8816	0.9661	1.0451
Weather	24	0.8366	3.1188	1.0286	3.9192	1.5906	5.5559	1.2172	4.5725	0.8559	1.1458	0.9814
	48	1.2255	4.2271	1.5166	5.2157	2.0036	7.4569	2.3295	7.0973	1.2320	1.5704	1.3568
	96	1.5707	5.4175	1.7032	5.9116	2.3629	6.3492	2.5581	7.5705	1.5756	2.0330	2.7653
Exchange	24	0.0978	0.9420	0.1194	1.1297	1.3767	2.0546	0.3484	1.5485	0.1006	0.6320	0.1003
	48	0.1958	1.3183	0.2751	1.6777	1.5329	2.3714	0.4324	1.9456	0.1795	0.7999	0.1975
	96	0.3714	1.8539	0.4993	1.9946	2.0034	2.9024	0.6383	2.2401	0.3438	1.2071	0.3713
Traffic	24	0.4286	99.3810	0.4739	118.9577	452.1133	1559.1104	0.6289	134.4888	0.3982	0.4749	0.4208
	48	0.4796	118.3927	0.5115	134.8096	450.2468	1558.7734	0.6502	144.6871	0.4405	0.4969	0.4612
	96	0.4885	121.8329	0.5305	173.5257	449.8357	1561.8476	0.7186	188.3512	0.4584	0.5493	0.4948
ECL	24	4.2343	35.9336	4.6112	38.0907	473.5493	927.2150	73.7036	92.1066	4.2802	7.4755	4.2464
	48	4.8780	42.3543	5.2336	42.9658	465.2544	942.9948	68.5046	91.6626	4.7217	8.7462	4.9554
	96	5.9822	49.4468	6.5662	47.9619	456.3580	942.7905	67.1452	95.4560	5.9124	10.2931	6.2052
Solar	24	0.2565	33.0573	0.2845	33.2555	267.5163	610.6117	0.4030	42.5690	0.2228	0.2766	0.3535
	48	0.3369	36.7514	0.3664	36.9959	256.4438	597.8569	1.2394	81.9621	0.3308	0.3962	0.6693
	96	0.2762	29.2612	0.2994	30.7602	255.5198	599.2925	0.4289	45.4507	0.2768	0.3383	12.5598

Table 10 reports the corresponding horizon-level ablations. These results substantiate the averaged findings in the main text and expose the contribution of each model component across forecasting regimes.

G Visualization on More Datasets

Figures 5 and 6 present forecast visualizations for ETTh2 and Weather, respectively. In both examples, the predictive mean tracks the ground-truth trajectory, while the 90% intervals adapt their width over time and represent forecast uncertainty around the central prediction.

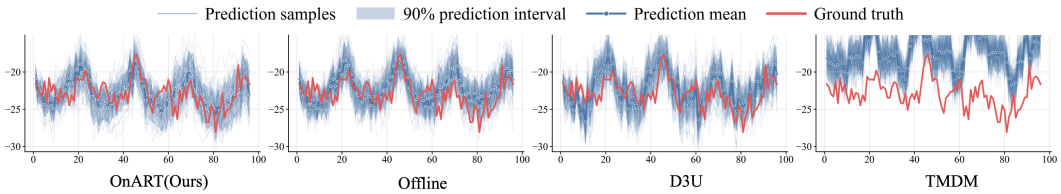


Figure 5: Forecast visualization on ETTh2, where each timestep is aggregated across all variables.

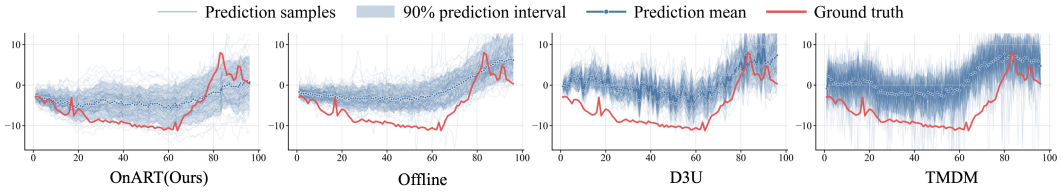


Figure 6: Forecast visualization on Weather, where each timestep is aggregated across all variables.

Table 10: Detailed ablation results under different forecasting horizons. Lower values indicate better performance for all metrics.

Dataset	Horizon	Online				Offline				w/o Start Point Assignment				w/o Codebook				Point Update Only				Naive Online Flow Matching			
		MSE	MAE	CRPS	CRPS_{sum}	MSE	MAE	CRPS	CRPS_{sum}	MSE	MAE	CRPS	CRPS_{sum}	MSE	MAE	CRPS	CRPS_{sum}	MSE	MAE	CRPS	CRPS_{sum}	MSE	MAE	CRPS	CRPS_{sum}
ETTh2	24	1.9841	0.5986	0.4765	2.3564	2.5028	0.6233	0.5011	2.6165	1.9583	0.6050	0.5257	2.5875	1.9710	0.6144	0.4851	2.3728	1.9724	0.6105	0.4897	2.4259	4.1436	0.8255	0.6526	3.2616
	48	3.2842	0.7172	0.5782	2.8945	3.7009	0.7204	0.5850	3.0695	3.2683	0.7140	0.6131	3.0891	3.2865	0.7261	0.5849	2.9086	3.2817	0.7234	0.5848	2.9561	6.0259	0.9950	0.7948	4.1991
	96	5.6313	0.8639	0.7082	3.5873	6.1417	0.8630	0.7152	3.7016	5.5895	0.8635	0.7410	3.7336	5.6191	0.8828	0.7339	3.6896	5.5967	0.8744	0.7185	3.6317	9.1166	1.1684	0.9365	5.0475
ETTm1	24	0.5044	0.4332	0.3410	1.4184	1.3175	0.6196	0.5215	2.3047	0.5013	0.4352	0.4022	1.7628	0.5317	0.4565	0.3488	1.4957	0.5185	0.4446	0.3540	1.5324	0.8169	0.5460	0.4089	1.8725
	48	0.6858	0.5086	0.4021	1.7309	1.9202	0.7786	0.6537	3.0334	0.6825	0.5047	0.4584	2.0718	0.7160	0.5292	0.4082	1.7687	0.6993	0.5179	0.4077	1.8037	1.2917	0.6981	0.5273	2.5483
	96	0.8056	0.5577	0.4409	1.9598	2.1174	0.8290	0.6892	3.3563	0.7920	0.5488	0.4819	2.2150	0.8201	0.5731	0.4458	2.0141	0.8204	0.5679	0.4450	1.9751	1.2580	0.7224	0.5406	2.5651
Weather	24	0.8366	0.4312	0.3498	3.1188	1.0003	0.4593	0.3807	3.5187	0.8181	0.4181	0.3903	3.8463	0.8470	0.4552	0.3552	3.3804	0.8412	0.4463	0.3606	3.4352	1.0587	0.5041	0.3772	4.2589
	48	1.2255	0.5821	0.4697	4.2271	1.4781	0.6179	0.5110	5.1161	1.1913	0.5644	0.5178	5.0791	1.2408	0.5919	0.4684	4.4402	1.2576	0.5917	0.4844	4.5768	1.6289	0.6914	0.5193	6.1391
	96	1.5707	0.6686	0.5463	5.4175	1.7850	0.7307	0.5976	6.0738	1.6039	0.6862	0.6166	6.2054	1.6513	0.7153	0.5695	5.8381	1.6330	0.7116	0.5812	5.8214	2.7594	0.9481	0.7148	8.8727
Exchange	24	0.0978	0.2022	0.1576	0.9420	0.1015	0.2028	0.1596	0.9549	0.1105	0.2070	0.1705	1.0489	0.1213	0.2308	0.1770	0.9833	0.1124	0.2100	0.1661	1.0183	0.1175	0.2284	0.1703	1.1065
	48	0.1958	0.2845	0.2208	1.3183	0.1971	0.2859	0.2227	1.3260	0.2032	0.2835	0.2245	1.3762	0.3148	0.3931	0.2979	1.8044	0.2068	0.2883	0.2250	1.3582	0.2471	0.3345	0.2492	1.7013
	96	0.3714	0.3841	0.2971	1.8539	0.3723	0.3851	0.2984	1.8615	0.3909	0.3827	0.3018	1.9454	0.4253	0.4077	0.3215	2.1268	0.3964	0.3894	0.3029	1.9136	0.5205	0.4646	0.3539	2.4255
ECL	24	4.2343	0.3274	0.2697	35.9336	4.3508	0.3250	0.2710	37.9931	4.2309	0.3213	0.2804	38.8221	4.3374	0.4120	0.3641	46.7549	4.2319	0.3226	0.2683	37.4440	Excessive Runtime			
	48	4.8780	0.3596	0.2953	42.3543	5.0214	0.3592	0.2971	44.6341	4.8758	0.3557	0.3077	44.6555	4.9143	0.3878	0.3175	44.3354	4.8766	0.3572	0.2950	43.2553	Excessive Runtime			
	96	5.9822	0.3823	0.3149	49.4468	6.0118	0.3976	0.3265	49.2307	5.9806	0.3790	0.3280	51.1517	6.2398	0.4502	0.3712	54.8372	5.9812	0.3802	0.3151	49.7769	Excessive Runtime			
Solar	24	0.2565	0.2927	0.2217	33.0573	0.2611	0.2935	0.2252	31.2348	0.2686	0.3276	0.2620	33.3929	0.3175	0.3602	0.2806	38.5150	0.3058	0.3570	0.2772	37.7813	Excessive Runtime			
	48	0.3369	0.3625	0.2718	36.7514	0.3775	0.3549	0.2694	38.0689	0.3275	0.3611	0.2846	36.4728	0.3469	0.3837	0.3025	40.3661	0.3431	0.3870	0.3046	40.6093	Excessive Runtime			
	96	0.2762	0.3164	0.2337	29.2612	0.2954	0.3137	0.2251	29.9975	0.2858	0.3143	0.2377	29.4077	0.2869	0.3217	0.2395	30.3610	0.2941	0.3316	0.2465	31.4505	Excessive Runtime			