Goal-Oriented Time-Series Forecasting: Foundation Framework Design

Luca-Andrei Fechete*† Mohamed Sana† Fadhel Ayed† Nicola Piovesan† Wenjie Li†
Antonio De Domenico† Tareq Si Salem†‡

April 25, 2025

Abstract

Traditional time-series forecasting often focuses only on minimizing prediction errors, ignoring the specific requirements of real-world applications that employ them. This paper presents a new training methodology, which allows a forecasting model to dynamically adjust its focus based on the importance of forecast ranges specified by the end application. Unlike previous methods that fix these ranges beforehand, our training approach breaks down predictions over the entire signal range into smaller segments, which are then dynamically weighted and combined to produce accurate forecasts. We tested our method on standard datasets, including a new dataset from wireless communication, and found that not only it improves prediction accuracy but also improves the performance of end application employing the forecasting model. This research provides a basis for creating forecasting systems that better connect prediction and decision-making in various practical applications.

*École Polytechnique, Palaiseau, France (Research Intern)
†Paris Research Center, Huawei Technologies, Boulogne-Billancourt, France
‡Lead researcher for this study. Corresponding author: tareq.si.salem@huawei.com

1 Introduction

Time-series forecasting (TSF) represents a significant area of study within machine learning (ML), with practical applications demonstrable in various domains, including but not limited to, economics [14], energy resource management [25, 30], transportation optimization [6], meteorological prediction [38], inventory optimization [19], and healthcare [11, 31]. At its core, TSF is concerned with constructing predictive models for time-dependent sequential data. This involves leveraging historical patterns and relationships within observations to forecast future data points. The methodologies employed in TSF are diverse, ranging from classical statistical approaches, such as Autoregressive Integrated Moving Average (ARIMA) [17] and Exponential Smoothing (ETS) [5], to deep learning (DL) approaches, including Multi-Layer Perceptrons (MLPs) [40], Recurrent Neural Networks (RNNs) [41], Long Short-Term Memory (LSTMs) [33], Temporal Convolutional Networks (TCNs) [15], and Transformer architectures [26, 20]. More recently, the field has witnessed the appearance of foundation Large Time-Series Models (LTSMs), pre-trained on extensive time-series datasets to enable zero-shot forecasting. These models, including Timer [23], Moirai [36], TimesFM [9], Chronos [1], Moment [12], and Toto [8] predominantly utilize variations of the Transformer architecture.

Generally, TSF methodologies prioritize the minimization of predictive error, often neglecting the integration of predicted outputs within subsequent downstream processes. In numerous downstream applications of forecasting, the practical significance of forecast errors is not uniform, and this treatment introduces suboptimal model performance with respect to the ultimately desired objective. This is effectively illustrated by forecasting challenges *IEEE-CIS Technical Challenge on Predict+Optimize for Renewable Energy Scheduling* [3] and the *M5 Accuracy Competition* [3], where evaluations based solely on forecast accuracy substantially mismatches with evaluations based on the eventual optimization objective, such as energy minimization. This necessitates the development of TSF models that explicitly incorporate downstream task objectives during development and evaluation. This integration is essential given the widespread of real world analytical systems combining predictive and optimization components. For example, Bertsimas and Kallus [4] propose learning weight functions from data for integration into optimization objectives. Similarly, Elmachtoub and Grigas [10] proposed a "Predict-then-Optimize" framework within linear programming, directly leveraging optimization structure to inform loss function design. Furthermore, the ML community has witnessed a significant trend towards end-to-end (E2E) learning paradigms, applied across domains such as finance [2], image recognition [34], robotic manipulation [18], and inventory-management [29] highlighting the potential for this approach in TSF.

A major limitation within existing literature [4, 10, 29] is the assumption of pre-defined task specifications. Specifically, these methodologies presuppose that regions of forecasting importance are both provided and static. However, numerous real-world TSF applications, such as wireless traffic prediction, necessitate adaptive approaches due to unknown and dynamically shifting importance regions. For instance, in energy efficiency policies, low-traffic periods are more important for base-station deactivation, while high-traffic periods are less relevant. Conversely, power allocation strategies may require sensitivity to both extremes, with lower importance on intermediate values. In practice, these thresholds are often not known a priori and potentially time-varying. Consequently, there is a pressing need for TSF frameworks that enable post-hoc configuration to give importance to specific regions of interest during inference. This work takes the first step, to the best of our knowledge, to address the observed gap. A motivating problem that can be tackled by our approach is illustrated in Figure 1.

Contributions. The specific contributions of this work are outlined as follows:

- This research introduces a new training methodology that modifies current transformer-based TSF

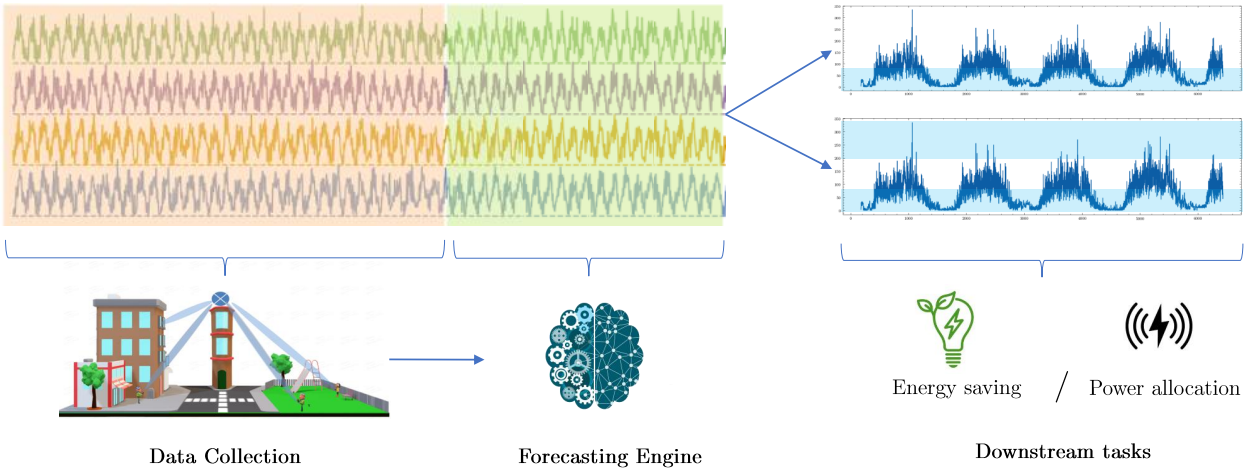


Figure 1: This figure illustrates the time-series forecasting problem in a wireless network context. The process involves data collection to gather historical time-series information, followed by the development and training of a forecasting model to predict future values. These forecasts are then used for downstream applications. In this example, we show two downstream tasks: energy efficiency policy and power allocation. For energy efficiency, accurate forecasts during low-traffic periods are crucial, as indicated by the highlighted blue interval, to optimize base-station deactivation. High-traffic periods are less critical. Power allocation, however, requires accurate forecasts across both high and low traffic periods, also highlighted in blue, with less emphasis on intermediate values. Traditional forecasting methods treat prediction as an independent problem, neglecting the specific requirements of downstream tasks. This is suboptimal because different applications have varying sensitivities to different forecast ranges. A second approach considers an E2E system tailored to a specific downstream task. This work advances this concept by enabling the forecasting model to adapt to different downstream tasks during inference.

models. These models are designed to serve as foundational architectures, enabling adaptation to multiple downstream tasks at inference-time. A comprehensive empirical evaluation, including thorough baseline comparisons and ablation studies showcase the efficacy of the proposed scheme.

- The performance of the proposed models was assessed using a synthetic trace and well as a realistic wireless mobile network measurements trace. In addition, we intend to release the wireless mobile network dataset.

Outline of Paper. This paper proceeds as follows. Section 2 reviews the related literature. Section 3 formalizes the forecasting problem. The methodology employed in this research is detailed in Section 4. Finally, Section 6 concludes the paper and outlines avenues for future research.

2 Literature Review

Time-Series Forecasting. Traditionally, statistical approaches like ARIMA [17] and ETS [5] were predominant in TSF. However, the surge in computational power and data availability has spurred the adoption of deep learning techniques. This transition encompasses MLPs [40], RNNs [41], LSTMs [33], TCNs [15], and Transformer architectures [26, 20, 37, 43, 44]. Adapting Transformers for TSF is now a key research area, with efforts focused on: (1) refining internal components, particularly attention mechanisms [37, 43, 44]; (2) transforming input token representations using techniques like

stationarization [22], channel independence [26], and patching [26, 20]; and (3) broadly modifying the Transformer architecture and its modules [42]. Recent advances include foundation LTSMs like Timer [23], Moirai [36], TimesFM [9], Chronos [1], Moment [12], and Toto [8] which leverage pre-training and Transformer variants for zero-shot forecasting. This research extends state-of-the-art transformer architectures by integrating a task-specific configuration mechanism, enabling dynamic adaptation of model predictions during inference.

Task-Aware Forecasting. Prior research has explored customizations of the objective function for TSF, through loss reshaping or re-weighing [28, 39, 7, 16]. These works primarily aim to integrate uncertainty quantification or fairness considerations, diverging from the objective of tailoring models to specific downstream tasks. Conversely, other research have concentrated on developing frameworks that derive loss weights from downstream task characteristics, thereby guiding the training process of forecasting models [3, 10, 13]. While our methodology shares conceptual similarities with this latter line of work, specifically in the incorporation of loss shaping during training, it distinguishes itself by proposing a training methodology that enables the same model to accommodate diverse downstream task requirements at inference-time.

3 The Forecasting Problem

Time-Series. A multivariate time-series is a sequence of vector-valued observations ordered temporally. Let $\mathbf{x}_t \in \mathbb{R}^n$ denote a n-dimensional real-valued vector observed at time t, where $t \in [T]$ and $n \geq 1$. The full time-series is represented as the ordered set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, where each $\mathbf{x}_t = (x_{t,i})_{i\in[n]}$ corresponds to measurements of n variables at time t. The temporal dependency between observations is intrinsic to the series, with $\mathbf{x}_t$ potentially influenced by past values $\mathbf{x}_{t-k}$ for some lag $k > 0$. We assume each series $i \in [n]$ lies within a bounded set $\mathcal{X} \subset \mathbb{R}$, where $|\mathcal{X}| < \infty$.

The Learning Problem. Time-series forecasting is reformulated as a supervised regression task by constructing input-output pairs from sliding windows over the series. Let $\tau \in \mathbb{N}$ be the forecast horizon and $w \in \mathbb{N}$ be the window size. In practice, the window size w is treated as a hyperparameter. For each time t, the input $\mathbf{X}_t = \mathbf{x}_{t-w:t-1} \in \mathcal{X}^{w\times n}$ consists of a history window $\{\mathbf{x}_{t-w}, \ldots, \mathbf{x}_{t-1}\}$, and the target $\mathbf{Y}_t = \mathbf{x}_{t:t+\tau-1} \in \mathcal{X}^{\tau\times n}$ is the future window $\{\mathbf{x}_t, \ldots, \mathbf{x}_{t+\tau-1}\}$. The mapping $f_{\boldsymbol{\theta}} : \mathcal{X}^{w\times n} \to \mathcal{X}^{\tau\times n}$, parameterized by $\boldsymbol{\theta}$, is learned to approximate the underlying dynamic $\mathbf{Y}_t = f(\mathbf{X}_t) + \boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \in \mathbb{R}^{\tau\times n}$ is noise and f is some unknown dynamic. This generates a dataset $D = \{(\mathbf{X}_t, \mathbf{Y}_t)\}_{t=w}^{T-\tau}$. The goal is to learn the model $\boldsymbol{\theta}_\star \in \mathbb{R}^d$ for $d \geq 1$ that minimizes the expected loss over the data distribution $\mathcal{D}(\mathbf{X}, \mathbf{Y})$:

$$\boldsymbol{\theta}_\star \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim\mathcal{D}} \left[L(f_{\boldsymbol{\theta}}(\mathbf{X}), \mathbf{Y}) \right], \tag{1}$$

where $L : \mathbb{R}^{\tau\times n} \times \mathbb{R}^{\tau\times n} \to \mathbb{R}$ is a differentiable loss function. In practice, this expectation is approximated by the empirical risk over D:

$$\tilde{\boldsymbol{\theta}}_\star \in \arg\min_{\theta} \frac{1}{|D|} \sum_{(\mathbf{X}_t,\mathbf{Y}_t)\in D} L(f_{\boldsymbol{\theta}}(\mathbf{X}_t), \mathbf{Y}_t) + R(\boldsymbol{\theta}), \tag{2}$$

The regularization term $R : \mathbb{R}^d \to \mathbb{R}$ (e.g., the Euclidean distance $R(\boldsymbol{\theta}) = \lambda\|\boldsymbol{\theta}\|_2^2$) is incorporated to control the complexity of the model parameters $\boldsymbol{\theta}$, and can be thought of as a model's complexity score. Regularization aims to prevent the memorization of noise within the training data, enhancing the model's ability to generalize to novel data [32]. Among the most commonly used loss functions are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE), each of which serves a distinct

statistical purpose. The MSE loss function is defined as: $L_{\text{MSE}}(\hat{\mathbf{Y}}, \mathbf{Y}) \triangleq \frac{1}{n\tau}\|\hat{\mathbf{Y}} - \mathbf{Y}\|_2^2$. MSE assigns a quadratic penalty to deviations between the predicted values $\hat{\mathbf{Y}}$ and the true target values $\mathbf{Y}$. Due to this quadratic nature, larger errors contribute disproportionately to the loss, making MSE sensitive to outliers. From a statistical perspective, minimizing MSE leads to an estimator that approximates the *conditional expectation* of the target variable given the input features: $\hat{\mathbf{Y}}_{\text{MSE}} = \mathbb{E}[\mathbf{Y} \mid \mathbf{X}]$. Alternatively, the MAE loss function is defined as: $L_{\text{MAE}}(\hat{\mathbf{Y}}, \mathbf{Y}) \triangleq \frac{1}{n\tau}\|\hat{\mathbf{Y}} - \mathbf{Y}\|_1$. Unlike MSE, the MAE loss applies a linear penalty to deviations, making it more robust to outliers. The optimal estimator under MAE minimizes the sum of absolute residuals, which corresponds to the *conditional median* of the target variable given the input: $\hat{\mathbf{Y}}_{\text{MAE}} = \text{Median}(\mathbf{Y} \mid \mathbf{X})$.

4 Methodology

This section describes the different approaches we investigate to address the challenge of adapting to varying intervals at inference time. Our investigation progresses through five stages: (1) a baseline policy, neglecting forecasting intervals; (2) task-specific policy, optimized for an individual interval; (3) a uniform interval policy, incorporating a predefined uniform distribution of interval endpoints; (4) a discrete interval policy, incorporating a set of predefined intervals; and (5) an adaptive interval policy, adapting to any interval at inference time. The adaptive policy leverages discrete interval training and an add-on classifier to selectively utilize interval-specific forecasts, achieving flexibility across arbitrary target intervals.

Baseline Policy (`B-Policy`). This policy follows the baseline training procedure detailed in Section 3. This training approach does not incorporate interval sensitivity, and the loss is computed with respect to the forecasting target over the domain $\mathcal{X}$.

Task-specific Policy (`E2E-Policy`). Given a target interval $\mathcal{I} \subseteq \mathcal{X}$, the task-specific policy only considers a forecasting target within this range. In particular, this policy can be considered as a variation of the baseline policy that is limited to consider a forecasting loss solely over the intervals of interest. Our goal is to learn the model $\boldsymbol{\theta}^\star_{\text{Task}} \in \mathbb{R}^d$ that minimizes the interval-specific expected loss over a data distribution $\mathcal{D}(\mathbf{X}, \mathbf{Y})$:

$$\boldsymbol{\theta}^\star_{\text{Task}} \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim\mathcal{D}} \left[L\left(f_{\boldsymbol{\theta}}(\mathbf{X}), \mathbf{Y}\right) \cdot \mathbb{1}\left(\mathbf{Y} \in \mathcal{I}^{\tau\times n}\right)\right], \tag{3}$$

where $\mathbb{1}(\chi) \in \{0, 1\}$ is the indicator function which is equal to 1 when condition χ is true. Again, the expectation can be approximated by the interval-specific empirical risk over some dataset D consisting of samples from $\mathcal{D}$:

$$\tilde{\boldsymbol{\theta}}^\star_{\text{Task}} \in \arg\min_{\theta\in\mathbb{R}^d} \frac{1}{|D|} \sum_{(\mathbf{X}_t,\mathbf{Y}_t)\in D} L\left(f_{\boldsymbol{\theta}}(\mathbf{X}), \mathbf{Y}\right) \cdot \mathbb{1}\left(\mathbf{Y} \in \mathcal{I}^{\tau\times n}\right) + R(\boldsymbol{\theta}). \tag{4}$$

Continuous-Interval Training Policy (`C-Policy`). To allow the forecasting model to differentiate between any target intervals within $\mathcal{X}$, we incorporate the interval as a covariate. A covariate interval $\mathcal{I} \subseteq \mathcal{X}$ is represented as a two-dimensional vector in $\mathcal{X}^2$ containing its boundary values (e.g., a vector encoding $(I_{\min}, I_{\max}) \in \mathcal{X}^2$, for some interval $[I_{\min}, I_{\max}] \subseteq \mathcal{X}$). This enables the model to learn the relationship between varying intervals and its predictive outputs. Specifically, the learned mapping $f_{\boldsymbol{\theta}}$ becomes a function of both the input time-series $\mathbf{X}$ and the interval $\mathcal{I}$, i.e., the mapping $f_{\boldsymbol{\theta}} : \mathcal{X}^{w\times n+2} \to \mathcal{X}^{\tau\times n}$. During training, we sample intervals uniformly at random over the set $\mathcal{U}_\delta = \{\mathcal{I} \subset \mathcal{X} : |\mathcal{I}| \geq \delta\}$, where δ represents the minimum length of the sampled intervals. The minimal

distance constraint is introduced to make the training more stable, because small intervals contain less samples introducing high variance. This is further demonstrated numerically in Figure 4c. Consequently, for a data sample $(\mathbf{X}_t, \mathbf{Y}_t) \sim \mathcal{D}$ and its corresponding sampled interval $\mathcal{I}_t$, our custom loss function is defined as:

$$L_{\text{Cont}}(f_{\boldsymbol{\theta}}(\mathbf{X}_t, \mathcal{I}_t), \mathbf{Y}_t, \mathcal{I}_t) = L\left(f_{\boldsymbol{\theta}}(\mathbf{X}_t, \mathcal{I}_t), \mathbf{Y}_t\right) \cdot \mathbb{1}(\mathbf{Y}_t \in \mathcal{I}_t^{\tau \times n}). \tag{5}$$

Thus, our training objective is to learn the model parameters $\boldsymbol{\theta}^\star_{\text{Cont}} \in \mathbb{R}^d$ $(d \geq 1)$ that minimize the interval-uniform expected loss over the data distribution $\mathcal{D}(\mathbf{X}, \mathbf{Y})$ and the interval distribution $\mathcal{U}_\delta$:

$$\boldsymbol{\theta}^\star_{\text{Cont}} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathbb{E}_{\mathcal{I} \sim \mathcal{U}_\delta} \left[\mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathcal{D}} \left[L_{\text{Cont}}(f_{\boldsymbol{\theta}}(\mathbf{X}, \mathcal{I}), \mathbf{Y}, \mathcal{I}) \mid \mathcal{I} \right] \right]. \tag{6}$$

Similar to the previous policies, the above problem is approximated by minimizing the empirical risk over the dataset D and the interval distribution $\mathcal{U}_\delta$:

$$\tilde{\boldsymbol{\theta}}^\star_{\text{Cont}} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{|D|} \sum_{(\mathbf{X}_t, \mathbf{Y}_t) \in D} L_{\text{Cont}}(f_{\boldsymbol{\theta}}(\mathbf{X}_t, \mathcal{I}_t), \mathbf{Y}_t, \mathcal{I}_t) + R(\boldsymbol{\theta}), \tag{7}$$

where $\mathcal{I}_t$ is the sampled interval from $\mathcal{U}_\delta$ for each $(\mathbf{X}_t, \mathbf{Y}_t) \in D$.

Discretized-Interval Training Policy (`D`$_L$`-Policy`). This policy closely resembles the `C-Policy`, except that the support space of the distribution of possible interval is severely reduced. In particular, the interval selection process now involves sampling an interval from a discrete distribution, denoted by $\mathcal{C}_L$, defined over a set of L disjoint intervals that collectively cover the interval $\mathcal{X}$, i.e., for a given L the support $\text{supp}\left(\mathcal{C}_L\right)$ satisfies $\dot{\bigcup}_{\mathcal{I} \in \text{supp}(\mathcal{C}_L)} = \mathcal{X}$ where $\dot{\bigcup}$ denotes the union of disjoint sets. The loss function is then calculated for a particular sampled interval $\mathcal{I}_t \sim \mathcal{C}_L$ and data sample $(\mathbf{X}_t, \mathbf{Y}_t) \sim \mathcal{D}$ as:

$$L_{\text{Disc}_L}(f_{\boldsymbol{\theta}}(\mathbf{X}_t, \mathcal{I}_t), \mathbf{Y}_t, \mathcal{I}_t) = L(f_{\boldsymbol{\theta}}(\mathbf{X}_t, \mathcal{I}_t), \mathbf{Y}_t, \mathcal{I}_t) \cdot \mathbb{1}(\mathbf{Y}_t \in \mathcal{I}_t^{\tau \times n}) \tag{8}$$

The training objective remains to learn the model parameters $\boldsymbol{\theta}^\star_{\text{Disc}_L} \in \mathbb{R}^d$ $(d \geq 1)$ that minimize the expected loss, now considering the discrete distribution $\mathcal{C}_L$:

$$\boldsymbol{\theta}^\star_{\text{Disc}_L} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathbb{E}_{\mathcal{I} \sim \mathcal{C}_L} \left[\mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathcal{D}} \left[L_{\text{Disc}_L}(f_{\boldsymbol{\theta}}(\mathbf{X}, \mathcal{I}), \mathbf{Y}, \mathcal{I}) \mid \mathcal{I} \right] \right]. \tag{9}$$

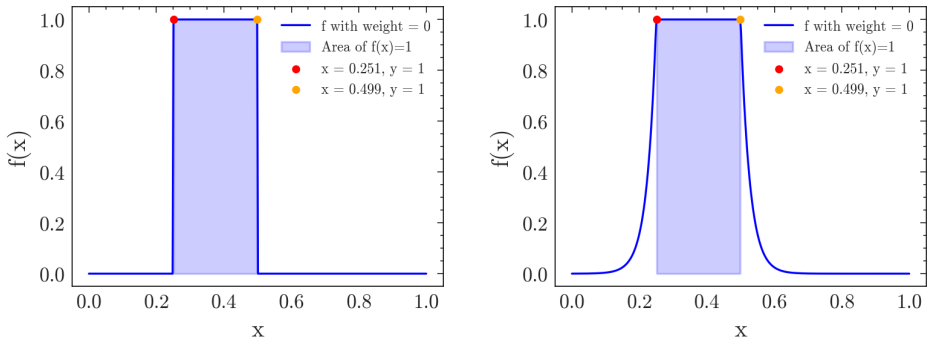
This objective is approximated empirically using the dataset D and the interval distribution $\mathcal{C}_L$:

$$\tilde{\boldsymbol{\theta}}^\star_{\text{Disc}_L} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{|D|} \sum_{(\mathbf{X}_t, \mathbf{Y}_t) \in D} L_{\text{Disc}_L}(f_{\boldsymbol{\theta}}(\mathbf{X}_t, \mathcal{I}_t), \mathbf{Y}_t, \mathcal{I}_t) + R(\boldsymbol{\theta}), \tag{10}$$

where $\mathcal{I}_t$ is the sampled interval from $\mathcal{C}_t$ for each $(\mathbf{X}_t, \mathbf{Y}_t) \in D$.

Patching-Augmented Discretized Training Policy (`D`$^\star_L$`-Policy`). The premise behind the design of this policy is to consider the `D`$_L$`-Policy` configured with a sufficiently large L corresponding to a finer discretization of the domain $\mathcal{X}$. At inference time when requested with an unknown interval $\mathcal{I}$, the policy combines the constituent intervals to make the predictions. This is made possible with two main modifications. First, the loss function incorporates a decay function displayed in Figure 2. This decay function modulates the contribution of each sample to the loss based on its distance from the center of its associated interval. This decay mechanism is defined as

$$d_\nu(y, \mathcal{I}) = \exp\left(-\nu \cdot \max\left(0, |y - \Delta_{\text{avg}}| - \Delta_{\text{diff}}\right)\right), \tag{11}$$



(a) The weighing mechanism for values inside interval [0.25, 0.5] with a large decay rate $\nu \to \infty$.

(b) The weighing mechanism for values inside interval [0.25, 0.5] with decay rate $\nu = 37$.

Figure 2: This figure depicts the decay function with varying decay rates, illustrating how different rates influence the smoothness of the transition in importance for values outside the interval of interest.

where ν is the decay rate, and $\Delta_{\text{avg}} = \frac{\max(\mathcal{I})+\min(\mathcal{I})}{2}$ is the midpoint of an interval and $\Delta_{\text{diff}} = \frac{\max(\mathcal{I})-\min(\mathcal{I})}{2}$ is half of the length of the interval. When the decay rate is large $\nu \to \infty$, the weight $\prod_{i\in[n],t\in[\tau]} d_\nu(y_i, \mathcal{I}) \in [0,1]$ converges to the indicator function $\mathbb{1}\left(\mathbf{Y} \in \mathcal{I}^{n\times\tau}\right) \in \{0,1\}$ in Eq. (8). This weighing mechanism allows the model to learn values at the boundary of the interval of interest, and it can also be viewed as an introduction of a soft overlap across the different intervals. This soft boundary is needed to reduce errors during the patching process of different intervals, since boundary uncertainties can introduce errors. Second, a classification head is appended to the model. This addition enables the model to predict whether the true value falls within or outside the interval of interest. Using the same notation as in the previous policy, the classifier introduces an additional mapping $f^c_{\boldsymbol{\theta}} : \mathcal{X}^{w\times n+2} \to [0,1]^{\tau\times n}$ parameterized by a *shared*$\boldsymbol{\theta}$, the learning objective for the augmented model parameters, now encompassing the classifier, defined as follows:

$$\boldsymbol{\theta}^\star_{\text{Disc}^\star_L} \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{\mathcal{I}\sim\mathcal{C}_L}\left[\mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim\mathcal{D}}\left[L(f_{\boldsymbol{\theta}}(\mathbf{X},\mathcal{I}),\mathbf{Y},\mathcal{I}) + \phi\cdot L'(f^c_{\boldsymbol{\theta}}(\mathbf{X},\mathcal{I}),\mathbb{1}(\mathbf{Y}\in\mathcal{I}),\mathcal{I}) \mid \mathcal{I}\right]\right], \tag{12}$$

where $\phi \in [0,1]$ is a hyperparameter introduced to balance the importance of the regression and classification tasks and the loss $L' : \mathbb{R}^{\tau\times n}\times\mathbb{R}^{\tau\times n} \to \mathbb{R}$ is a binary classification loss (e.g., Cross-Entropy Loss). Note that both losses are scaled by the decay function $\prod_{i\in[n],t\in[\tau]} d_\nu(y_i,\mathcal{I})$ in (11) instead of the indicator function $\mathbb{1}\left(\mathbf{Y}\in\mathcal{I}^{n\times\tau}\right)$ in (8), configured with the decay rate ν.

This objective is approximated empirically using the dataset D and the interval distribution $\mathcal{C}_L$:

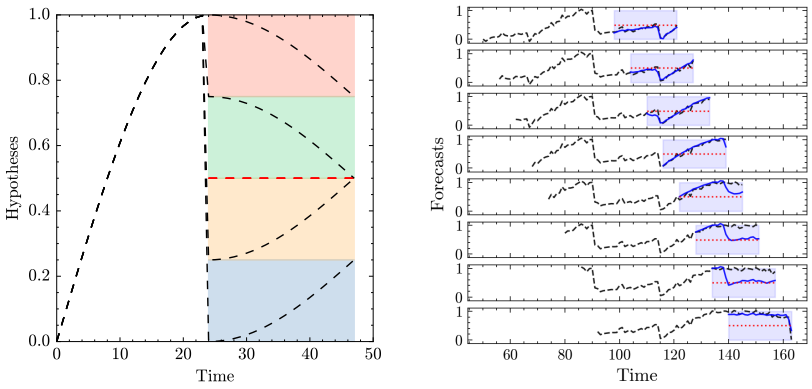
$$\tilde{\boldsymbol{\theta}}^\star_{\text{Disc}^\star_L} \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{|D|}\sum_{(\mathbf{X}_t,\mathbf{Y}_t)\in D}\left(L(f_{\boldsymbol{\theta}}(\mathbf{X}_t,\mathcal{I}_t),\mathbf{Y}_t,\mathcal{I}_t) + \phi\cdot L'(f^c_{\boldsymbol{\theta}}(\mathbf{X}_t,\mathcal{I}_t),\mathbb{1}(\mathbf{Y}_t\in\mathcal{I}_t),\mathcal{I}_t)\right) + R(\boldsymbol{\theta}), \tag{13}$$

where $\mathcal{I}_t$ is the sampled interval from $\mathcal{C}_t$ for each $(\mathbf{X}_t,\mathbf{Y}_t)\in D$.

Patching Mechanism. Given an arbitrary interval $\mathcal{I}$, we first identify the subset of training intervals that intersect with $\mathcal{I}$, given by the mapping

$$\Xi_L(\mathcal{I}) \triangleq \left\{\mathcal{I}' \in \text{supp}\left(\mathcal{C}_L\right) : \mathcal{I}'\cap\mathcal{I}\neq\emptyset\right\}, \tag{14}$$

where $\text{supp}\left(\mathcal{C}_L\right)$ is the support of the distribution $\mathcal{C}_L$ of size L. The logic behind this is that when L is large enough, the union of intervals in $\Xi_L(\mathcal{I})$ is a good approximation of $\mathcal{I}$. For an input series $\mathbf{X}$ and target interval $\mathcal{I}$ the patching mechanism strategies are defined as follows.



(a) Noise-free hypotheses in `SynthDS`

(b) Averaging tendencies of baseline model

Figure 3: Subfigure (a) depicts the noise-free hypotheses used to construct the synthetic trace `SynthDS`. Subfigure (b) depicts the inability of the baseline policy `B-Policy` to distinguish between the different patterns in `SynthDS` dataset, as it predicts the average hypothesis (red line at 0.5). The model used is a trained iTransformer. Each subplot shows a time-shift of six steps. The black dotted line indicates the true values. The blue line shows the model's predictions. The shaded region depicts the forecasting horizon.

- Averaging Strategy (*1-strategy*). This strategy computes a weighted average of the regression predictions for all intersecting training intervals, weighted by their respective classification probabilities for the given input and the target interval $\mathcal{I}$. Formally:

$$\hat{\mathbf{Y}} = \frac{\sum_{\mathcal{I}' \in \Xi_L(\mathcal{I})} f_\theta^c(\mathbf{X}, \mathcal{I}') f_\theta(\mathbf{X}, \mathcal{I}')}{\sum_{\mathcal{I}' \in \Xi_L(\mathcal{I})} f_\theta^c(\mathbf{X}, \mathcal{I}')}. \tag{15}$$

- Maximum Confidence Strategy (∞-*strategy*). This strategy selects the regression prediction corresponding to the training interval with the highest classification probability for the given input and the target interval $\mathcal{I}$. Formally:

$$\hat{\mathbf{Y}} = f_\theta \left(\mathbf{X}, \operatorname*{argmax}_{\mathcal{I}' \in \Xi_L(\mathcal{I})} f_\theta^c(\mathbf{X}, \mathcal{I}') \right).$$

5 Experiments

This section presents the numerical results that demonstrate the performance of our training methodology with a comparison to several baselines. We start by describing the experimental setup, including the datasets used, evaluation metrics, and model configurations. Finally, we present a comprehensive analysis of the results, comparing the accuracy of different models and training methodologies.[1]

5.1 Experimental Setup

Time-Series Forecasting Models. We utilize four state-of-the-art models: iTransformer [21], DLinear [40], PatchTST [27], and TimeMixer [35]. Additionally, we propose modified variants of these architectures to integrate target interval information. For the iTransformer framework, the

[1]The code is publicly available github.com/netop-team/gotsf.

vectorized representation of the interval $\mathcal{I}$ was concatenated directly to the temporal encoding tensor as supplementary covariate features. In contrast, for all other models, the $\mathcal{I}$ values were processed as two auxiliary temporal channels appended to the primary multivariate input tensor. We also adapt these regression-oriented architectures for our dual-task objective for which we implemented a hybrid output structure. Each model was augmented with a classification head by doubling the dimensionality of the final projection layer τ which corresponds to the target forecasting horizon. The expanded output tensor was split such that the first τ dimensions generated the regression forecast, while the remaining τ dimensions produced classification logits. This architectural adaptation enables the model to perform forecasts and probabilistically assess whether forecasts reside within the specified target interval $\mathcal{I}$.

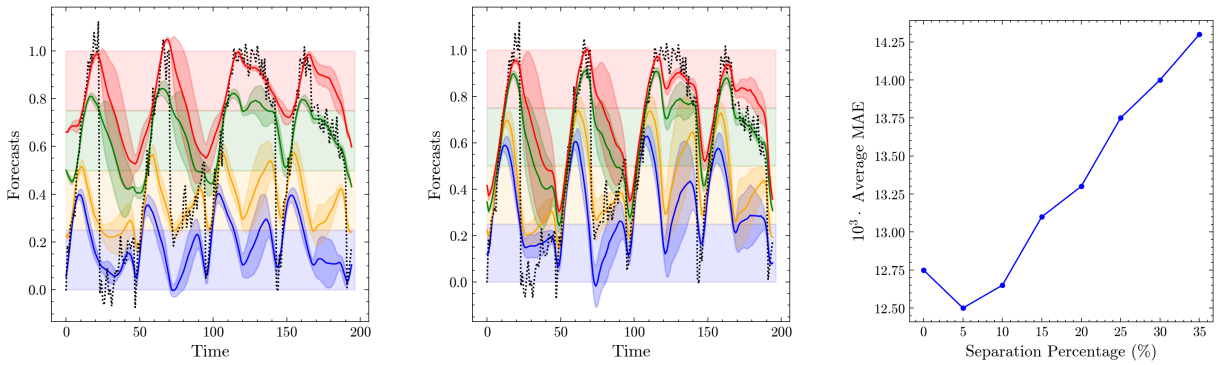
Training Policies. We examine diverse training methodologies to evaluate the adaptability of TSF models across heterogeneous downstream tasks with different prediction intervals. The following training policies, comprehensively described in Section 4, are analyzed: (a) `B-Policy` implements task-agnostic training policy without downstream optimization, serving as a baseline for comparative evaluation. (b) `E2E-Policy` is the task-specific training policy that only trains a model within an interval of interest in an end-to-end Fashion. (c) `C-Policy` is the uniform training policy employing δ-uniform sampling across task intervals to ensure minimum separation constraints. (d) $\texttt{D}_L\texttt{-Policy}$ is the discretized training policy sampling u.a.r. from L partitioned target intervals. (e) $\texttt{D}^{\star}_{L}\texttt{-Policy}$ augments the $\texttt{D}_L\texttt{-Policy}$ through integration of interval patching techniques where $\star \in \{1, \infty\}$ denotes the strategies average or maximum, respectively.

Benchmarking Datasets. In our experimental evaluation, we employ several benchmark datasets to substantiate our claims. This includes a synthetic dataset that we construct, and a wireless dataset that we will release to the public domain.

- `SynthDS` is a synthetic dataset used to highlight the various components of our proposed methodology. We construct the trace by combining an input signal $\mathbf{s} = \left(\sin\left(\frac{\pi n}{2D}\right)\right)_{n\in[D]}$, where $D = 24$ represents the length of the signal, and a corresponding output signal selected uniformly at random (u.a.r.) from the set $\left\{0.25\left(\sin\left(\frac{\pi}{2D}n + \frac{\pi}{2}\right) + k\right)_{n\in[D]} : k \in [4]\right\}$. To generate a trace spanning $T = 1.5 \times 10^3$ timesteps, we concatenate the same randomly selected signal multiple times. Subsequently, we introduce Gaussian noise to the trace, where the noise has a mean of 0 and a standard deviation of 0.05. A noise-free version of this trace is depicted in Figure 3a.
- `BLW-TrafficDS` is released to the public domain as part of this study.[2] The dataset comprises a wireless beam-level traffic with four modalities (throughput volume and time, physical resource block utilization, and user count) and was used as a benchmark for Spatio-Temporal Beam-Level Traffic Forecasting Challenge [45].

Training Configuration. To ensure our experimental is rigorous and reproducible, we describe all the training parameters used in our training framework. We partitioned the datasets into training, validation, and testing sets. For the `BLW-TrafficDS` dataset, we used a 70-10-20 split, while for the `SynthDS` dataset, we employed a 66-17-17 split. We evaluated four state-of-the-art time series forecasting models: iTransformer, DLinear, PatchTST, and TimeMixer. These models were applied the forecasting task. The `BLW-TrafficDS` dataset was processed using a multivariate-to-multivariate approach, whereas the `SynthDS` dataset utilized a univariate-to-univariate configuration. For the `BLW-TrafficDS` dataset, the input sequence length was 96 time steps, and the forecasting horizon was 24 time steps. For the

[2]The dataset is publicly released huggingface.co/datasets/netop/Beam-Level-Traffic-Timeseries-Dataset.



(a) Models trained with `E2E-Policy` on four intervals of interests.

(b) The `C-Policy` evaluated on the four different intervals associated to the four different hypotheses of our `SynthDS` dataset.

(c) The impact of the interval-separation percentage (δ) of our `C-Policy` on the MAE.

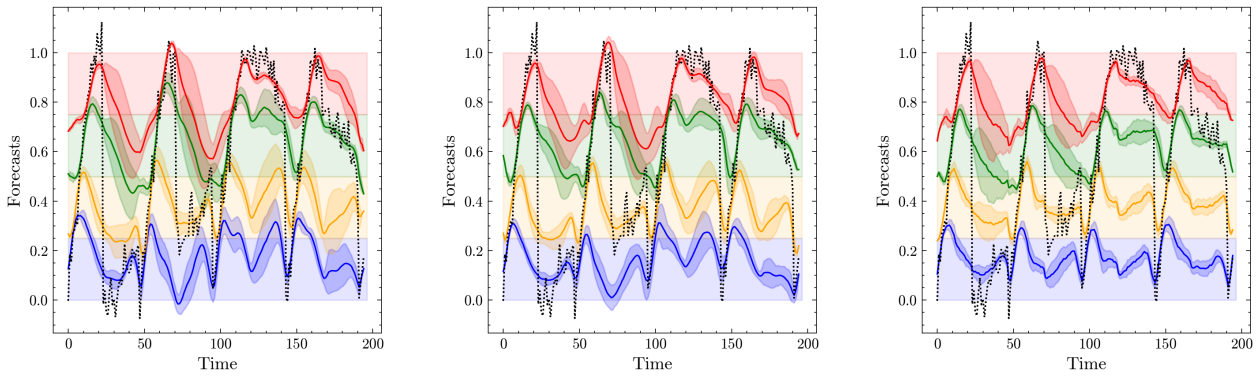
Figure 4: Subfigures (a) and (b) depict the performance fo the `E2E-Policy` and `C-Policy`, respectively. Subfigure (c) explores the `C-Policy`'s additional hyperparameter (δ). The black dotted lines are the target forecast values. Subsequently, the predictions are represented through each time step's mean value, highlighted by the solid lines, and the the shaded area around the mean represents the standard deviation. The different intervals of interest are depicted as colored regions (red, green, yellow, and blue).

`SynthDS` dataset, the input sequence length was 48 time steps, and the forecasting horizon was 24 time steps. TimeMixer was configured with 100 channel dimensions. iTransformer and other applicable models were configured with a dimension of 128, 4 attention heads, a 2-layer shallow encoder, and a feed-forward network dimension of 128. All models were trained using a batch size of 32. For regression tasks, we used the Mean Absolute Error (MAE) loss functions unless specified otherwise. For classification tasks, we employed Binary Cross Entropy loss. The AdamW optimizer [24] was used with an initial learning rate of 10^{-3}. A cosine annealing learning rate schedule was implemented, defined as: $\eta_t = \eta_{\min} + 0.5(\eta_{\max} - \eta_{\min})(1 + \cos(\pi \cdot t/n_{\text{epochs}}))$ where $\eta_{\min} = 10^{-5}$. All models were trained for 60 epochs. For the `SynthDS` dataset, $\texttt{D}^{\star}_{L}$`-Policy` was evaluated with 4 and 8 intervals. For the `BLW-TrafficDS` dataset, 8 and 16 intervals were used. To mitigate overfitting, we implemented early stopping and checkpoint saving. The validation loss was computed as the average loss of the model evaluated on each training interval, with a patience of 5 epochs. Interval endpoint sampling was performed for each sample within a batch and integrated into the model during forward propagation, as detailed in Section 4.

5.2 Numerical Results

Averaging Tendencies. In this study, we investigate the averaging behavior exhibited by the baseline policy over the `SynthDS` trace. The policy, denoted as `B-Policy`, is trained using the `SynthDS` trace. As illustrated in Figure 3b, the policy converges to an averaged hypothesis that aligns with the midpoint (1/2) of the trace. This highlights a critical limitation of the policy, namely its insensitivity to the interval of interest.

End-to-End Training. When downstream tasks define an interval of interest, the training focuses on errors or deviations from the true value within that specific interval over the `SynthDS` trace. In Figure 4a, we evaluate `E2E-Policy`, which is configured to target a particular interval of interest. This



(a) Forecasts of iTransformer model trained using the $\texttt{D}_4\texttt{-Policy}$.

(b) Forecasts of iTransformer model trained and evaluated using the $\texttt{D}_8^1\texttt{-Policy}$.

(c) Forecasts of iTransformer model trained and evaluated using the $\texttt{D}_8^\infty\texttt{-Policy}$.

Figure 5: Qualitative comparison of the $\texttt{D}_4\texttt{-Policy}$, $\texttt{D}_8^\infty\texttt{-Policy}$ and $\texttt{D}_8^1\texttt{-Policy}$ training schemes used to train iTransformer model on `SynthDS` dataset. Patching was performed by combining eight intervals to four.

approach illustrates an end-to-end training method designed to adjust the forecasting loss function, ensuring greater emphasis on intervals relevant to the downstream task. The intervals of interest within the set $\{\mathcal{I}_i : i \in [4]\} \triangleq \{[(i-1), i/4] : i \in [4]\}$ correspond to distinct hypotheses. This policy acts as an optimal benchmark, demonstrating the potential performance of a policy capable of adapting to any interval and being configurable during inference-time.

Continuous Exploration of Intervals. We assess the proposed policy described in Section 4 using the `SynthDS` trace. The results, depicted in Figure 4b, reveal that this training methodology fails to accurately capture the relationship between the intervals of interest and their associated hypotheses. The suboptimal performance highlights the inherent challenges of the learning task, even within a synthetic environment. Specifically, the model is required to learn a mapping from every possible interval within the interval $\mathcal{I} \subset \mathcal{X}$ to its corresponding hypothesis. Furthermore, in Figure 4c, we examine the effect of narrowing the support space of the intervals of interest during training by modifying the minimum separation between intervals (i.e., imposing the constraint $|\mathcal{I}| \geq \delta$). We observe that while model performance improves as the sampling space support is reduced, beyond a certain threshold δ', performance deteriorates. This degradation occurs due to overshooting the true length of the underlying intervals, introducing bias into the model.

Targeted Exploration of Intervals. Building on insights from the previous experiment, which revealed inefficiencies in capturing the underlying time-series, we explore a discrete policy ($\texttt{D}_L\texttt{-Policy}$) in Figure 5a over the `SynthDS` trace. This policy uniformly samples from a predefined set of intervals of interest, with the key distinction that the set of intervals supported at inference time is now finite and significantly smaller. We first test this approach using the optimal benchmark `E2E-Policy` capable of perfectly distinguishing different patterns in the signal. The results show that this policy achieves performance comparable to `E2E-Policy`, while offering the added advantage of adaptability to various downstream tasks at inference time. However, a practical limitation arises from the assumption that the set of intervals, or even their number, is known a priori. This limitation motivates the next experiment, where we overshoot the number of true intervals and demonstrate that the true signal can be reconstructed.

Model	Training Strat.	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$	$\mathcal{I}_8$	Avg.
DLinear	$\mathtt{D}_8$-Policy	167.1	76	56.7	44.2	33.5	28.1	19.3	11.1	**54**
	$\mathtt{D}^1_{16}$-Policy	172.3	**74.9**	**56**	43.8	33.4	27.9	19.2	11.1	54.8
	$\mathtt{D}^\infty_{16}$-Policy	164.9	75.7	56.7	44.4	33.8	28.2	19.4	11.2	54.3
	B-Policy	**163.8**	82.4	62	48.9	37.6	31.8	21.7	12.5	57.6
	$\mathtt{C}_{0.1}$-Policy	218	84.8	58.8	**43.4**	**31.7**	**26.4**	**18.8**	**10.9**	61.6
	Improvement	0.0%	9.1%	9.7%	11.2%	15.7%	16.9%	13.4%	12.8%	6.3%
TimeMixer	$\mathtt{D}_8$-Policy	127.5	40.1	24.3	16.1	10.8	8	**5.5**	**3.3**	**29.4**
	$\mathtt{D}^1_{16}$-Policy	148.1	**37.8**	**22.9**	**15.4**	**10.2**	**7.8**	5.5	3.5	31.4
	$\mathtt{D}^\infty_{16}$-Policy	138	41.7	23.8	15.8	10.4	8	5.6	3.4	30.9
	B-Policy	**125.9**	83	64.9	48.1	35.8	30.2	21.8	12.1	52.7
	$\mathtt{C}_{0.1}$-Policy	143.7	50.7	32	20.8	13.1	9.3	6.1	3.6	34.9
	Improvement	0.0%	54.5%	64.7%	68.0%	71.5%	74.2%	74.8%	72.7%	44.2%
iTransformer	$\mathtt{D}_8$-Policy	**119.3**	**74.7**	57.1	44	31.4	25.2	19.2	**11.2**	**47.8**
	$\mathtt{D}^1_{16}$-Policy	121.1	75.9	58.4	44.6	32.1	25.7	19.5	11.2	48.6
	$\mathtt{D}^\infty_{16}$-Policy	122.8	75.7	57.6	44.1	31.9	25.7	19.5	11.2	48.6
	B-Policy	130.8	77.3	59.5	46.8	35.5	29.8	20.6	11.8	51.5
	$\mathtt{C}_{0.1}$-Policy	133.1	75.6	**56**	**42.1**	**28.7**	**23.3**	**18.1**	11.4	48.5
	Improvement	8.8%	3.4%	5.9%	10.0%	19.2%	21.8%	12.1%	5.1%	7.2%
PatchTST	$\mathtt{D}_8$-Policy	**102.8**	31.1	18.3	11.4	7.1	4.5	2.7	1.7	**22.4**
	$\mathtt{D}^1_{16}$-Policy	158.2	**30.4**	**17.2**	**10.4**	**6.5**	**4**	**2.4**	**1.5**	28.8
	$\mathtt{D}^\infty_{16}$-Policy	117.8	35.3	20.8	12.6	7.9	5	2.8	1.8	25.5
	B-Policy	124.1	77.9	60.3	47.2	34.9	29.4	20.4	11.7	50.7
	$\mathtt{C}_{0.1}$-Policy	118.1	34.6	22.5	15.2	9.8	6.8	3.9	2.6	26.7
	Improvement	17.2%	60.9%	71.5%	78.0%	81.4%	86.3%	88.2%	87.2%	55.8%

Table 1: MAE values ($\times 10^3$) for the four different models (DLinear, TimeMixer, iTransformer, and PatchTST) trained with the four policies ($\mathtt{D}_8$-Policy, $\mathtt{D}^1_{16}$-Policy, $\mathtt{D}^\infty_{16}$-Policy, B-Policy, and $\mathtt{C}_{0.1}$-Policy) on the BLW-TrafficDS trace. The highlighted values represent the lowest MAEs for a specific model in its respective column, while the underlined values represent the overall lowest MAEs for their respective columns. The improvement of a model is measured as a comparison between the best performing training policy and the baseline policy. The intervals of interest, denoted as $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_8$, correspond to a partition of the time series range $\mathcal{X}$ into eight contiguous sub-intervals of equal length.

Adaptive Interval Exploration with Patching Techniques. Figure 5 analyzes patching strategies (introduced in Section 4) for learning the underlying dynamics when intentionally increasing the number of intervals beyond the true quantity over the SynthDS trace. For $\mathtt{D}^\star_8$-Policy, which merges eight intervals into four, the ∞-strategy (selecting outputs with highest classifier confidence) outperforms the 1-strategy (confidence-weighted averaging). This advantage stems from classifier noise occasionally assigning non-negligible weights to low-confidence predictions, distorting reconstructions. Conversely, merging four intervals into two—where hypotheses inherently require averaging—shows superior performance for the 1-strategy.

Quantitative Analysis of Results. Following the qualitative evaluation of our proposed training policies, Table 1 and Table 2 present the numerical results obtained on the BLW-TrafficDS dataset and on the SynthDS dataset respectively. These tables represent the MAE values for our four models, each trained with the four different policies and evaluated on the intervals that the $\mathtt{D}_4$-Policy was trained on. For each interval, we showed the improvement percentage of the best policy compared to our baseline (B-Policy). The results prove our theoretical assumptions on both datasets by showcasing the quality of the $\mathtt{D}^\star_n$-Policy compared to our baseline. Moreover, by examining Table 2, we can clearly observe

Model	Training Strat.	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	Avg.
DLinear	$\mathtt{D}_4$-Policy	15.036	**11.678**	11.735	16.825	13.818
	$\mathtt{D}_8^1$-Policy	**14.538**	11.949	11.247	**16.782**	**13.629**
	$\mathtt{D}_8^\infty$-Policy	18.513	12.782	10.723	18.212	15.057
	B-Policy	50.942	20.285	17.074	40.994	32.324
	$\mathtt{C}_{0.1}$-Policy	24.805	12.846	**10.574**	20.340	17.141
	Improvement	**71.46%**	**42.43%**	**38.07%**	**59.06%**	**57.89%**
PatchTST	$\mathtt{D}_4$-Policy	**13.183**	**9.083**	**7.736**	**12.697**	**10.675**
	$\mathtt{D}_8^1$-Policy	13.210	10.180	8.606	12.851	11.212
	$\mathtt{D}_8^\infty$-Policy	15.146	11.240	8.919	13.290	12.149
	B-Policy	37.883	26.799	14.877	36.842	29.100
	$\mathtt{C}_{0.1}$-Policy	15.343	10.967	9.186	13.669	12.291
	Improvement	**65.20%**	**66.18%**	**48.00%**	**65.55%**	**63.32%**
TimeMixer	$\mathtt{D}_4$-Policy	16.207	**11.420**	11.489	**15.503**	**13.655**
	$\mathtt{D}_8^1$-Policy	**15.719**	12.236	11.747	17.294	14.249
	$\mathtt{D}_8^\infty$-Policy	17.895	15.048	13.291	19.693	16.482
	B-Policy	45.535	18.831	13.902	34.852	28.280
	$\mathtt{C}_{0.1}$-Policy	17.515	13.515	**10.783**	16.570	14.596
	Improvement	**65.48%**	**39.35%**	**22.44%**	**55.52%**	**51.72%**
iTransformer	$\mathtt{D}_4$-Policy	14.142	**7.960**	**7.968**	13.116	**10.796**
	$\mathtt{D}_8^1$-Policy	**14.096**	9.168	8.377	**12.427**	11.017
	$\mathtt{D}_8^\infty$-Policy	17.493	9.646	8.631	13.598	12.342
	B-Policy	40.213	26.184	16.239	33.737	29.093
	$\mathtt{C}_{0.1}$-Policy	19.999	10.587	8.707	13.378	13.168
	Improvement	**64.95%**	**69.60%**	**50.94%**	**63.17%**	**62.89%**

Table 2: MAE values ($\times 10^3$) for the four different models (DLinear, TimeMixer, iTransformer, and PatchTST) trained with the four policies ($\mathtt{D}_4$-Policy, $\mathtt{D}_8^1$-Policy, $\mathtt{D}_8^\infty$-Policy, B-Policy, and $\mathtt{C}_{0.1}$-Policy) on the SynthDS trace. The highlighted values represent the lowest MAEs for a specific model in its respective column, while the underlined values represent the overall lowest MAEs for their respective columns. The improvement of a model is measured as a comparison between the best performing training policy and the baseline policy. The intervals of interest, denoted as $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$, and $\mathcal{I}_4$, correspond to a partition of the time series range $\mathcal{X}$ into four contiguous sub-intervals of equal length.

that the reduction in accuracy of the discrete policies that is induced during patching is significant compared to the $\mathtt{C}_{0.1}$-Policy which has a very high variance in the MAE values. In general, these tables present three training policies that drastically improve the performance of our models, when evaluated at intervals determined at the inference time, compared to the baseline policy, which further motivates our foundational training policy in order for the model to adapt to specific intervals and thus diminish the averaging tendencies of the B-Policy.

6 Conclusion

This study presents a new training approach that modifies existing TSF models based on transformer architectures. These modified models are intended to function as foundational architectures, allowing them to be adapted to various downstream tasks through adjustments during inference. We conducted a detailed empirical evaluation, including comparisons with established baselines and ablation studies, to validate the effectiveness of the proposed method.

Potential avenue of future research directions include scaling these models to multiple domains and fine-tuning publicly available time series foundation models, such as Timer [23], Moirai [36], TimesFM [9], Chronos [1], Moment [12], and Toto [8], to incorporate the ability to adapt to arbitrary intervals, as introduced in this work. Additionally, a theoretical investigation into why the `C-Policy` policy underperforms compared to the $\texttt{D}_L^\star\texttt{-Policy}$ approach using frameworks such as PAC-learning [32], presents a promising avenue for future exploration and could reveal further improvements to our framework.

7 Acknowledgment

We thank Zhi-Quan Luo for his insightful and constructive feedback, which significantly contributed to the development of this study. We also appreciate the assistance of Dachao Lin and Xi Zheng for their valuable help in providing and curating the wireless dataset used in this research.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. 2024.

[2] Yoshua Bengio. Using a financial training criterion rather than a prediction criterion. *International journal of neural systems*, 8(04):433–443, 1997.

[3] Christoph Bergmeir, Frits de Nijs, Abishek Sriramulu, Mahdi Abolghasemi, Richard Bean, John Betts, Quang Bui, Nam Trong Dinh, Nils Einecke, Rasul Esmaeilbeigi, et al. Comparison and evaluation of methods for a predict+ optimize problem in renewable energy. *arXiv preprint arXiv:2212.10723*, 2022.

[4] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

[5] Robert G Brown. *Exponential smoothing for predicting demand*. Little, 1956.

[6] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: mining loop detector data. *Transportation research record*, 1748(1):96–102, 2001.

[7] Jiezhu Cheng, Kaizhu Huang, and Zibin Zheng. Fitting imbalanced uncertainties in multi-output time series forecasting. *ACM Transactions on Knowledge Discovery from Data*, 17(7):1–23, 2023.

[8] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability. *arXiv preprint arXiv:2407.07874*, 2024.

[9] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[10] Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

[11] Marzyeh Ghassemi, Marco Pimentel, Tristan Naumann, Thomas Brennan, David Clifton, Peter Szolovits, and Mengling Feng. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[12] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[13] Josif Grabocka, Randolf Scholz, and Lars Schmidt-Thieme. Learning surrogate losses. *arXiv preprint arXiv:1905.10108*, 2019.

[14] Clive William John Granger and Paul Newbold. *Forecasting economic time series*. Academic press, 2014.

[15] Yangdong He and Jiabao Zhao. Temporal convolutional networks for anomaly detection in time series. In *Journal of Physics: Conference Series*, volume 1213, page 042050. IOP Publishing, 2019.

[16] Ignacio Hounie, Javier Porras-Valenzuela, and Alejandro Ribeiro. Loss shaping constraints for long-term time series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, pages 19062–19084, 2024.

[17] Rob J Hyndman and George Athanasopoulos. 8.9 seasonal arima models. *Forecasting: principles and practice. oTexts. Retrieved*, 19, 2015.

[18] S Levine, C Finn, T Darrell, and P Abbeel. End-to-end training of deep visuomotor policies. arxiv. *URL: https://arxiv. org/abs/1504.00702*, 2016.

[19] Na Li, Donald M Arnold, Douglas G Down, Rebecca Barty, John Blake, Fei Chiang, Tom Courtney, Marianne Waito, Rick Trifunov, and Nancy M Heddle. From demand forecasting to inventory ordering decisions for red blood cells through integrating machine learning, statistical modeling, and inventory optimization. *Transfusion*, 62(1):87–99, 2022.

[20] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[21] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting, 2024.

[22] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35:9881–9893, 2022.

[23] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Transformers for time series analysis at scale. *arXiv e-prints*, pages arXiv–2402, 2024.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[25] Luis Martín, Luis F Zarzalejo, Jesus Polo, Ana Navarro, Ruth Marchante, and Marco Cony. Prediction of global solar irradiance based on time series analysis: Application to solar thermal power plants energy production planning. *Solar Energy*, 84(10):1772–1781, 2010.

[26] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[27] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023.

[28] Junwoo Park, Jungsoo Lee, Youngin Cho, Woncheol Shin, Dongmin Kim, Jaegul Choo, and Edward Choi. Deep imbalanced time-series forecasting via local discrepancy density. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 139–155. Springer, 2023.

[29] Meng Qi, Yuanyuan Shi, Yongzhi Qi, Chenxin Ma, Rong Yuan, Di Wu, and Zuo-Jun Shen. A practical end-to-end inventory management model with deep learning. *Management Science*, 69(2):759–773, 2023.

[30] Zheng Qian, Yan Pei, Hamidreza Zareipour, and Niya Chen. A review and discussion of decomposition-based hybrid models for wind energy forecasting applications. *Applied energy*, 235:939–953, 2019.

[31] Evan L Ray, Yijin Wang, Russell D Wolfinger, and Nicholas G Reich. Flusion: Integrating multiple data sources for accurate influenza predictions. *Epidemics*, 50:100810, 2025.

[32] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[33] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International conference on big data (Big Data)*, pages 3285–3292. IEEE, 2019.

[34] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *2011 International conference on computer vision*, pages 1457–1464. IEEE, 2011.

[35] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting, 2024.

[36] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. 2024.

[37] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[38] Haixu Wu, Hang Zhou, Mingsheng Long, and Jianmin Wang. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence*, 5(6):602–611, 2023.

[39] Wang Xue, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. Card: Channel aligned robust blend transformer for time series forecasting. *arXiv preprint arXiv:2305.12095*, 2023.

[40] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

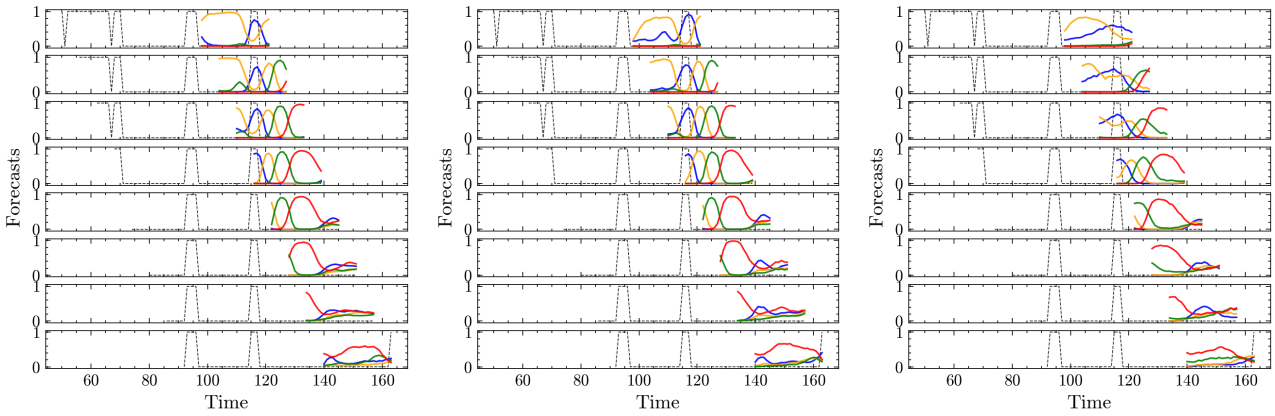
[41] Jun Zhang and Kim-Fung Man. Time series prediction using rnn in multi-dimension embedding phase space. In *SMC'98 conference proceedings. 1998 IEEE international conference on systems, man, and cybernetics (cat. no. 98CH36218)*, volume 2, pages 1868–1873. IEEE, 1998.

[42] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

[43] 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, pages 11106–11115, 2021.

[44] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[45] Zindi. Spatio-Temporal Beam-Level Traffic Forecasting Challenge. https://zindi.africa/competitions/spatio-temporal-beam-level-traffic-forecasting-challenge. [Accessed 04-03-2025].

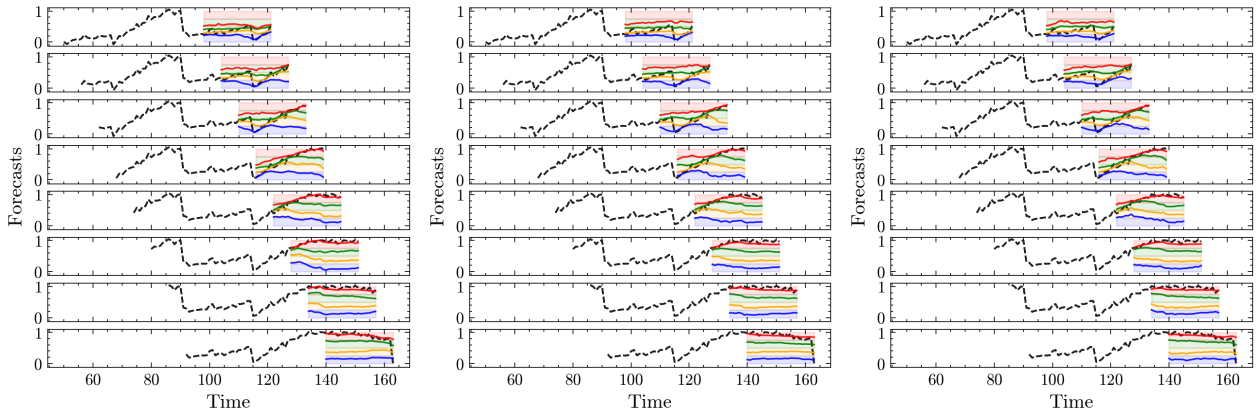


(a) iTransformer with the $\texttt{D}_4\texttt{-Policy}$ (b) PatchTST with the $\texttt{D}_4\texttt{-Policy}$ (c) DLinear with the $\texttt{D}_4\texttt{-Policy}$

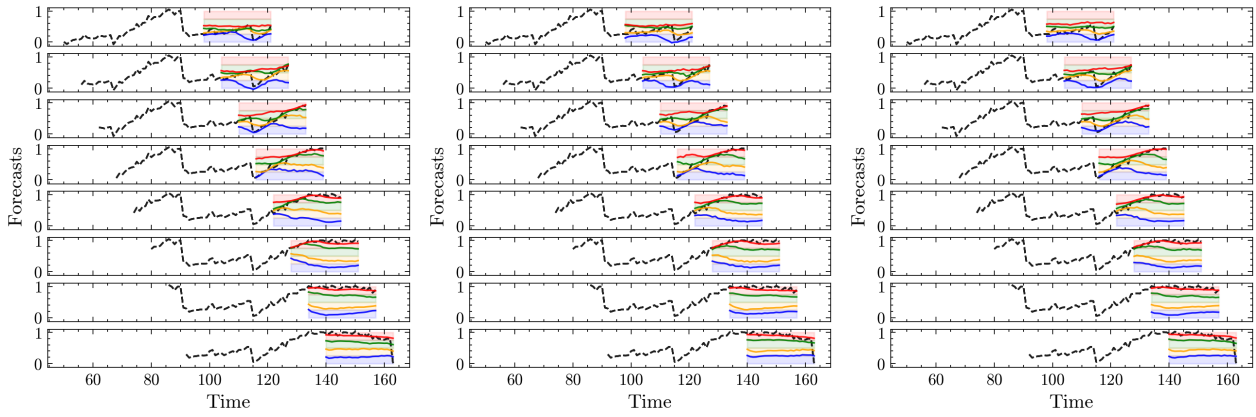
Figure 6: These figures represent the classification predictions for the $\texttt{D}_4\texttt{-Policy}$ for the best performing three models on the `SynthDS` dataset. The logic behind these plots and the color scheme are chosen according to the ones in Figure 7.

A Appendix

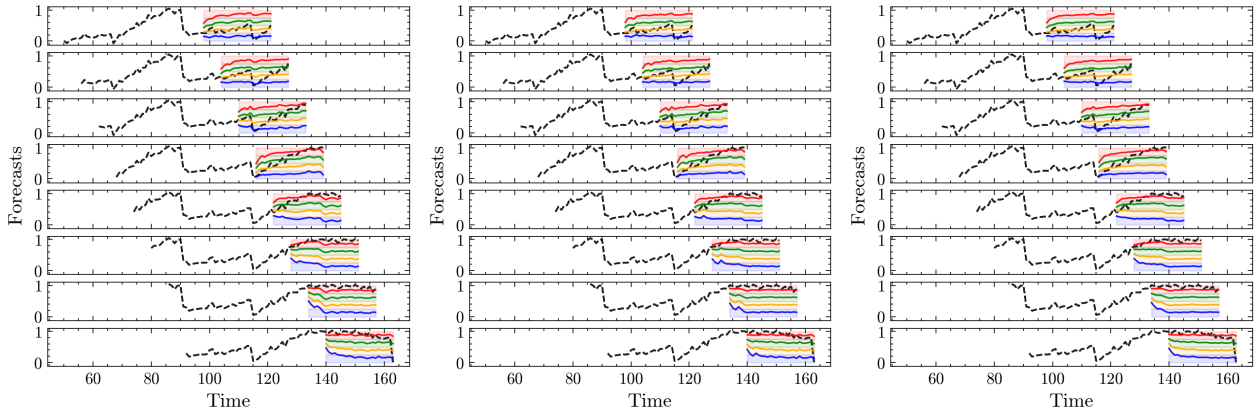
Additional Visualizations. In addition to the qualitative description of the training policies presented throughout this work, we present some additional visualizations that enrich both the context and the significance of our results. Figure 7 provides a comparison between the best performing models trained with the $\texttt{D}_4\texttt{-Policy}$, $\texttt{D}_8^{\infty}\texttt{-Policy}$ and $\texttt{D}_8^{1}\texttt{-Policy}$ on our `SynthDS` dataset. The figures represent sliding plots of the regression predictions on the four representative intervals described in Section 5.2. Subsequently, in Figure 6 we can visualize the sliding plots of the classification predictions of the previously chosen models trained with the $\texttt{D}_4\texttt{-Policy}$ on the `SynthDS` dataset, thus providing some intuition behind the patching methodologies. On the other hand, as a visual continuation to the results presented in Table 2.



(a) PatchTST with the $\texttt{D}_4\texttt{-Policy}$ (b) PatchTST with the $\texttt{D}_8^{\infty}\texttt{-Policy}$ (c) PatchTST with the $\texttt{D}_8^{1}\texttt{-Policy}$



(d) iTransformer with the $\texttt{D}_4\texttt{-Policy}$ (e) iTransformer with the $\texttt{D}_8^{\infty}\texttt{-Policy}$ (f) iTransformer with the $\texttt{D}_8^{1}\texttt{-Policy}$



(g) DLinear with the $\texttt{D}_4\texttt{-Policy}$ (h) DLinear with the $\texttt{D}_8^{\infty}\texttt{-Policy}$ (i) DLinear with the $\texttt{D}_8^{1}\texttt{-Policy}$

Figure 7: These figures represent the sliding plots of the three best performing models on the `SynthDS` dataset and have the purpose of adding qualitative context to the quantitative analysis presented in Table 2. They are formed of an input sequence of true value followed by the predicted output sequence for the four different intervals along with the actual true output sequence.