# Online Test-time Adaptation for Time Series Forecasting

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Multivariate time series forecasting, which predicts future dynamics by analyzing historical data, has become an essential tool in modern data analysis. With the development of deep models, batch-training based time series forecasting has made significant progress. However, in real-world applications, time series data is often collected incrementally in a streaming manner, with only a portion of the data available at each time step. As time progresses, distribution shifts in the data can occur, leading to a drastic decline in model performance. To address this challenge, online test-time adaptation and online time series forecasting have emerged as a promising solution. However, for the former, most online test-time adaptation methods are primarily designed for images and do not consider the specific characteristics of time series. As for the latter, online time series forecasting typically relies on updating the model with each newly collected sample individually, which may be problematic when the sample deviates significantly from the historical data distribution and contains noise, which may lead to a worse generalization performance. In this paper, we propose Batch Training with Transferable Online Augmentation (BTOA), which enhances model performance through three key ideas while enabling batch training. First, to fully leverage historical information, Transferable Historical Sample Selection (THSS) is proposed with theoretical guarantees to select historical samples that are most similar to the test-time distribution. Then, to mitigate the negative impact of distribution shifts through batch training and take advantage of the unique characteristics of time series, Transferable Online Augmentation (TOA) is proposed to augment the selected historical samples from the perspective of amplitude and phase in the frequency domain in a two-stream manner. Finally, a prediction module that utilizes a series decomposition module and a two-stream forecaster is employed to extract the complex patterns in time series, boosting the prediction performance. Moreover, BTOA is a general approach that is readily pluggable into any existing batch-training based deep models. Experiments demonstrate that our method achieves superior performance across seven benchmark datasets. Compared to state-of-the-art approaches, our method reduces the Mean Squared Error (MSE) by up to 13.7%. The code is available at https://anonymous.4open.science/r/BTOA/.

## 1 Introduction

Time series forecasting is crucial in real-world applications and is widely used across various fields, such as weather forecasting (Zhang et al., 2022a), power demand prediction (Gasparin et al., 2022), traffic flow analysis (Jin et al., 2021), and financial market modeling (Lai et al., 2018). In these practical applications, time series forecasting techniques not only help decision-makers better plan and optimize resources but also improve system efficiency and stability, driving the intelligent development of various industries. To improve forecasting accuracy, recent research has proposed advanced forecasting methods (Zhou et al., 2022; Wu et al., 2021; 2022). However, they typically rely on a conventional machine learning assumption that the training and test data follow the same distribution. This assumption often does not hold in real-world applications, where dataset shifts frequently occur (Quionero-Candela et al., 2009). Consequently, model performance can be significantly degraded when tested with data that deviates substantially from the training distribution. It is also worth noting that due to the inherent temporal nature, time series often arrive continuously in real-world scenarios, which means that models are typically required to handle streaming data. Recently, online test-time adaptation and online time series forecasting have emerged as promising solutions to address this issue, allowing pre-trained models to adapt to previously unseen data distributions during inference without the need for labeled data (Wang et al., 2023; Liang et al., 2023).

Unlike traditional batch training methods, online test-time adaptation and online time series forecasting adapt models in real-time using streaming data. Current online test-time adaptation methods can be broadly classified into three categories (Liang et al., 2023): (1) Data-based methods (Gong et al., 2024; Wang et al., 2022a), which focus on maximizing prediction consistency across different test datasets. (2) Model-based methods (Jang et al., 2022; Liu et al., 2023; Shu et al., 2022a), which aim to modify the original model architecture by adapting specific layers or mechanisms. (3) Optimization-based methods (Wang et al., 2022b; Shu et al., 2022b; Mummadi et al., 2021), which focus on optimizing prediction results using various optimization techniques. However, most existing online test-time adaptation methods are predominantly designed for image-based tasks, with few approaches specifically tailored for the complex patterns inherent in time series data. Current online time series forecasting methods typically utilize traditional Bayesian theory or add additional adapter modules to achieve adaptation (Pham et al., 2022; Zhang et al., 2023). These methods often rely on updating the model individually with each newly collected sample. When a sample deviates significantly from the historical data distribution and may contain substantial noise, these approaches can lead to reduced generalization performance.

In this paper, Batch training with Transferable Online Augmentation (BTOA) framework is proposed to address online test-time adaptation in time series in three aspects: Firstly, to fully leverage the information from the historical distribution, Transferable Historical Sample Selection (THSS) module is introduced to select the historical samples that are most similar to the test-time distribution from the memory bank. Secondly, to address distributional shift, we implement batch training through the Transferable Online Augmentation (TOA) module. Unlike traditional data augmentation methods in the time domain, such as Linear-Mixup (Zhang et al., 2017), Cut-Mixup (Yun et al., 2019), which tend to disrupt the frequency information of time series data (Verma et al., 2021; Demirel & Holz, 2024), TOA augments the selected samples in a two-stream manner from both the amplitude and phase perspectives in the frequency domain. This two-stream augmentation approach preserves the critical frequency information which is essential for accurate predictions. Finally, the prediction is produced through a prediction block, which consists of a series decomposition module and a two-stream forecaster. This design improves the model's prediction performance by extracting complex patterns in time series data.

Our main contributions are summarized as follows:

- Transferable Batch Training with Transferable Online Augmentation (BTOA) framework is proposed to address the distribution shift in online learning from three key perspectives. First, to fully leverage historical distribution information, Transferable Historical Sample Selection (THSS) module is proposed to select historical samples that have a smaller distribution discrepancy to the test-time distribution. Next, Transferable Online Augmentation (TOA) module is proposed to augment the selected samples in two-stream manner from the perspectives of frequency domain amplitude and phase, which preserves essential frequency domain information and enables batch training to alleviate distribution shift. Finally, a prediction block is employed to extract the complex temporal patterns and boost performance.
- BTOA is a general approach that is readily pluggable into any online time series forecasting model. This approach effectively mitigates the negative impact of noise in test-time samples, alleviates distribution shift, and enhances the effectiveness and robustness of the online learning model.
- We conduct experiments on seven popular real-world datasets, and our method achieves superior performance across all benchmark datasets. Compared to state-of-the-art approaches, our method reduces the Mean Squared Error (MSE) by up to 13.7%.

# 2 Related work

## 2.1 Online Test-time Adaptation

Online test-time adaptation (OTTA) continuously updates the model in real-time as it encounters new data during inference. This ensures swift adaptation to evolving data distributions without altering the original training procedure (Chen et al., 2022; Nguyen et al., 2023; Zhang et al., 2022b). Notably, TENT (Wang et al., 2020) addresses distributional shift by dynamically adjusting batch normalization parameters through entropy loss minimization during inference. Similarly, EATA (Niu et al., 2022) introduces a selective approach to optimizing unsupervised surrogate losses akin to TENT, focusing solely on reliable and informative data points. ViDA (Liu et al., 2023) employs supervision of the student output by leveraging predictions from the teacher with augmented input. Additionally, it introduces high/low-rank adapters that are updated to accommodate continual online test-time adaptation. ECL (Zeng et al., 2024) marks a departure from traditional methods by integrating a memory bank containing output distributions to establish thresholds for complementary labels. This innovative approach ensures the memory bank's continual relevance and effectiveness through periodic updates with the latest model parameters. Although these methods have shown promising results in the fields of computer vision and natural language processing (Wang et al., 2023; Liang et al., 2023), they do not take advantage of the unique characteristics of time series.

## 2.2 Online Time Series Forecasting

Online time forecasting focuses on streaming data, that is, for each $N$ variates sample $\mathbf{x}_i$ received, the model constructs a $L$-length look-back window $\mathbf{X}$ and outputs a $H$-length prediction window $\mathbf{Y}$, and then the true values are used to improve the model's performance in predicting the next sample. Online time series forecasting has a wide range of real-world applications due to the sequential nature of the data (Anava et al., 2013; Gultekin & Paisley, 2018; Aydore et al., 2019).

Previous methods have attempted to solve the online time series forecasting problem using Bayesian continuous learning theory, however, they are unable to quickly utilize information from historical samples. Inspired by Complementary Learning Systems (CLS) theory, FsNet (Pham et al., 2022) achieves great online time series forecasting by quickly adapting to historical data using the adapter module and slowly learning the newly collected sample with the Temporal Convolutional Network architecture. OneNet (Zhang et al., 2023) builds on FsNet by exploring the need for inter-channel dependencies, using the Online Convex Programming module to balance cross-time dependencies with cross-variate dependencies. This allows OneNet to achieve significant performance gains on some datasets with multiple variates such as the ECL dataset. Current models update based on a single received sample when processing streaming data. However, if a single sample is noisy, it can disrupt the optimal update path and significantly degrade model performance. To mitigate this issue, our BTOA implements batch training, which enhances the model's robustness against noisy data while maintaining effective online test-time adaptation.

# 3 Method

**Problem Formulation.** Given a well-trained time series forecasting model $f$ on the training set and a sequence of unlabeled time series segments. Online test-time adaptation aims to leverage the labeled knowledge embedded in prediction model $f$ to infer the future values of samples under distribution shift, in an online manner. In this problem, the learning process takes place over a sequence of rounds, where the model receives a $L$-length look-back window $\mathbf{X} = \{x_1, \ldots, x_L\} \in \mathbb{R}^{N \times L}$ and predicts the forecast window $\mathbf{Y} = \{y_1, \ldots, y_H\} \in \mathbb{R}^{N \times H}$. The true values are then revealed to improve the model's performance in the next rounds. Our goal is to continuously optimize the prediction model $f$, which can mitigate the negative impact of distribution shifts.

**Structure overview.** Figure 1 illustrates the comprehensive workflow of Batch Training with Transferable Online Augmentation (BTOA). BTOA is meticulously structured into three principal modules: a transferable historical sample selection module that fully leverages historical distribution information, a transferable

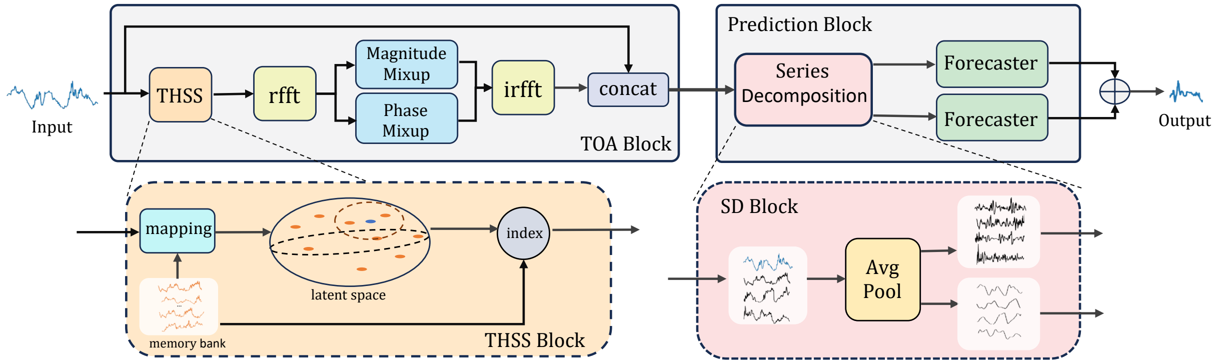


Figure 1: Overall architecture of Batch training with Transferable Online Augmentation (BTOA). The THSS block is used to select historical samples, the TOA block implements batch training through data augmentation, and the Prediction block generates the final output.

online augmentation module that enables batch training to alleviate the negative effects of distribution shift, and a prediction block that extracts complex temporal patterns and produces predictions.

### 3.1 Transferable Historical Sample Selection

The distribution of the data stream changes dynamically over time, which can adversely affect time series forecasting accuracy. To mitigate this issue, Transferable Historical Sample Selection is proposed to effectively utilize historical data. Firstly, we aim to select historical samples that have smaller distribution discrepancy to the test-time distribution.

Specifically, we establish a memory bank, which is a set and denoted as $\mathcal{M}$. $\mathcal{M}$ stores historical samples and is updated using a First-In-First-Out (FIFO) policy to maintain a fixed size. Upon receiving test-time sample, we use the THSS module to select historical samples that are semantically similar to the test-time sample and most closely align with test-time distribution. This selection is achieved through a mapping model, where intuitively, any model that can preserve semantic consistency between the original and mapped data can be used. Due to the inherent ability of the Variational Autoencoder (VAE) (Higgins et al., 2017) to maintain semantic consistency between input and output, we choose VAE model as our mapping model. Initially, the VAE model is pre-trained in an unsupervised manner on the training set, and during online test-time adaptation, its parameters are frozen to remain unchanged. Once the test-time sample $\mathbf{X}_{test}$ is introduced, both the test-time sample and the historical samples stored in $\mathcal{M}$ are projected into a latent space. We then calculate the distances between these samples in the latent space and select those historical samples that have smaller distribution discrepancy to the test-time distribution. These selected samples are semantically similar to the test-time sample, forming the selected historical sample set $\mathcal{X}_h$. The above procedure can be formulated as follows:

$$\begin{aligned}
&\mathbf{z}_{test} = E(\mathbf{X}_{test}), \mathbf{z}_i = E(\mathbf{m}_i), \forall \mathbf{m}_i \in \mathcal{M} \\
&d_i = \text{cosine\_similarity}(\mathbf{z}_{test}, \mathbf{z}_i) = \frac{\mathbf{z}_{test} \cdot \mathbf{z}_i}{\|\mathbf{z}_{test}\| \|\mathbf{z}_i\|} \\
&\mathcal{X}_h = \{\mathbf{m}_i\}_{i \in S_n} \text{ where } S_n = \arg\text{sort}_i(d_i)[:n],
\end{aligned} \tag{1}$$

where $E(\cdot)$ represents the encoder of the VAE model, and $n$ is a hyperparameter indicating the number of historical samples we need to utilize. By effectively utilizing historical samples that have a smaller distribution discrepancy to the test-time distribution, we alleviate the distribution shift and boost the prediction performance.

### 3.2 Transferable Online Augmentation

We mitigate the negative impact of distribution shift during online test-time adaptation by introducing batch training, which can be achieved through data augmentation techniques. However, existing data augmentation methods, such as Linear-Mixup (Zhang et al., 2017) and Cut-Mixup (Yun et al., 2019), primarily mix time series in the time domain, which can affect the frequency domain information that is crucial for accurate prediction (Demirel & Holz, 2024; Ullrich et al., 2020; Zhang et al., 2022c). Since distribution shift is more pronounced in online test-time adaptation, preserving the frequency domain information of time series becomes particularly important.

To preserve the frequency domain information, we propose a two-stream augmentation approach that focuses on both the amplitude and phase in the frequency domain, and we select the aforementioned set of selected historical samples $\mathcal{X}_h$, which are closer to the test-time distribution, as the source for augmentation. By doing so, we ensure that the augmented instances' phase and amplitude are properly interpolated based on the test-time sample, avoiding destructive interference in the frequency domain. We first apply the Fast Fourier Transform (FFT) to both the test-time sample and historical samples stored in the set $\mathcal{X}_h$, decomposing them into amplitude and phase components, which can be formulated as:

$$\mathrm{A}(\mathbf{X}_i)e^{j\mathrm{P}(\mathbf{X}_i)} = \mathcal{F}(\mathbf{X}_i), \quad \mathbf{X}_i \in \{\mathbf{X}_{test}\} \cup \mathcal{X}_h, \tag{2}$$

where $\mathcal{F}$ denotes the Fast Fourier Transform, $\mathrm{A}(\cdot), \mathrm{P}(\cdot)$ means the amplitude and phase. Then, we combine the amplitude and phase of the test-time sample with those of the historical samples. This process ensures that the frequency domain information remains intact while enhancing the data with relevant historical patterns. To ensure more appropriate historical samples, we perform aggressive data augmentation primarily using historical samples when their distance from the test-time sample in the latent space is small, indicating similar distributions. Conversely, when the distance between the historical and test-time sample is large, implying a significant distributional shift, we prioritize the test-time sample for data augmentation. Specifically, the process of mixup is:

$$\mathrm{A}(\mathcal{X}_{aug}) = \{\mathbf{X}_j | \lambda_A \mathrm{A}(\mathbf{X}_{test}) + (1-\lambda_A)\mathrm{A}(\mathbf{X}_j), \mathbf{X}_j \in \mathcal{X}_h\} \tag{3}$$
$$\mathrm{P}(\mathcal{X}_{aug}) = \{\mathbf{X}_j | \lambda_P \mathrm{P}(\mathbf{X}_{test}) + (1-\lambda_P)\mathrm{P}(\mathbf{X}_j), \mathbf{X}_j \in \mathcal{X}_h\}, \tag{4}$$

where $\mathcal{X}_{aug}$ means the augmented sample set. $\lambda_A, \lambda_P$ are hyperparameters representing the mixing coefficients for amplitude and phase, respectively. When the distance between latent vectors is below a distance threshold, we sample the mixing coefficients for amplitude and phase from a uniform distribution, denoted as $\lambda_A, \lambda_P \sim \mathcal{U}(\beta, 1.0)$, prioritizing data augmentation on the historical samples, with $\beta$ being a lower value. Conversely, if the distance exceeds the threshold, we focus on augmenting the test-time sample. In this case, the coefficients are drawn from a truncated normal distribution, $\lambda_A, \lambda_P \sim \mathcal{N}(\mu, \theta)$, characterized by a high mean and low standard deviation. The process for determining the sampling distribution of $\lambda_A$ and $\lambda_P$ is as follows:

$$\lambda_A, \lambda_P \in \begin{cases} \mathcal{U}(\beta, 1.0), & \text{if } d_i \leq \tau \\ \mathcal{N}(\mu, \theta), & \text{if } d_i > \tau, \end{cases} \tag{5}$$

where $d_i$ represents the distance between latent vectors, and $\tau$ denotes a predefined distance threshold. Finally, the augmented sample set $\mathcal{X}_{aug}$ is obtained by applying the inverse FFT to the mixed components. For ease of understanding, we use $\mathbf{X}_{aug}$ to represent an element of $\mathcal{X}_{aug}$ hereafter. As shown below:

$$\mathbf{X}_{aug} = \mathcal{F}^{-1}\left(A(\mathbf{X}_{aug})e^{jP(\mathbf{X}_{aug})}\right), \tag{6}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fast Fourier Transform. After obtaining the augmented set $\mathbf{X}_{aug}$, we concatenate these augmented samples with the test-time sample and input them as a batch into the next module for training. Compared to the previous approach, which only used the test-time sample to update the model, this method significantly reduces the negative impact of noise in the test-time sample on model optimization.

### 3.3 Prediction Model and Training Objective

**Prediction Model.** To effectively learn complex temporal patterns in time series forecasting, we use series decomposition (RB, 1990; Anderson, 1976). This technique simplifies complex raw data, allowing the model to make better predictions. Specifically, we extract the trend component of the time series by applying a moving average kernel to the input series. The difference between the trend component and the original series is regarded as the seasonal component. These components reflect the long-term trend and cyclical relationship of the time series, respectively. The series decomposition is handled as follows:

$$\mathbf{X}_t = \text{AvgPool}(\text{padding}(\mathbf{X}_{aug})) \tag{7}$$

$$\mathbf{X}_s = \mathbf{X}_{aug} - \mathbf{X}_t, \tag{8}$$

where $\mathbf{X}_t, \mathbf{X}_s$ denote the extracted long-term trend and seasonal terms, respectively. We use padding to maintain the original series length, and then apply the AvgPool layer for moving average calculations.

After decomposition, the trend component $\mathbf{X}_t$ and the seasonal component $\mathbf{X}_s$ will be fed into two-stream forecaster with identical structures. The outputs from two-stream forecaster are combined to generate the final prediction $\mathbf{Y}$. As shown below:

$$\mathbf{Y} = \text{Forecaster}_s(\mathbf{X}_s) + \text{Forecaster}_t(\mathbf{X}_t). \tag{9}$$

**Training Objective.** We use the $L2$ loss to optimize the parameters of the BTOA model, with the loss function defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{j=1}^{N}\left\|\hat{\mathbf{Y}}_{1:H}^{j} - \mathbf{Y}_{1:H}^{j}\right\|, \tag{10}$$

where $N$ represents the number of channels in the time series. During the test-time adaptation process, we use the mean MSE and MAE between the ground truth $\hat{\mathbf{Y}}$ and the model's predicted output $\mathbf{Y}$ across all samples as the final evaluation metrics to compare model performance.

It is worth noting that BTOA is a general module designed to mitigate the distributional shift that occur during the learning process. This adaptability makes it applicable to any online time series forecasting model. The inherent flexibility of BTOA allows it to be integrated seamlessly with a variety of models, enhancing their robustness against changes in data distribution. Moreover, BTOA is not limited to a specific algorithm or framework. This means that as more advanced deep models are developed, BTOA can be incorporated into these advanced deep models to further improve performance.

### 3.4 Theoretical Insights

In the **Transferable Historical Sample Selection** module, we need to select a mapping model that can reflect the semantic consistency of samples before and after mapping. Proposition 3.1 theoretically demonstrates the rationale for using the VAE model as a mapping model.

**Proposition 3.1** *(**Consistency in Latent Space** (Li et al., 2022)) Given a well-trained unconditional VAE with the encoder $E(\cdot)$ that produces distribution $p_E(z|\mathbf{x})$, the decoder $D(\cdot)$ that produces distribution $q_D(\mathbf{x}|z)$ while the prior for $z$ is $p(z)$, let $\mathbf{z_1}$ and $\mathbf{z_2}$ be two latent vectors of two different real samples $\mathbf{x_1}$ and $\mathbf{x_2}$, i.e., $E(\mathbf{x_1}) = \mathbf{z_1}$ and $E(\mathbf{x_2}) = \mathbf{z_2}$. If the distance $d(\mathbf{z_1}, \mathbf{z_2}) \leq \delta$, then $D(\mathbf{z_1})$ and $D(\mathbf{z_2})$ will have a similar semantic label as in Equation equation 11.*

$$|I(D(\mathbf{z_1}); \mathbf{y}) - I(D(\mathbf{z_2}); \mathbf{y})| \leq \epsilon, \tag{11}$$

*where $\epsilon$ stands for tolerable semantic difference, $\delta$ is the maximum distance to maintain semantic consistency, and $d(\cdot)$ is a distance measure such as cosine similarity between two vectors.*

Let the historical sample set $\mathcal{P}$ be the set that includes the training set and all samples received prior to the test-time sample. The test-time sample set $\mathcal{Q}$ refers to the samples being received at present. Due to distribution shift, the data distributions of $\mathcal{P}$ and $\mathcal{Q}$ may differ. In the **Transferable online augmentation** module, the augmented set $\mathcal{X}_{aug}$ derived from the selected historical sample set $\mathcal{X}_h$, has a data distribution that is closer to the historical sample set $\mathcal{P}$ compared to using the test-time sample alone. Proposition 3.2 provides theoretical support for the performance advantages of using $\mathcal{X}_{aug}$ as input, suggesting that it can lead to a smaller upper bound on the generalization error. Moreover, our choice of $L2$ loss as the loss function aligns with the requirements of the proposition.

**Proposition 3.2** *(**Generalization Error Upper Bound** (Mansour et al., 2009)) Let $f_Q^* \in \arg\min_{f\in F} \mathcal{L}_\mathcal{Q}(f, G_Q)$ and similarly let $f_P^*$ be a minimizer of $\mathcal{L}_\mathcal{P}(f, G_P)$. Note that these minimizers may not be unique. For adaptation to succeed, it is natural to assume that the average loss $\mathcal{L}_\mathcal{Q}(f_Q^*, f_P^*)$ between the best-in-class hypotheses is small. Under that assumption and for a small discrepancy distance, there is a useful bound on the error of a hypothesis with respect to the test-time sample set as in Equation  equation 12.*

$$\begin{aligned}\mathcal{L}_\mathcal{Q}(f, G_Q) \leq \mathcal{L}_\mathcal{Q}(f_Q^*, G_Q) + \mathcal{L}_\mathcal{P}(f, f_P^*) + disc_L(\mathcal{Q}, \mathcal{P})\\ + \min\{\mathcal{L}_\mathcal{P}(f_P^*, f_Q^*), \mathcal{L}_\mathcal{Q}(f_P^*, f_Q^*)\},\end{aligned} \tag{12}$$

*where $G$ represents the ideal prediction model and the loss function $\mathcal{L}$ is symmetric and obeys the triangle inequality.*

## 4 Experiments

In this section, we evaluated BTOA across a range of online time series forecasting applications, demonstrating its effectiveness in diverse scenarios. In addition to the primary evaluation, we conducted comprehensive ablation studies to investigate the contribution of each individual component of BTOA.

Table 1: Statistics of popular datasets for benchmark.

| Datasets | ETTh1 | ETTh2 | ETTm1 | ETTm2 | WTH | Electricity | Traffic |
|---|---|---|---|---|---|---|---|
| Features | 7 | 7 | 7 | 7 | 11 | 321 | 862 |
| Timesteps | 17420 | 17420 | 69680 | 69680 | 35065 | 26304 | 17544 |
| ADF | -5.90 | -4.13 | -14.98 | -5.66 | -12.52 | -8.44 | -15.02 |

**Dataset.** We evaluate the performance of BTOA on seven real-world datasets, including ETT (with 4 subsets), ECL, Traffic used in iTransformer (Liu et al., 2024), and WTH used in FsNet (Pham et al., 2022). The presence and severity of distributional shifts in these datasets can be measured using the Augmented Dickey-Fuller (ADF) test statistic (Liu et al., 2022). Basic information about these datasets is provided in Table 1, where it can be observed that they exhibit varying degrees of distributional shift. Detailed dataset descriptions are available in appendix B.1.

**Baseline.** We evaluate multiple baselines in our experiments, incorporating methods from continual learning, time series forecasting, and online learning. (1) The Experience Replay (**ER**) (Chaudhry et al., 2019) and its three recent advanced variants. **ER** deploy a buffer that store previous data and interleave with newer samples during learning. The first variant is **TFCL** (Aljundi et al., 2019b), which introduces a task-boundary detection mechanism and a knowledge consolidation strategy. The second variant is **MIR** (Aljundi et al., 2019a), which selects samples that cause the most forgetting. The final variant is **DER++** (Buzzega et al., 2020), which Adds a knowledge distillation module compared with the standard ER. (2) **Stationary** (Liu et al., 2022) focuses on modeling non-stationary time series through the De-stationary Attention module. (3) **Revin** (Kim et al., 2022) dynamically normalizes the time series to mitigate the negative impact of non-stationarity. (4) **PatchTST** (Nie et al., 2022) is a traditional time series forecasting method that models time series through channel independence and patch. (5)**iTransformer** (Liu et al., 2024) is a traditional time series forecasting method that models time series by transforming the role of the attention mechanism and feed-forward networks. (6) **FsNet** (Pham et al., 2022) avoids catastrophic forgetting of historical samples through the use of TCN structures and Adapter modules, and can be quickly adapted for new samples. (7) **OneNet** (Zhang et al., 2023), which is the previous state-of-the-art online time series forecasting method.

**Implementation details.** We follow the experimental setup of FsNet (Pham et al., 2022), setting the look-back window length to 60 across all benchmarks, with the prediction horizon varying from $H = 1$, 24, to 48. Since learning is conducted in sequential rounds, the model receives a look-back window in each round and predicts the forecast window. All models are evaluated based on their cumulative Mean Squared Error (MSE) and Mean Absolute Error (MAE), which assess the models' performance over the entire learning process. We use the AdamW optimizer (Loshchilov & Hutter, 2017) to minimize the $L2$ loss. The data is split into two phases: warm-up and online training, with a 25:75 ratio. To simulate streaming data, we set the batch size to 1, reflecting the arrival of data in a streaming fashion. Our method is a general-purpose module, and here we instantiate it using OneNet as the forecaster. More details about the implementation, architectures, and hyperparameters with the trained VAE model are given in Appendix B.5.

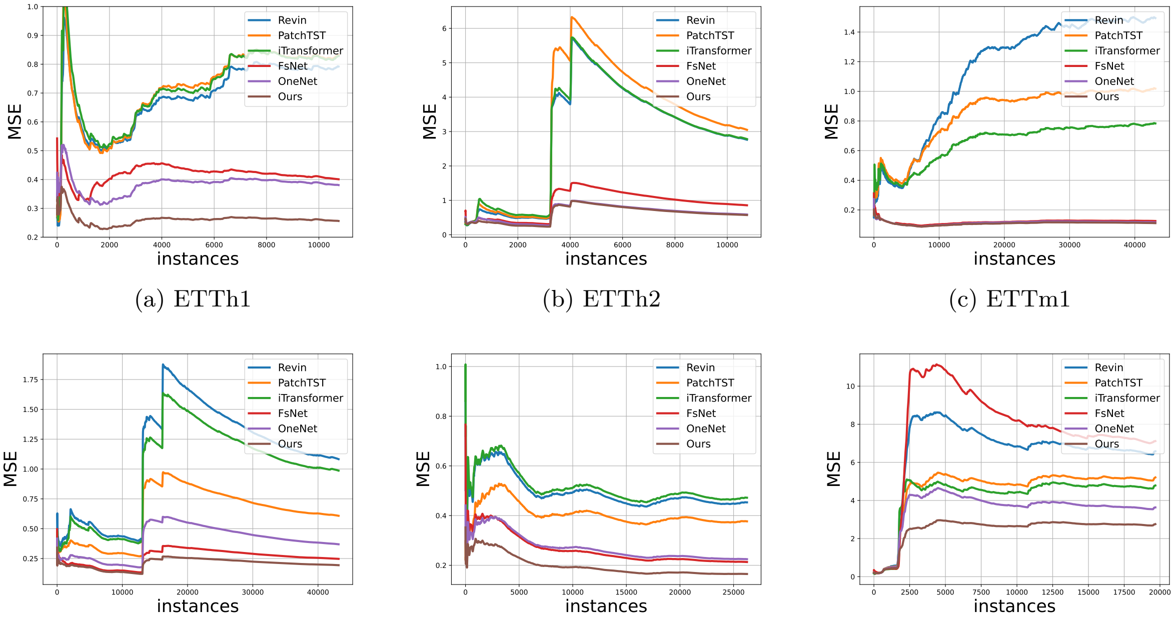
Table 2: Full results of the online time-series forecasting task. We compare extensive competitive models under different prediction lengths following the setting of FsNet. The best results are in **bold**, and the second best are underlined.

| Models | | **BTOA** | | OneNet | | FsNet | | iTransformer | | PatchTST | | Revin | | Stationary | | DER++ | | MIR | | TFCL | | ER | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | 1 | **0.223** | 0.301 | 0.235 | 0.303 | 0.286 | 0.343 | **0.223** | **0.294** | 0.246 | 0.311 | 0.238 | 0.304 | 0.383 | 0.395 | 0.239 | 0.305 | 0.239 | 0.304 | 0.241 | 0.310 | 0.240 | 0.316 |
| | 24 | **0.271** | **0.325** | 0.400 | 0.442 | 0.411 | 0.436 | 0.703 | 0.524 | 0.810 | 0.570 | 0.672 | 0.510 | 0.759 | 0.565 | 0.648 | 0.534 | 0.662 | 0.539 | 0.670 | 0.542 | 0.673 | 0.547 |
| | 48 | **0.265** | **0.356** | 0.447 | 0.454 | 0.402 | 0.452 | 0.828 | 0.570 | 0.831 | 0.575 | 0.792 | 0.557 | 0.747 | 0.570 | 0.606 | 0.525 | 0.677 | 0.524 | 0.679 | 0.540 | 0.634 | 0.538 |
| ETTh2 | 1 | 0.390 | 0.362 | 0.383 | 0.351 | 0.467 | 0.371 | 0.418 | 0.352 | **0.362** | 0.351 | 0.383 | **0.344** | 0.770 | 0.383 | 0.508 | 0.375 | 0.486 | 0.410 | 0.557 | 0.472 | 0.508 | 0.376 |
| | 24 | **0.505** | **0.397** | 0.538 | 0.414 | 0.693 | 0.473 | 1.716 | 0.587 | 1.622 | 0.577 | 1.741 | 0.581 | 2.090 | 0.659 | 0.828 | 0.540 | 0.812 | 0.541 | 0.846 | 0.548 | 0.808 | 0.543 |
| | 48 | **0.587** | **0.436** | 0.604 | 0.445 | 0.867 | 0.516 | 2.781 | 0.676 | 2.716 | 0.672 | 2.762 | 0.664 | 2.938 | 0.722 | 1.157 | 0.577 | 1.103 | 0.565 | 1.208 | 0.592 | 1.136 | 0.571 |
| ETTm1 | 1 | 0.106 | 0.187 | 0.117 | 0.202 | **0.104** | 0.187 | 0.106 | 0.192 | 0.116 | **0.186** | 0.122 | 0.208 | 0.111 | 0.197 | 0.110 | 0.192 | 0.112 | 0.197 | 0.109 | 0.195 | 0.114 | 0.197 |
| | 24 | **0.114** | **0.222** | 0.134 | 0.243 | 0.137 | 0.249 | 0.777 | 0.529 | 0.427 | 0.471 | 1.531 | 0.704 | 0.536 | 0.449 | 0.196 | 0.326 | 0.192 | 0.325 | 0.211 | 0.329 | 0.202 | 0.333 |
| | 48 | **0.118** | **0.227** | **0.118** | 0.228 | 0.124 | 0.240 | 0.783 | 0.545 | 0.553 | 0.549 | 1.018 | 0.614 | 1.433 | 0.721 | 0.208 | 0.340 | 0.210 | 0.342 | 0.236 | 0.350 | 0.220 | 0.351 |
| ETTm2 | 1 | 0.174 | 0.226 | 0.191 | 0.233 | 0.179 | 0.229 | **0.168** | **0.221** | 0.184 | 0.228 | 0.173 | 0.226 | 0.194 | 0.228 | 0.190 | 0.231 | 0.192 | 0.230 | 0.191 | 0.235 | 0.191 | 0.233 |
| | 24 | **0.206** | **0.259** | 0.267 | 0.261 | 0.233 | 0.276 | 0.639 | 0.430 | 0.547 | 0.412 | 0.652 | 0.435 | 0.954 | 0.579 | 0.307 | 0.345 | 0.309 | 0.341 | 0.311 | 0.346 | 0.310 | 0.347 |
| | 48 | **0.204** | **0.267** | 0.273 | 0.284 | 0.299 | 0.313 | 0.987 | 0.502 | 0.608 | 0.427 | 1.083 | 0.502 | 1.209 | 0.592 | 0.329 | 0.359 | 0.330 | 0.360 | 0.339 | 0.364 | 0.331 | 0.363 |
| WTH | 1 | **0.156** | **0.197** | 0.158 | 0.201 | 0.161 | 0.215 | 0.160 | 0.205 | 0.162 | 0.200 | 0.165 | 0.211 | 0.152 | 0.196 | 0.208 | 0.235 | 0.179 | 0.244 | 0.177 | 0.240 | 0.180 | 0.244 |
| | 24 | **0.161** | **0.241** | 0.189 | 0.273 | 0.189 | 0.276 | 0.375 | 0.399 | 0.372 | 0.393 | 0.370 | 0.394 | 0.428 | 0.446 | 0.270 | 0.351 | 0.291 | 0.355 | 0.301 | 0.363 | 0.293 | 0.356 |
| | 48 | **0.173** | **0.255** | 0.197 | 0.278 | 0.223 | 0.303 | 0.472 | 0.467 | 0.465 | 0.459 | 0.453 | 0.452 | 0.487 | 0.484 | 0.294 | 0.359 | 0.297 | 0.361 | 0.323 | 0.382 | 0.297 | 0.363 |
| ECL | 1 | 2.430 | 0.266 | 2.590 | 0.258 | 3.317 | 0.542 | **1.897** | **0.218** | 2.022 | 0.341 | 3.873 | 0.331 | 2.613 | 0.508 | 2.657 | 0.421 | 2.575 | 0.504 | 2.732 | 0.273 | 2.579 | 0.506 |
| | 24 | **2.493** | **0.346** | 2.700 | 0.366 | 6.071 | 1.024 | 4.009 | 0.313 | 4.325 | 0.375 | 4.862 | 0.332 | 3.469 | 0.579 | 8.996 | 1.035 | 9.265 | 1.066 | 12.094 | 0.383 | 9.327 | 1.057 |
| | 48 | **2.423** | **0.462** | 3.261 | 0.400 | 7.234 | 1.089 | 4.787 | 0.346 | 5.030 | 0.399 | 6.583 | 0.379 | 4.987 | 0.789 | 9.009 | 1.048 | 9.411 | 1.079 | 12.110 | 0.410 | 9.685 | 1.074 |
| Traffic | 1 | **0.232** | **0.205** | 0.233 | 0.215 | 0.295 | 0.253 | 0.298 | 0.321 | 0.322 | 0.307 | 0.257 | 0.295 | 0.418 | 0.325 | 0.280 | 0.241 | 0.290 | 0.251 | 0.323 | 0.524 | 0.286 | 0.247 |
| | 24 | **0.310** | **0.261** | 0.348 | 0.269 | 0.360 | 0.287 | 1.097 | 0.519 | 0.913 | 0.508 | 1.097 | 0.502 | 1.275 | 0.575 | 0.384 | 0.289 | 0.390 | 0.302 | 0.553 | 1.256 | 0.383 | 0.299 |
| | 48 | **0.351** | **0.293** | 0.384 | 0.302 | 0.378 | 0.297 | 1.615 | 0.568 | 1.519 | 0.571 | 1.678 | 0.573 | 1.765 | 0.617 | 0.398 | 0.295 | 0.391 | 0.310 | 0.564 | 1.303 | 0.394 | 0.307 |
| AVG | | **0.567** | **0.285** | 0.656 | 0.306 | 1.068 | 0.395 | 1.183 | 0.418 | 1.160 | 0.423 | 1.452 | 0.434 | 1.320 | 0.504 | 1.325 | 0.425 | 1.353 | 0.436 | 1.656 | 0.474 | 1.371 | 0.437 |

## 4.1 Online Forecasting Results

**Cumulative performance.** Table 2 fully demonstrates the advantages of BTOA in terms of both MSE and MAE metrics. Compared to the previous state-of-the-art model, OneNet, BTOA achieves superior performance. Notably, BTOA exhibits outstanding results on multivariate datasets such as ECL and Traffic, significantly improving forecasting performance. This highlights BTOA's critical role in addressing distribution shift issues that arise during online test-time adaptation. By rationally leveraging historical samples, BTOA effectively mitigates distribution shift, reduces the impact of potential noise in test-time samples on the model, and markedly enhances model robustness. Furthermore, compared to specialized methods for non-stationary time series data like Revin and Stationary, BTOA still demonstrates superior performance.

This may stem from the fact that Revin and Stationary are fundamentally designed for batch-training scenarios, leading to suboptimal performance under online learning experimental settings where models are updated with individual samples.



(a) ETTh1 (b) ETTh2 (c) ETTm1

(d) ETTm2 (e) WTH (f) ECL

Figure 2: Evolution of the cumulative MSE loss during training with forecasting window $H = 48$. The horizontal axis represents instances.

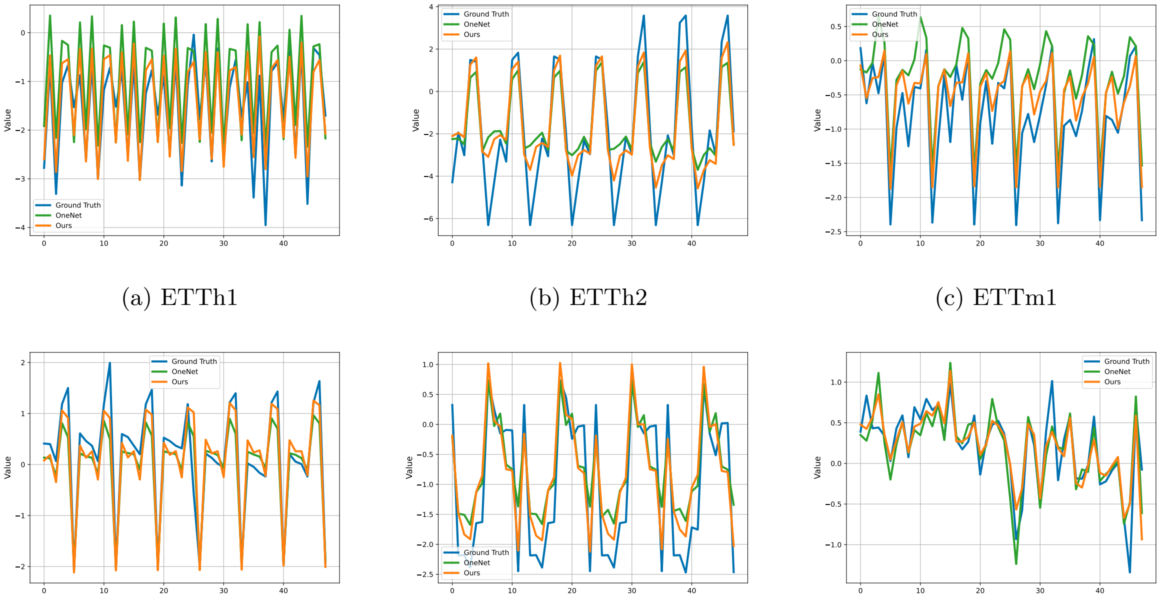
**Convergence of different deep models.** Figure 2 illustrates the convergence behavior of BTOA compared to baselines across six datasets. It is evident that the peaks in the loss curve tend to occur during the early and middle stages of learning, indicating that changes in data distribution significantly impair model performance. Such shifts in distribution are particularly challenging for time series models, as they disrupt the model's learned patterns, causing a temporary performance degradation. Traditional batch learning settings typically evaluate performance only on a small portion of data at the end, often neglecting the distributional shifts that occur during training, which leads to suboptimal adaptation to real-world scenarios. From Figure 2, we observe that when a distribution shift occurs, the increase in MSE for BTOA is noticeably smaller than that of all baselines. This demonstrates BTOA's superior ability to detect and rapidly adapt to distributional changes, thereby maintaining more stable performance. The key to this adaptability lies in the integration of our data augmentation module, which enhances the model's robustness by leveraging historical data and two-stream augmentation to account for shifts in both the amplitude and phase of the data. This capability allows BTOA to make effective adjustments without compromising critical frequency domain information, setting it apart from other models that struggle with such changes.In summary, these results underscore the effectiveness of BTOA in quickly adapting to distributional shifts, ensuring more stable and reliable performance. The ability to mitigate the impact of these shifts highlights the potential of BTOA as a versatile and robust solution for real-world time series forecasting challenges.

**The batch size at the online test-time adaptation.** In Table 3, we also investigated the impact of the batch size hyperparameter, which determines the number of historical samples to be augmented during batch training. Our experiments revealed that optimal performance is achieved with batch sizes of either 16 or 32. Considering both performance and computational efficiency, we selected a batch size of 16 as the optimal choice. This decision strikes a balance between ensuring high model performance and reducing training time, thus improving the model's practicality for real-world applications, especially when handling

Table 3: Performance metrics for different batch sizes.

| Dataset | ETTh1 | | | ETTm2 | | | WTH | | | ECL | | | Traffic | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| batch-size | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 |
| 8 | 0.278 | 0.321 | 0.323 | 0.119 | 0.130 | 0.119 | 0.159 | 0.170 | 0.192 | 2.678 | 3.386 | 4.313 | 0.241 | 0.379 | 0.389 |
| 16 | **0.231** | **0.276** | 0.269 | **0.107** | **0.124** | **0.116** | **0.156** | **0.160** | **0.173** | 2.430 | **2.493** | **2.423** | **0.232** | **0.310** | **0.351** |
| 32 | 0.235 | 0.281 | **0.268** | 0.118 | 0.127 | **0.116** | 0.157 | 0.168 | 0.180 | **2.001** | 2.511 | 2.343 | 0.241 | 0.470 | 0.511 |
| 64 | 0.281 | 0.331 | 0.336 | 0.119 | 0.128 | 0.118 | 0.163 | 0.192 | 0.215 | 2.509 | 2.981 | 3.324 | 0.237 | 0.336 | 0.439 |

large-scale streaming data. By optimizing this hyperparameter, we can significantly accelerate the training process without sacrificing accuracy, making our approach better suited for time-sensitive tasks.



(a) ETTh1 (b) ETTh2 (c) ETTm1

(d) ETTm2 (e) WTH (f) ECL

Figure 3: forecast results when distribution shift occur, with forecasting window $H = 48$.

**Visualization.** Figure 3 presents sample instances from six datasets during the phase where the loss curves increase, representing the comparison between BTOA, OneNet, and the ground truth under distributional shifts. It is clear that at the onset of the shift, all models experience a decline in performance due to the sudden change in data distribution. However, BTOA recovers more quickly and aligns more closely with the ground truth compared to OneNet, demonstrating its robustness in handling distributional shifts. As training progresses, BTOA not only adapts more rapidly but also captures complex temporal patterns more effectively, resulting in superior predictions. This highlights BTOA's advantage in mitigating the negative impacts of distributional shifts and underscores its ability to maintain, and even improve, predictive accuracy over time, making it a more reliable solution for online scenarios.

### 4.2 Ablation Study

In this section, we will examine the contribution of each module in BTOA individually. The first module is Transferable Historical Sample Selection module, which selects historical samples that are close to the information of the test-time sample, a critical factor in fully leveraging historical information. The second module is the Transferable Online Augmentation module, which performs batch training to help mitigate noise present in the test-time sample and reduce distribution shifts. More ablation results are provided in Appendix B.4.

Table 4: Comparison of Historical Sample Selection Methods.

| Dataset | ETTh1 | | | ETTm2 | | | WTH | | | ECL | | | Traffic | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 |
| closest-16 | 0.237 | 0.391 | 0.410 | 0.220 | 0.256 | 0.279 | 0.192 | 0.232 | 0.241 | 2.859 | 2.988 | 2.713 | 0.269 | 0.458 | 0.469 |
| L2-distance | 0.292 | 0.320 | 0.276 | 0.194 | 0.232 | 0.227 | 0.158 | 0.215 | 0.226 | 3.285 | 3.128 | 2.923 | 0.238 | 0.412 | 0.470 |
| THSS (ours) | **0.223** | **0.271** | **0.265** | **0.174** | **0.206** | **0.204** | **0.156** | **0.161** | **0.174** | **2.430** | **2.493** | **2.432** | **0.232** | **0.310** | **0.351** |

**The Components of BTOA.** First, we examine the THSS module. We conducted a comparison between three methods: (1) directly selecting the 16 most recent samples in time closest to the test-time sample for data augmentation, without choosing from the memory bank; (2) using L2 distance as a representative naive distance metric for the selection criterion; and (3) using the THSS module as the selection standard. As shown in Table 4, the THSS module outperforms the other two methods in selecting historical samples. This result demonstrates that the VAE successfully captures the semantic information of time series in the latent space, confirming the effectiveness of the latent space.

Table 5: Comparison of Data Augmentation Methods.

| Dataset | ETTh1 | | | ETTm2 | | | WTH | | | ECL | | | Traffic | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 |
| Non-Aug | 0.233 | 0.381 | 0.407 | 0.193 | 0.274 | 0.280 | 0.158 | 0.187 | 0.196 | 2.523 | 2.998 | 3.309 | 0.247 | 0.421 | 0.445 |
| Linear-Mixup | 0.388 | 0.460 | 0.336 | 0.180 | 0.234 | 0.227 | 0.174 | 0.228 | 0.292 | 2.987 | 3.091 | 3.112 | 0.302 | 0.398 | 0.422 |
| Cut-Mixup | 0.273 | 0.321 | **0.263** | 0.178 | 0.227 | 0.262 | 0.159 | 0.240 | 0.337 | 2.549 | 2.762 | 2.918 | 0.265 | 0.332 | 0.384 |
| TOA (ours) | **0.223** | **0.271** | 0.265 | **0.174** | **0.206** | **0.204** | **0.156** | **0.161** | **0.174** | **2.430** | **2.493** | **2.432** | **0.232** | **0.310** | **0.351** |

Second, we evaluate the effectiveness of the TOA module. We compare four methods on five datasets: (1) without data augmentation; (2) using Linear-Mixup (Zhang et al., 2017); (3) using Cut-Mixup (Yun et al., 2019); and (4) using our proposed TOA module. Table 5 demonstrates the superior performance of the TOA method across all five datasets. This indicates that, for time series, simple time domain data augmentation can easily lead to interference between different series. In contrast, the TOA method performs data augmentation in the frequency domain, focusing on amplitude and phase, reducing the interference between time series and achieving positive data augmentation, thereby improving prediction performance.

Table 6: The Generality of BTOA.

| Dataset | ETTh1 | | | ETTm2 | | | WTH | | | ECL | | | Traffic | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 |
| Revin | 0.238 | 0.672 | 0.792 | 0.173 | 0.652 | 1.083 | 0.162 | 0.370 | 0.452 | 3.873 | 4.862 | 6.583 | 0.257 | 1.097 | 1.678 |
| Revin+BTOA | 0.236 | 0.633 | 0.723 | 0.172 | 0.650 | 0.927 | 0.160 | 0.352 | 0.444 | 3.675 | 4.563 | 6.132 | 0.271 | 0.954 | 1.239 |
| PatchTST | 0.246 | 0.810 | 0.831 | 0.184 | 0.547 | 0.608 | 0.162 | 0.372 | 0.465 | 2.022 | 4.325 | 5.030 | 0.322 | 0.913 | 1.519 |
| PatchTST+BTOA | 0.239 | 0.711 | 0.745 | 0.173 | 0.513 | 0.592 | 0.156 | 0.331 | 0.397 | 2.329 | 4.154 | 4.958 | 0.276 | 0.621 | 0.686 |
| FsNet | 0.286 | 0.411 | 0.402 | 0.179 | 0.233 | 0.299 | 0.161 | 0.189 | 0.223 | 3.317 | 6.071 | 7.234 | 0.295 | 0.360 | 0.378 |
| FsNet+BTOA | 0.308 | 0.299 | 0.300 | 0.189 | 0.227 | 0.286 | 0.162 | 0.181 | 0.213 | 4.122 | 5.534 | 6.071 | 0.283 | 0.335 | 0.354 |

**Generalisability.** As a plug-and-play module, BTOA has demonstrated significant effectiveness in improving model performance. In Table 6, we compare the results of adding BTOA to the Revin, PatchTST, and FsNet models. The experimental results show that, in all models, the performance after adding BTOA outperforms the original models without BTOA. This result fully validates the effectiveness of BTOA in enhancing the model's generalization ability and overall performance. Specifically, BTOA improves the model's robustness against performance degradation caused by distribution shifts in online time series prediction by implementing batch-training. Additionally, the inclusion of BTOA also positively impacts the stability of model training. Therefore, BTOA, as a general-purpose module, not only improves the performance of various models but also enhances their feasibility and stability in real-world applications.

## 5 Conclusion

In this paper, a general online test-time adaptation method, Batch training with Transferable Online Augmentation (BTOA) for time series, is designed to effectively mitigate the performance degradation caused by distribution shifts during the test process. Our approach incorporates a transferable historical sample selection module, a transferable online augmentation module, and a prediction block. The THSS module fully leverages the distributional information from historical samples. Through the TOA module, we reintroduce batch training in the online setting to alleviate the negative impact of distribution shifts. Finally, the prediction block extracts complex patterns in time series, enabling more accurate outputs. Extensive experiments demonstrate that our approach can be integrated into any online time series forecasting model and achieves superior performance.

### Broader Impact Statement

Our work proposes a plug-and-play module from the perspective of online test-time adaptation to mitigate the negative impact of distribution shift. This is crucial for practical time series forecasting tasks. This research can serve as a reference for future studies in machine learning and data science, promoting the development of more complex and accurate online time series forecasting techniques. Therefore, our paper primarily focuses on scientific research and does not have any obvious negative social impact.

## References

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019a.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11254–11263, 2019b.

Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir. Online learning for time series prediction. In *Conference on learning theory*, pp. 172–184. PMLR, 2013.

Oliver D Anderson. Time-series. 2nd edn., 1976.

Sergul Aydore, Tianhao Zhu, and Dean P Foster. Dynamic local regret for non-convex online forecasting. *Advances in neural information processing systems*, 32, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

Berken Utku Demirel and Christian Holz. Finding order in chaos: A novel data augmentation method for time series in contrastive learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Alberto Gasparin, Slobodan Lukovic, and Cesare Alippi. Deep learning for time series forecasting: The electric load case. *CAAI Transactions on Intelligence Technology*, 7(1):1–25, 2022.

Taesik Gong, Yewon Kim, Taeckyung Lee, Sorn Chottananurak, and Sung-Ju Lee. Sotta: Robust test-time adaptation on noisy data streams. *Advances in Neural Information Processing Systems*, 36, 2024.

San Gultekin and John Paisley. Online forecasting matrix factorization. *IEEE Transactions on Signal Processing*, 67(5):1223–1236, 2018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

Minguk Jang, Sae-Young Chung, and Hye Won Chung. Test-time adaptation via self-training with nearest neighbor information. *arXiv preprint arXiv:2207.10792*, 2022.

KyoHoon Jin, JeongA Wi, EunJu Lee, ShinJin Kang, SooKyun Kim, and YoungBin Kim. Trafficbert: Pre-trained model with large-scale data for long-range traffic flow forecasting. *Expert Systems with Applications*, 186:115738, 2021.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jangho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL `https://api.semanticscholar.org/CorpusID:251647808`.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Yinqi Li, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Optimal positive generation via latent transformation for contrastive learning. *Advances in Neural Information Processing Systems*, 35: 18327–18342, 2022.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023.

Jiaming Liu, Senqiao Yang, Peidong Jia, Ming Lu, Yandong Guo, Wei Xue, and Shanghang Zhang. Vida: Homeostatic visual domain adapter for continual test time adaptation. *arXiv preprint arXiv:2306.04344*, 2023.

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.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*, 2021.

A Tuan Nguyen, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip HS Torr. Tipi: Test time adaptation with transformation invariance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24162–24171, 2023.

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*, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven Hoi. Learning fast and slow for online time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. Dataset shift in machine learning. 2009. URL `https://api.semanticscholar.org/CorpusID:61294087`.

CLEVELAND RB. Stl: A seasonal-trend decomposition procedure based on loess. *J Off Stat*, 6:3–73, 1990.

Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022a.

Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022b.

Martin Ullrich, Arne Küderle, Julius Hannink, Silvia Del Din, Heiko Gassner, Franz Marxreiter, Jochen Klucken, Bjoern M Eskofier, and Felix Kluge. Detection of gait from continuous inertial sensor data using harmonic frequencies. *IEEE Journal of Biomedical and Health Informatics*, 24(7):1869–1878, 2020.

Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*, pp. 10530–10541. PMLR, 2021.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2020.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022a.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022b.

Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey. *arXiv preprint arXiv:2310.20199*, 2023.

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.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Longbin Zeng, Jiayi Han, Liang Du, and Weiyang Ding. Rethinking precision of pseudo label: Test-time adaptation via complementary learning. *Pattern Recognition Letters*, 177:96–102, 2024.

Gang Zhang, Dazhi Yang, George Galanis, and Emmanouil Androulakis. Solar forecasting with hourly updated numerical weather prediction. *Renewable and Sustainable Energy Reviews*, 154:111768, 2022a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35:38629–38642, 2022b.

Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35:3988–4003, 2022c.

YiFan 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 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11106–11115, Sep 2022.

# A Appendix

# B Additional experimental details

## B.1 Datasets

We conduct experiments on 7 real-world datasets to evaluate the performance of our method including (1) ETT(Electricity Transformer Temperature) (Wu et al., 2021) are collected from two electricity transformers with 7 factors. There are four subsets where ETTh1 and ETTh2 are recorded every hour, and ETTm1 and ETTm2 are recorded every 15 minutes. (2) ECL (Wu et al., 2021) collects 321 customers' hourly electricity consumption. (3) Traffic (Wu et al., 2021)collects the road occupancy rates from different sensors on San Francisco freeways. (4) WTH (Pham et al., 2022)Collects hourly records of 11 climate features from nearly 1,600 locations in the U.S. from 2010 to 2013.

## B.2 Implementation Details

All experiments are implemented in PyTorch and conducted on a single NVIDIA RTX4090 24GB GPU. The learning rate for the experiments is set between 1e-3 and 7e-3.

## B.3 Analysis of Inference Time and Memory Consumption

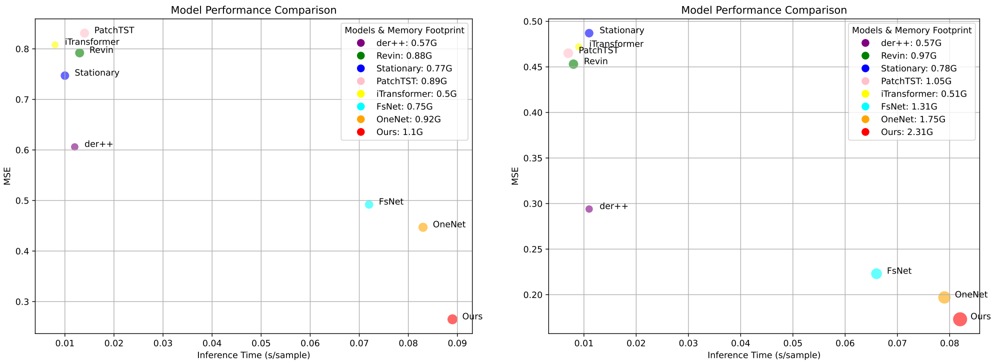


Figure 4: Comparison of inference time and memory consumption on ETTh1 and WTH.

Since online test-time adaptation deals with streaming data that arrives in a continuous and sequential manner, Figure 4 compares the inference time for a single sample and overall memory usage between our method and the baselines. The results show that BTOA introduces minimal increases in inference time and memory usage, which are acceptable in an online setting, while outperforming the baseline methods. Notably, for the VAE module during inference, BTOA only requires simple forward propagation without the need for backpropagation. On the ETTh1 dataset, BTOA improves prediction performance by 40% while only increasing inference time by 7%, whereas OneNet increases inference time by 13% while only improving prediction performance by 9%. This comparison further highlights the superiority of BTOA.

## B.4 Ablation Study

Table 7: Performance Impact of Series Decomposition Block.

| Dataset | ETTh1 | | | ETTm2 | | | WTH | | |
|---|---|---|---|---|---|---|---|---|---|
| Methods | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 |
| Non-SD | 0.230 | 0.280 | 0.273 | 0.180 | 0.212 | 0.210 | 0.163 | 0.169 | 0.181 |
| SD | **0.223** | **0.271** | **0.265** | **0.174** | **0.206** | **0.204** | **0.156** | **0.161** | **0.174** |

we evaluate the effect of using a series decomposition module in the prediction block by comparing the performance of models with and without series decomposition. Although OneNet suggests that series decomposition is not a universally effective method for improving performance, Table 7 shows that for the BTOA model, series decomposition can effectively extract complex temporal patterns and enhance the model's predictive ability.

Table 8: Performance Impact of different distance threshold.

| Dataset | ETTh1 | | | WTH | | | Traffic | | |
|---|---|---|---|---|---|---|---|---|---|
| distance | 1 | 24 | 48 | 1 | 24 | 48 | 1 | 24 | 48 |
| 0.7 | 0.241 | 0.295 | 0.259 | 0.157 | 0.169 | **0.171** | 0.249 | **0.306** | 0.362 |
| 0.8 | **0.223** | **0.271** | **0.265** | **0.156** | **0.161** | 0.174 | **0.232** | 0.310 | **0.351** |
| 0.9 | 0.233 | 0.272 | 0.266 | 0.159 | 0.170 | 0.182 | 0.262 | 0.316 | 0.355 |

We also conduct ablation experiments regarding the distance threshold hyperparameter as shown in Table 8, which determines from which distribution $\lambda_A$ and $\lambda_P$ are sampled. If the distance between the historical sample and the newly received sample is below a predefined distance threshold (set to 0.8 in our experiment), $\lambda_A, \lambda_P$ are sampled from a uniform distribution with a lower mean. If the distance between the historical sample and the newly received sample exceeds the predefined distance threshold, $\lambda_A, \lambda_P$ are sampled from a truncated Gaussian distribution with a higher mean and lower standard deviation.

### B.5 VAE models architecture

We use the Total Correlation Variational Autoencoder (TC-VAE) (Chen et al., 2018) to compute the distance in latent space between the current incoming sample and the historical samples stored in the memory bank. The model is trained for 100 epochs with a learning rate of 3e-3, using the Evidence Lower Bound (ELBO) as the loss function, and a batch size of 128. The latent dimensions and the $\beta$ parameter are set to 10 and 5, respectively. Below, we provide detailed information about the encoder and decoder architectures, which differ across datasets due to variations in input channels.

Table 9: Encoder Network for ETT dataset

| **Layer Name** | **Output size** | **# of kernels** | **Kernel size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Input | $N \times 1 \times 60 \times 7$ | | | | |
| Convolution | $N \times 32 \times 26 \times 5$ | 32 | $9 \times 3$ | $2 \times 1$ | ReLU |
| Convolution | $N \times 32 \times 10 \times 3$ | 32 | $7 \times 3$ | $2 \times 1$ | ReLU |
| Convolution | $N \times 64 \times 2 \times 1$ | 64 | $5 \times 3$ | $3 \times 1$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $2 \times 1$ | $1 \times 1$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $1 \times 1$ | $1 \times 1$ | |

Table 10: Decoder Network for ETT dataset

| **Layer Name** | **Output size** | **# of kernels** | **Kernel size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Input | $N \times 10 \times 1 \times 1$ | | | | |
| Transposed Convolution | $N \times 512 \times 2 \times 7$ | 512 | $2 \times 7$ | $1 \times 1$ | ReLU |
| Transposed Convolution | $N \times 128 \times 2 \times 7$ | 128 | $4 \times 1$ | $6 \times 1$ | ReLU |
| Transposed Convolution | $N \times 64 \times 16 \times 7$ | 64 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 32 \times 7$ | 32 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 1 \times 7$ | 1 | $4 \times 1$ | $2 \times 1$ | |

Table 11: Encoder Network for WTH dataset

| Layer Name | Output size | # of kernels | Kernel size | Stride | Activation |
|---|---|---|---|---|---|
| Input | $N \times 1 \times 60 \times 12$ | | | | |
| Convolution | $N \times 32 \times 18 \times 11$ | 32 | $9 \times 2$ | $3 \times 1$ | ReLU |
| Convolution | $N \times 32 \times 6 \times 5$ | 32 | $3 \times 3$ | $3 \times 2$ | ReLU |
| Convolution | $N \times 64 \times 2 \times 5$ | 64 | $3 \times 3$ | $3 \times 2$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $2 \times 2$ | $2 \times 1$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $1 \times 1$ | $1 \times 1$ | |

Table 12: Decoder Network for WTH dataset

| Layer Name | Output size | # of kernels | Kernel size | Stride | Activation |
|---|---|---|---|---|---|
| Input | $N \times 10 \times 1 \times 1$ | | | | |
| Transposed Convolution | $N \times 512 \times 2 \times 12$ | 512 | $2 \times 7$ | $1 \times 1$ | ReLU |
| Transposed Convolution | $N \times 128 \times 2 \times 12$ | 128 | $4 \times 1$ | $6 \times 1$ | ReLU |
| Transposed Convolution | $N \times 64 \times 16 \times 12$ | 64 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 32 \times 12$ | 32 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 1 \times 12$ | 1 | $4 \times 1$ | $2 \times 1$ | |

Table 13: Encoder Network for ECL dataset

| Layer Name | Output size | # of kernels | Kernel size | Stride | Activation |
|---|---|---|---|---|---|
| Input | $N \times 1 \times 60 \times 321$ | | | | |
| Convolution | $N \times 32 \times 18 \times 64$ | 32 | $9 \times 2$ | $3 \times 5$ | ReLU |
| Convolution | $N \times 32 \times 6 \times 13$ | 32 | $3 \times 3$ | $3 \times 5$ | ReLU |
| Convolution | $N \times 64 \times 2 \times 3$ | 64 | $3 \times 3$ | $3 \times 5$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $2 \times 2$ | $2 \times 3$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $1 \times 1$ | $1 \times 1$ | |

Table 14: Decoder Network for ECL dataset

| Layer Name | Output size | # of kernels | Kernel size | Stride | Activation |
|---|---|---|---|---|---|
| Input | $N \times 10 \times 1 \times 1$ | | | | |
| Transposed Convolution | $N \times 512 \times 2 \times 321$ | 512 | $2 \times 321$ | $1 \times 1$ | ReLU |
| Transposed Convolution | $N \times 128 \times 2 \times 321$ | 128 | $4 \times 1$ | $6 \times 1$ | ReLU |
| Transposed Convolution | $N \times 64 \times 16 \times 321$ | 64 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 32 \times 321$ | 32 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 1 \times 321$ | 1 | $4 \times 1$ | $2 \times 1$ | |

Table 15: Encoder Network for traffic dataset

| **Layer Name** | **Output size** | **# of kernels** | **Kernel size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Input | $N \times 1 \times 60 \times 862$ | | | | |
| Convolution | $N \times 32 \times 18 \times 173$ | 32 | $9 \times 2$ | $3 \times 5$ | ReLU |
| Convolution | $N \times 32 \times 6 \times 35$ | 32 | $3 \times 3$ | $3 \times 5$ | ReLU |
| Convolution | $N \times 64 \times 2 \times 7$ | 64 | $3 \times 3$ | $3 \times 5$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $2 \times 2$ | $2 \times 5$ | ReLU |
| Convolution | $N \times 128 \times 1 \times 1$ | 128 | $1 \times 1$ | $1 \times 1$ | |

Table 16: Decoder Network for traffic dataset

| **Layer Name** | **Output size** | **# of kernels** | **Kernel size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Input | $N \times 10 \times 1 \times 1$ | | | | |
| Transposed Convolution | $N \times 512 \times 2 \times 862$ | 512 | $2 \times 862$ | $1 \times 1$ | ReLU |
| Transposed Convolution | $N \times 128 \times 2 \times 862$ | 128 | $4 \times 1$ | $6 \times 1$ | ReLU |
| Transposed Convolution | $N \times 64 \times 16 \times 862$ | 64 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 32 \times 862$ | 32 | $4 \times 1$ | $2 \times 1$ | ReLU |
| Transposed Convolution | $N \times 32 \times 1 \times 862$ | 1 | $4 \times 1$ | $2 \times 1$ | |