OneNet: Enhancing Time Series Forecasting Models under Concept Drift by Online Ensembling

————Appendix————

Contents

A Extended Related Work

Concept Drift Concepts in the real world are often dynamic and can change over time, which is especially true for scenarios like weather prediction and customer preferences. Because of unknown changes in the underlying data distribution, models learned from historical data may become inconsistent with new data, thus requiring regular updates to maintain accuracy. This phenomenon, known as concept drift [41], adds complexity to the process of learning a model from data. Concept drift poses several challenging subproblems, ranging from fast learning under concept drift [33, 46, 19], which involves adjusting the offline model with new observations to recognize recent patterns, to forecasting future data distributions [27, 35], which predicts the data distribution of the next time-step sequentially, enabling the model of the downstream learning task to be trained on the data sample from the predicted distribution. In this paper, we focus on the first problem, i.e. online learning for time series forecasting. Unlike most existing studies for online time series forecasting [27, 35, 33] that only focus on how to online update their models, this work goes beyond parameter updating and introduces multiple models and a learnable ensembling weight, yielding rich and flexible hypothesis space.

Time Series Modeling Time series models have been developed for decades and are fundamental in various fields. While autoregressive models like ARIMA [8] were the first data-driven approaches, they struggle with nonlinearity and non-stationarity. Recurrent neural networks (RNNs) were designed to handle sequential data, with LSTM [21] and GRU [14] using gated structures to address gradient problems. Attention-based RNNs [36] use temporal attention to capture long-range dependencies, but are not parallelizable and struggle with long dependencies. Temporal convolutional networks [37] are efficient, but have limited reception fields and struggle with long-term dependencies. Recently, transformer-based [42, 43] models have been renovated and applied in time series forecasting. Although a large body of work aims to make Transformer models more efficient and powerful [49, 48, 32], we are the first to evaluate the robustness of advanced forecasting models under concept drift, making them more adaptable to new distributions.

Reinforcement learning and offline reinforcement learning. Reinforcement learning is a mathematical framework for learning-based control, which allows us to automatically acquire policies that represent near-optimal behavioral skills to optimize user-defined reward functions [22, 39]. The reward function specifies the objective of the agent, and the reinforcement learning algorithm determines the actions necessary to achieve it. However, the online learning paradigm of reinforcement learning is a major obstacle to its widespread adoption. The iterative process of collecting experience by interacting with the environment is expensive or dangerous in many settings, making offline reinforcement learning a more feasible alternative. Offline RL [34, 26] learns exclusively from static datasets of previously collected interactions, enabling the extraction of policies from large and diverse training datasets. Effective offline RL algorithms have a much wider range of applications than online RL. Although there are many types of offline RL algorithms, such as those using value functions [18], dynamics estimation [25], or uncertainty quantification [1], RvS [15] has shown that simple conditioning with standard feedforward networks can achieve state-of-the-art results. In this work, we draw inspiration from RvS and learn an additional bias term for the OCP block for simplicity.

B Proofs of Theoretical Statements

B.1 Online Convex Programming Regret Bound

Proposition 1. *For $T > 2\log(d)$, denote the regret for time step $t = 1, \ldots, T$ as $R(T)$, set $\eta = \sqrt{2\log(d)/T}$, and the EGD update policy has regret.*

$$R(T) = \sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \inf_{\mathbf{u}} \sum_{t=1}^{T} \mathcal{L}(\mathbf{u}) \leq \sum_{t=1}^{T} \sum_{i=1}^{d} w_{t,i} \parallel f_i(\mathbf{x}) - \mathbf{y} \parallel^2 - \sum_{t=1}^{T} \inf_{\mathbf{u}} \mathcal{L}(\mathbf{u}) \leq \sqrt{2T\log(d)} \tag{5}$$

Proof. The proof of Exponentiated Gradient Descent is well-studied [20] and here we provide a simple Regret bound for both OCP.

Denote $\ell_{t,i} = \| f_i(\mathbf{x}) - \mathbf{y} \|^2$, recall that the normalizer for OCP-U is $Z_{t+1} = \sum_{i=1}^d w_{t,i} \exp(-\eta\ell_{t,i})$, then we have

$$\log \frac{Z_{t+1}}{Z_t} = \log \frac{\sum_{i=1}^d w_{t,i} \exp(-\eta\ell_{t,i})}{Z_t} = \log \sum_{i=1}^d p_{t,i} \exp(-\eta\ell_{t,i}), \tag{6}$$

where $p_{t,i} = w_{t,i}/Z_t \leq 1$. For clarity, we let $w_{t,i}$ be the unnormalized weight and $p_{t,i}$ be the normalized weight. Now, we assume $\eta\ell_{t,i} \in [0, 1]$. Although it is not guaranteed that $\ell_{t,i}$ will be small under concept shift, it is generally safe to assume that the concept will shift gradually and will not lead to a drastic change in the loss. As a result, the loss will not become arbitrarily large, and we can divide some large constant such that the loss is bounded in a small range. Based on the assumption, we can use the second Taylor expansion of $e^{-x} \leq 1 - x + x^2/2$ and the inequation $\log(1 - x) \leq -x$ for $x \in [0, 1]$. Then we have

$$\log \sum_{i=1}^d p_{t,i} \exp(-\eta\ell_{t,i}) \leq \log \left(1 - \eta \sum_{i=1}^d p_{t,i}\ell_{t,i} + \frac{\eta^2}{2} \sum_{i=1}^d p_{t,i}\ell_{t,i}^2\right) \tag{7}$$

$$\leq -\eta \sum_{i=1}^d p_{t,i}\ell_{t,i} + \frac{\eta^2}{2} \sum_{i=1}^d p_{t,i}\ell_{t,i}^2 \leq -\eta \sum_{i=1}^d p_{t,i}\ell_{t,i} + \frac{\eta^2}{2} \tag{8}$$

Note that $w_{t,i}$ is the unnormalized weight, and then $w_{1,i} = 1$ and $Z_1 = d$. Then we can get the lower bound and upper bound of $\log Z_{T+1}$:

$$\log Z_{T+1} = \sum_{t=1}^T \log \frac{Z_{t+1}}{Z_t} + \log(Z_1) \leq -\eta \sum_{t=1}^T \sum_{i=1}^d p_{t,i}\ell_{t,i} + \frac{T\eta^2}{2} + \log(d) \tag{9}$$

$$\log Z_{T+1} = \log \sum_{i=1}^d w_{t,i} \exp(-\eta\ell_{t,i}) \geq -\eta \sum_{i=1}^d \ell_{t,i} \tag{10}$$

Finally, recall that we set $\eta = \sqrt{2\log(d)/T}$, then we have

$$\eta \left(\sum_{t=1}^T \sum_{i=1}^d p_{t,i}\ell_{t,i} - \sum_{i=1}^d \ell_{t,i}\right) \leq \eta \left(\sum_{t=1}^T \sum_{i=1}^d p_{t,i}\ell_{t,i} - \inf_{\mathbf{u}} \sum_{t=1}^T \mathcal{L}(\mathbf{u})\right) \leq \frac{T\eta^2}{2} + \log(d), \tag{11}$$

which completes our proof. □

B.2 Theoretical guarantee for the K-step re-initialize algorithm.

The proposed K-step re-initialize algorithm is detailed as follows: at the beginning of the algorithm, we choose $\mathbf{w}_1 = [w_{1,i} = 1/d]_{i=1}^d$ as the center point of the simplex and denote $\ell_{t,i}$ as the loss for f_i at time step t, the updating rule for each w_i will be $w_{t+1,i} = \frac{w_{t,i} \exp(-\eta[\partial\mathcal{L}_U(\mathbf{w}_t)]_i)}{Z_t} = \frac{w_{t,i} \exp(-\eta\|f_i(\mathbf{x})-\mathbf{y}\|^2)}{Z_t} = \frac{w_{t,i} \exp(-\eta\ell_{t,i})}{Z_t}$, where $Z_t = \sum_{i=1}^d w_{t,i} \exp(-\eta l_{t,i})$ is the normalizer. Different from the native EGD algorithm, we re-initialize the weight $\mathbf{w}_{K+1} = [w_{K+1,i} = 1/d]_{i=1}^d$ per K time steps. We call each K step one round. This simple strategy interrupts the influence of the historical information of length K steps on the ensembling weights, which helps the model to quickly adapt to the upcoming environment.

Proposition 2. *For $T > 2\log(d)$, denote $I = [l, l+1, \cdots, r]$ as any period of time of length $r - l + 1$ where $l > 1$ and $r \leq T$. Denote the length of I as a sublinear sequence of T, namely, $|I| = T^n$, where $0 \leq n \leq 1$. We choose $K = T^{\frac{2n}{3}}$. We then have, the K-step re-initialize algorithm has an regret bound $R(I) \leq \mathcal{O}(T^{2n/3})$ at any $I = [l, l+1, \cdots, r]$. Namely, for any small internal $n < \frac{3}{4}$, we have $R([l, l+1, \cdots, r]) < \mathcal{O}(T^{1/2})$*

Proof. We discuss regret in three cases:

- At first, according to Proposition 1, if all K steps fall into $[l, l+1, \cdots, r]$, then we have $R(K) \leq 2\sqrt{K \ln d(d-1)}$. There exist L/K rounds that are all contained in $[l, ..., r]$ and the regret of these rounds will be $\mathcal{O}(L/K * 2\sqrt{K})$.

- For the first round, we do not know when the weights are reinitialized (l may or may not be a multiple of K) and the performance of the algorithm in historical time. let's think about the worst case, where regret will be less than $\mathcal{O}(K)$.

- For the last round, we know when the last round begins and the weights are reinitialized. However, some of the future time steps are not in $[l, l+1, \cdots, r]$ and we can only treat the case as the first round, which has a regret $\mathcal{O}(K)$.

Considering all cases, we have an internal regret that is bounded by

$$R(I) \leq \mathcal{O}(L/K \times \sqrt{K} + K) = \mathcal{O}(L/K\sqrt{K} + K) \tag{12}$$

When we choose an small internal length $L = T^n$ and $K = T^m$ where $m \leq n$. To minimize the upper bound, we should choose $m = \frac{2n}{3}$ and the bound will be $\mathcal{O}(T^{2n/3})$. Namely, for any small internal $n < \frac{3}{4}$, we have $R([l, l+1, \cdots, r]) < \mathcal{O}(T^{1/2})$, which is tighter than the regret bound of the EGD algorithm under the whole online sequence. Specifically, when we choose $L = T^{2/3}$ and $K = T^{4/9}$, we have $R([l, l+1, \cdots, r]) < \mathcal{O}(T^{4/9}) < \mathcal{O}(T^{1/2})$. When we choose $L = T^{1/4}$ and $K = T^{1/6}$, then $R([l, l+1, \cdots, r]) < \mathcal{O}(T^{1/6}) < \mathcal{O}(T^{1/2})$.

In other words, **the algorithm focuses on short-term information and leads to a better regret bound in any small time interval**. However, with increasing length of I, the bound of the simple algorithm will become worse. Consider an extreme case where $n = 1$, the bound will be $R([l, l+1, \cdots, r]) < \mathcal{O}(T^{2/3})$, which is inferior to the native EGD algorithm.

□

B.3 Necessary definitions and assumptions for evaluating model adaptation speed to environment changes.

To complete the proofs, we begin by introducing some necessary definitions and assumptions. Given the online data stream $\mathbf{x}_t$ and its forecasting target $\mathbf{y}_t$ at time t. Given d forecasting experts with different parameters $\mathbf{f}_t = \{f_{t,i}\}$, denote ℓ as a nonnegative loss function and $\ell_{t,i} := \ell(f_{t,i}(\mathbf{x}_t), \mathbf{y}_t)$ as the loss incurred by $f_{t,i}$ at time t, we define the following notions.

Definition 1. ***(Weighted average forecaster)***. *A weighted average forecaster makes predictions by*

$$\tilde{\mathbf{y}}_t = \frac{\sum_{i=1}^{d} w_{t-1,i} f_{t,i}}{\sum_{i=1}^{d} w_{t-1,i}}, \tag{13}$$

where $w_{t,i}$ is the weight for expert f_i at time t and $f_{t,i}$ is the prediction of f_i at time t.

Definition 2. ***(Cumulative regret and instantaneous regret)***. *For expert f_i, the cumulative regret (or simply regret) on the T steps is defined by*

$$R_{T,i} = \sum_{t=1}^{T} r_{t,i} = \sum_{t=1}^{T} \left(\ell(\tilde{\mathbf{y}}_t, \mathbf{y}_t) - \ell_{t,i}\right) = \hat{L}_T - L_{T,i} \tag{14}$$

where $r_{t,i}$ is the instantaneous regret of the expert f_i at time t, which is the regret that the forecaster feels of not having listened to the advice of the expert f_i right after the tth outcome that has been revealed. $\hat{L}_T = \sum_{t=1}^{T} \ell(\tilde{\mathbf{y}}_t, \mathbf{y}_t)$ is the cumulative loss of the forecaster and $L_{T,i}$ is the cumulative loss of the expert f_i.

Definition 3. ***(Potential function)***. *We can interpret the weighted average forecaster in an interesting way which allows us to analyze the theoretical properties easier. To do this, we denote $\mathbf{r}_t = (r_{t,1}, \ldots, r_{t,d}) \in \mathbb{R}^d$ as the instantaneous regret vector, and $\mathbf{R}_T = \sum_{t=1}^{T} \mathbf{r}_t$ is the corresponding regret vector. Now, we can introduce the potential function $\Phi : \mathbb{R}^d \to \mathbb{R}$ of the form*

$$\Phi(\mathbf{u}) = \psi\left(\sum_{i=1}^{d} \phi(u_i)\right) \tag{15}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is any nonnegative, increasing, and twice differentiable function, and $\psi : \mathbb{R} \to \mathbb{R}$ is nonnegative, strictly increasing, concave, and twice differentiable auxiliary function. With the notion of potential function, the prediction $\tilde{\mathbf{y}}_t$ will be

$$\tilde{\mathbf{y}}_t = \frac{\sum_{i=1}^{d} \nabla\Phi(\mathbf{R}_{t-1})_i f_{t,i}}{\sum_{i=1}^{d} \nabla\Phi(\mathbf{R}_{t-1})_i}, \tag{16}$$

where $\nabla\Phi(\mathbf{R}_{t-1})_i = \partial\Phi(\mathbf{R}_{t-1})/\partial R_{t-1,i}$. It is easy to prove that the exponentially weighted average forecaster used in Eq.(2) is based on the potential $\Phi_\eta(\mathbf{u}) = \frac{1}{\eta} \ln \left(\sum_{i=1}^{d} e^{\eta u_i} \right)$.

Theorem 1. ***(Blackwell condition, Lemma 2.1. in [11].)*** *If the loss function ℓ is convex in its first argument and we use $\mathbf{x}_1 \cdot \mathbf{x}_2$ denote the inner product of two vectors, then*

$$\sup_{\mathbf{y}_t} \mathbf{r}_t \cdot \nabla\Phi(\mathbf{R}_{t-1}) \leq 0 \tag{17}$$

The following theorem is applicable to any forecaster that satisfies the Blackwell condition, not limited to weighted average forecasters. Nevertheless, this theorem will lead to several interesting bounds for various variations of the weighted average forecaster.

Theorem 2. *(Theorem2.1 in [11].) Assume that a forecaster satisfies the Blackwell condition for a potential Φ, then for all $i = 1, \cdots$,*

$$\Phi(\mathbf{R}_T) \leq \Phi(0) + \frac{1}{2} \sum_{t=1}^{T} C(\mathbf{r}_t), \tag{18}$$

where

$$C(\mathbf{r}_t) = \sup_{\mathbf{u} \in \mathbb{R}^d} \psi' \left(\sum_{i=1}^{d} \phi(u_i) \right) \sum_{i=1}^{d} \phi''(u_i) r_{t,i}^2. \tag{19}$$

B.4 Existing theoretical intuition and empirical comparison to the proposed OCP block.

With the help of the two theorems in Section B.3, we now recall that the **Internal Regret** $R_{in}(t, \mathbf{w})$ [7] that measures forecaster's expected regret of having taken an action $\mathbf{w}$ at step t:

$$R_{in}(T, \mathbf{w}) = \max_{i,j=1,\ldots,d} \sum_{t=1}^{T} r_{t,(i,j)} = \max_{i,j=1,\ldots,d} \sum_{t=1}^{T} w_{t,i} \left(\ell_{t,i} - \ell_{t,j} \right). \tag{20}$$

While Proposition 1 ensures that a small external regret can be achieved, ensuring a small internal regret is a more challenging task. This is because any algorithm with a small internal regret also has small external regret but the opposite is not true, as demonstrated in [40]. The key question now is whether it is possible to define a policy $\mathbf{w}$ that attains small (i.e., sublinear in T) internal regret. For simplicity, we use R as internal regret in this subsection. To develop a forecasting strategy that can guarantee a small internal regret. We define the exponential potential function $\Phi : \mathbb{R}^M \to \mathbb{R}$ with $\eta > 0$ by

$$\Phi(\mathbf{u}) = \frac{1}{\eta} \ln \left(\sum_{i=1}^{M} e^{\eta u_i} \right), \tag{21}$$

where $M = d(d-1)$. Here, we denote $\mathbf{r}_t = (r_{t,(1,1)}, r_{t,(1,2)}, \ldots, r_{t,(d,d-1)}) \in \mathbb{R}^{d(d-1)}$ as the instantaneous regret vector and $\mathbf{R}_T = \sum_{t=1}^{T} \mathbf{r}_t$ is the corresponding regret vector. Then, any forecaster satisfying Blackwell's condition will have a bounded internal regret (Corollary 8 in [10]) by choosing a proper parameter η:

$$\max_{i,j} R_{t,(i,j)} \leq 2\sqrt{t \ln d(d-1)} \tag{22}$$

With the help of the two theorems in Section B.3, we now recall that the **Internal Regret** $R_{in}(t, \mathbf{w})$ [7] that measures forecaster's expected regret of having taken an action $\mathbf{w}$ at step t:

$$R_{in}(t, \mathbf{w}) = \max_{i,j=1,\ldots,d} \sum_{t=1}^{T} r_{t,(i,j)} = \max_{i,j=1,\ldots,d} \sum_{t=1}^{T} w_{t,i} \left(\ell_{t,i} - \ell_{t,j} \right). \tag{23}$$

For simplicity, we use R as internal regret in this subsection. To conduct a forecasting strategy that can guarantee a small internal regret. We define the exponential potential function $\Phi : \mathbb{R}^M \to \mathbb{R}$ with $\eta > 0$ by

$$\Phi(\mathbf{u}) = \frac{1}{\eta} \ln \left(\sum_{i=1}^{M} e^{\eta u_i} \right), \tag{24}$$

where $M = d(d-1)$. Here, we denote $\mathbf{r}_t = (r_{t,(1,1)}, r_{t,(1,2)}, \ldots, r_{t,(d,d-1)}) \in \mathbb{R}^{d(d-1)}$ as the instantaneous regret vector and $\mathbf{R}_T = \sum_{t=1}^{T} \mathbf{r}_t$ is the corresponding regret vector. Then, any forecaster satisfying Blackwell's condition will have a bounded internal regret (Corollary 8 in [10]) by choosing a proper parameter η:

$$\max_{i,j} R_{t,(i,j)} \le 2\sqrt{t \ln d(d-1)} \tag{25}$$

Now our target is to find a new policy that makes the forecaster satisfy the Blackwell condition.

$$\begin{aligned}
\nabla\Phi(\mathbf{R}_{t-1}) \cdot \mathbf{r}_t &= \sum_{i,j=1}^{d} \nabla_{(i,j)} \Phi(\mathbf{R}_{t-1}) w_{t,i} \left(\ell_{t,i} - \ell_{t,j}\right) \\
&= \sum_{i=1}^{d} \sum_{j=1}^{d} \nabla_{(i,j)} \Phi(\mathbf{R}_{t-1}) w_{t,i} \ell_{t,i} - \sum_{i=1}^{d} \sum_{j=1}^{d} \nabla_{(i,j)} \Phi(\mathbf{R}_{t-1}) w_{t,i} \ell_{t,j} \\
&= \sum_{i=1}^{d} \sum_{j=1}^{d} \nabla_{(i,j)} \Phi(\mathbf{R}_{t-1}) w_{t,i} \ell_{t,i} - \sum_{j=1}^{d} \sum_{i=1}^{d} \nabla_{(j,i)} \Phi(\mathbf{R}_{t-1}) w_{t,j} \ell_{t,i} \\
&= \sum_{i=1}^{d} \ell_{t,i} \left(\sum_{j=1}^{d} \nabla_{(i,j)} \Phi(\mathbf{R}_{t-1}) w_{t,i} - \sum_{k=1}^{d} \nabla_{(k,i)} \Phi(\mathbf{R}_{t-1}) w_{t,k} \right)
\end{aligned} \tag{26}$$

To ensure that this value is negative or zero, it is sufficient to demand that.

$$\sum_{i=1}^{d} \ell_{t,i} \left(\sum_{j=1}^{d} \nabla_{(i,j)} \Phi(\mathbf{R}_{t-1}) w_{t,i} - \sum_{k=1}^{d} \nabla_{(k,i)} \Phi(\mathbf{R}_{t-1}) w_{t,k} \right) = 0, \forall i = 1, \cdots, d \tag{27}$$

That is, we need to find a new policy vector $\mathbf{w}_t$ that satisfies $\mathbf{w}_t^T A = 0$, where $A = \begin{cases} -\nabla_{k,i} \Phi(\mathbf{R}_{t-1}) & \text{if } i \neq k, \\ \sum_{j \neq i} \nabla_{k,j} \Phi(\mathbf{R}_{t-1}) & \text{Otherwise.} \end{cases}$ is an $d \times d$ matrix. However, determining the existence of a new policy vector and efficiently calculating its values can be challenging. Even if we assume the new vector exists, the time complexity of calculating the new policy by the Gaussian elimination method[16] is $O(d^3)$, which is expensive, particularly for datasets with a large number of variables such as the ECL dataset with 321 variables and 321*2 policies. To address this issue, we propose the OCP block which utilizes an additional offline reinforcement learning block f_{rl} with parameter θ_{rl} to learn a bias vector $\mathbf{b}_t$ for the original policy $\mathbf{w}_t$. The new policy vector is then defined as $\tilde{\mathbf{w}}_t = \mathbf{w}_t + \mathbf{b}_t$. The learned bias pushes the predicted outcomes closer to the ground truth values, that is, we minimize $\min_{\theta_{rl}} \mathcal{L}(\tilde{\mathbf{w}}) := \| \sum_{i=1}^{d} \tilde{w}_i f_i(\mathbf{x}) - \mathbf{y} \|^2; \quad s.t. \quad \tilde{\mathbf{w}} \in \triangle$ to train θ_{rl}. We measure the internal regret $\max_{i,j=1,\ldots,d} w_{t,i}(\ell_{t,i} - \ell_{t,j})$ at each time step empirically. As shown in Figure 5, the proposed method significantly reduces internal regret without the need for constructing and computing a large matrix.

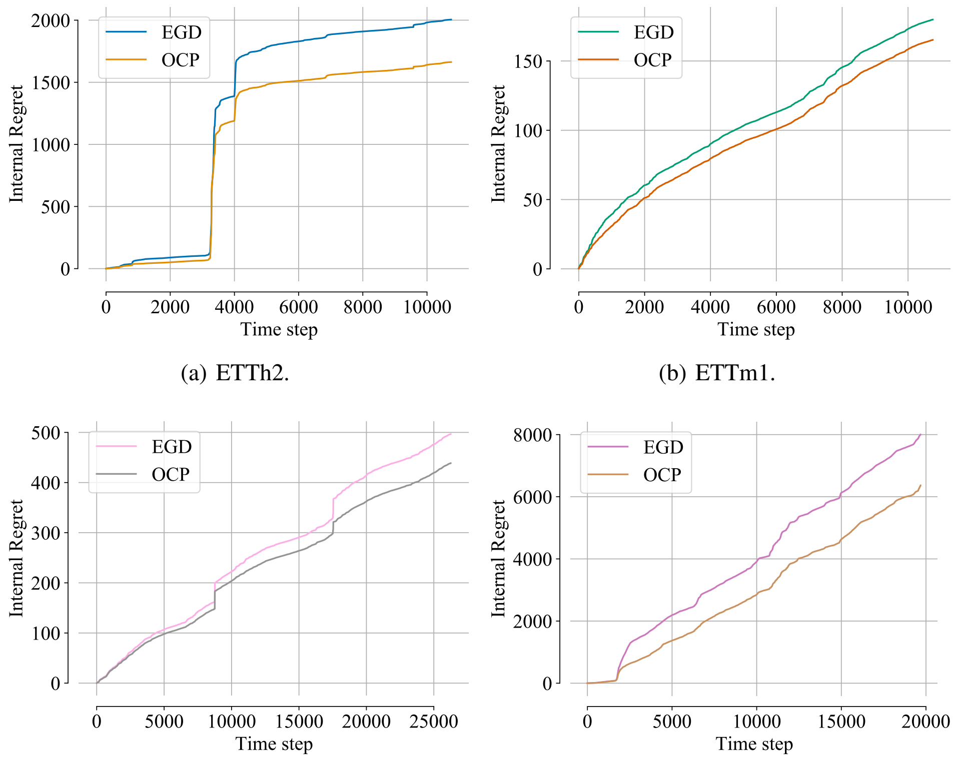
C Additional Experimental Results

C.1 Datasets

We investigate a diverse set of datasets for time series forecasting. ETT [48][1] logs the target variable of the "oil temperature" and six features of the power load over a two-year period. We also analyze the hourly recorded observations of ETTh2 and the 15-minute intervals of ETTm1 benchmarks. Additionally, we study ECL[2] (Electricity Consuming Load), which gathers electricity consumption

[1] https://github.com/zhouhaoyi/ETDataset

[2] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014



(a) ETTh2. (b) ETTm1.

(c) WTH. (d) ECL.

Figure 5: Empirical verification of the proposed OCP block can significantly reduce the internal regret compared to vanilla EGD, where the forecasting window $H = 48$.

data from 321 clients between 2012 and 2014. The Weather (WTH)[3] dataset contains hourly records of 11 climate features from almost 1,600 locations across the United States.

C.2 Implementation Details

For all benchmarks, we set the look-back window length at 60 and vary the forecast horizon from $H = 1, 24, 48$. We split the data into two phases: warm-up and online training, with a ratio of 25:75. We follow the optimization details outlined in [48] and utilize the AdamW optimizer [31] to minimize the mean squared error (MSE) loss. To ensure a fair comparison, we set the epoch and batch sizes to one, which is consistent with the online learning setting. We make sure that all baseline models based on the TCN backbone use the same total memory budget as FSNet, which includes three times the network sizes: one working model and two exponential moving averages (EMAs) of its gradient. For ER, MIR, and DER++, we allocate an episodic memory to store previous samples to meet this budget. For transformer backbones, we find that a large number of parameters do not benefit the generalization results and always select the hyperparameters such that the number of parameters for transformer baselines is fewer than that for FSNet. In the warm-up phase, we calculate the mean and standard deviation to normalize the online training samples and perform hyperparameter cross-validation. For different structures, we use the optimal hyperparameters that are reported in the corresponding paper.

License. All the assets (i.e., datasets and the codes for baselines) we use include an MIT license containing a copyright notice and this permission notice shall be included in all copies or substantial portions of the software.

[3] https://www.ncei.noaa.gov/data/local-climatological-data/

Environment. We conduct all the experiments on a machine with an Intel R Xeon (R) Platinum 8163 CPU @ 2.50GHZ, 32G RAM, and four Tesla-V100 (32G) instances. All experiments are repeated 3 times with different seeds.

Metrics Because learning occurs over a sequence of rounds. At each round, the model receives a look-back window and predicts the forecast window. All models are commonly evaluated by their accumulated mean-squared errors (MSE) and mean-absolute errors (MAE), namely the model is evaluated based on its accumulated errors over the entire learning process.

C.3 Baseline details

We present a brief overview of the baselines employed in our experiments.

First, OnlineTCN adopts a conventional TCN backbone [50] consisting of ten hidden layers, each layer containing two stacks of residual convolution filters.

Secondly, ER [12] expands on the OnlineTCN baseline by adding an episodic memory that stores previous samples and interleaves them during the learning process with newer ones.

Third, MIR [3] replaces the random sampling technique of ER with its MIR sampling approach, which selects the samples in the memory that cause the highest forgetting and applies ER to them.

Fourthly, DER++ [9] enhances the standard ER method by incorporating a knowledge distillation loss on the previous logits.

Finally, TFCL [4] is a task-free, online continual learning method that starts with an ER process and includes a task-free MAS-styled [2] regularization.

All the ER-based techniques utilize a reservoir sampling buffer, which is identical to that used in [33].

C.4 Hyper-parameters

For the hyper-parameters of FSNet and the baselines mentioned in Section C.3, we follow the setting in [33]. Besides, we cross-validate the hyper-parameters on the ETTh2 dataset and use them for the remaining ones. In particular, we use the following configuration:

- Learning rate $3e-3$ on Traffic and ECL and $1e-3$ for other datasets. Learning rate $1e-2$ for the EGD algorithm and $1e-3$ for the offline reinforcement learning block, where the selection scope is $\{1e-3, 3e-3, 1e-2, 3e-2\}$.
- Number of hidden layers 10 for both cross-time and cross-variable branches, where the selection scope is $\{6, 8, 10, 12\}$.
- Adapter's EMA coefficient 0.9, Gradient EMA for triggering the memory interaction 0.3, where the selection scope is $\{0.1, 0.2, \dots, 1.0\}$.
- Memory triggering threshold 0.75, where the selection scope is $\{0.6, 0.65, 0.7 \dots, 0.9\}$.
- Episodic memory size: 5000 (for ER, MIR, and DER++), 50 (for TFCL).

C.5 Additional Numerical Results

Additional forecasting results. In this section, we analyze the performance of different forecasting methods on four datasets, ECL, WTH, ETTh2, and ETTm1, with various starting points, as shown in Figure 8, Figure 9, Figure 10, and Figure 11, respectively. For the last three datasets, all methods produce similar results that can capture the underlying time series patterns, and the performance differences are not significant. However, when it comes to the ECL dataset, we observe that almost all baselines exhibit poor forecasting results at the onset of the concept shift (time step 2500). As we provide more instances, the performance of these methods improves, as evidenced by the cumulative loss curves in Figure 7 and Figure 6.

Abltion studies of hyper-parameters. We conduct detailed ablation studies about model layers, learning rate, and model dimension here. Taking into account the learning rate for the two-branch framework, the learning rate for the long-term weight, and the learning rate for the short-term weight: $lr, lr_{\mathbf{w}}, lr_{\mathbf{b}}$, as shown in Table 10 (left), the impact of the learning rate on dual-stream networks is quite significant. The optimal learning rate varies for each dataset, but we can see

Table 6: **Standard deviations of the metrics in Table. 2 and Table. 3.**

Method / H	MSE											
	ETTH2			ETTm1			WTH			ECL		
	1	24	48	1	24	48	1	24	48	1	24	48
Informer	1.370	2.254	2.088	0.088	0.035	0.020	0.005	0.003	0.009			
OnlineTCN	0.011	0.017	0.148	0.003	0.002	0.002	0.001	0.001	0.001	0.019	0.077	0.122
TFCL	0.030	0.005	0.279	0.004	0.006	0.010	0.002	0.001	0.004	0.047	0.338	0.253
ER	0.018	0.007	0.141	0.005	0.003	0.004	0.001	0.001	0.009	0.034	0.236	0.320
MIR	0.019	0.017	0.130	0.005	0.005	0.006	0.002	0.001	0.009	0.037	0.261	0.143
DER++	0.022	0.024	0.143	0.003	0.002	0.003	0.001	0.001	0.011	0.027	0.072	0.146
FSNet	0.018	0.014	0.128	0.003	0.002	0.003	0.001	0.001	0.001	0.021	0.096	0.105
Time-TCN	0.020	0.010	0.189	0.004	0.004	0.005	0.001	0.001	0.001	0.033	0.130	0.232
PatchTST	0.022	0.010	0.183	0.005	0.005	0.007	0.002	0.001	0.007	0.039	0.167	0.239
OneNet-TCN	0.015	0.012	0.104	0.003	0.003	0.003	0.001	0.001	0.007	0.025	0.114	0.152
OneNet	0.015	0.014	0.100	0.003	0.002	0.003	0.001	0.001	0.005	0.021	0.086	0.099

Method / H	MAE											
	ETTH2			ETTm1			WTH			ECL		
	1	24	48	1	24	48	1	24	48	1	24	48
Informer	0.043	0.102	0.091	0.060	0.023	0.014	0.005	0.003	0.008			
OnlineTCN	0.007	0.002	0.016	0.002	0.002	0.003	0.002	0.077	0.122	0.002	0.009	0.011
TFCL	0.003	0.003	0.024	0.008	0.005	0.008	0.002	0.001	0.006	0.011	0.019	0.008
ER	0.017	0.006	0.013	0.009	0.002	0.004	0.002	0.001	0.005	0.011	0.017	0.014
MIR	0.018	0.005	0.012	0.009	0.004	0.005	0.002	0.001	0.005	0.013	0.013	0.012
DER++	0.015	0.004	0.015	0.007	0.002	0.002	0.002	0.001	0.007	0.002	0.013	0.014
FSNet	0.009	0.005	0.012	0.004	0.002	0.002	0.001	0.001	0.001	0.001	0.011	0.011
Time-TCN	0.009	0.004	0.018	0.006	0.003	0.005	0.002	0.026	0.044	0.008	0.009	0.011
PatchTST	0.013	0.005	0.016	0.009	0.004	0.006	0.002	0.001	0.005	0.012	0.010	0.011
OneNet-TCN	0.017	0.005	0.013	0.008	0.003	0.004	0.002	0.001	0.006	0.009	0.009	0.013
OneNet	0.014	0.005	0.013	0.007	0.003	0.003	0.002	0.001	0.004	0.005	0.007	0.012

that for each dataset, the optimal learning rate is generally within the range of $[1e-4, 1e-2]$. $lr_{\mathbf{w}}$ has a relatively small impact on the final performance of the model. On the contrary, the offline-RL module determines whether the weights can quickly adapt to the new distribution, which has a greater impact on the final performance. In terms of model parameters, $\#$ Layers, d_m, and d_{head}, all three have a significant impact on the performance of the model. A small model may not be able to fit the training data, but a model that is too large increases the risk of overfitting, so each dataset has an optimal model size. However, in this paper, we use the same hyperparameters for all datasets to simplify the complexity of training and model selection. Specifically, we set $lr = 1e-3, lr_{\mathbf{w}} = 1e-2, lr_b = 1e-3, \#layers = 10, d_m = 64, d_{head} = 320$.

C.6 Ensembling more than two networks.

In the main paper, we verify the effectiveness of ensembling two branches with different model biases. Here, we show that the proposed OneNet framework enables us to incorporate more branches and the OCP block can fully utilize the benefit of each branch. As shown in Table 7, incorporating PatchTST to OneNet-TCN will further reduce the forecasting results during online forecasting.

C.7 More visualization results and Convergence Analysis of Different Structures

As shown in Figure 6 and Figure 7, ETTh2 and ECL datasets pose the greatest challenge to all models due to the sharp peaks in their loss curves. When the forecasting window is short, OneNet outperforms all baselines by a significant margin on all datasets. When the forecasting window is extended to $H = 48$, FSNet is comparable to OneNet in the first three datasets. However, when concept drift occurs in the ECL dataset, all baselines experience a drastic increase in their cumulative MSE, except OneNet, which maintains a low MSE. Furthermore, the initialized MSE error of OneNet is consistently lower than that of all baselines, thanks to the two-stream structure of OneNet. For

Table 7: **MSE and MAE of various adaptation methods**. H: forecast horizon. OneNet-TCN+Patch is the mixture of TCN, Time-TCN, and PatchTST.

Metric	Method	ETTH2			ETTm1			WTH			ECL			Avg
		1	24	48	1	24	48	1	24	48	1	24	48	
MSE	TCN	0.502	0.830	1.183	0.214	0.258	0.283	0.206	0.308	0.302	3.309	11.339	11.534	2.522
	Time-TCN	0.491	0.779	1.307	0.093	0.281	0.308	0.158	0.311	0.308	4.060	5.260	5.230	1.549
	PatchTST	0.362	1.622	2.716	0.083	0.427	0.553	0.162	0.372	0.465	2.022	4.325	5.030	1.512
	OneNet-TCN	0.411	0.772	0.806	0.082	0.212	0.223	0.171	0.293	0.310	2.470	4.713	4.567	1.253
	OneNet-TCN+Patch	0.355	0.844	1.120	0.079	0.239	0.255	0.163	0.298	0.314	2.172	4.142	4.149	1.178
MAE	TCN	0.436	0.547	0.589	0.085	0.381	0.403	0.276	0.367	0.362	0.635	1.196	1.235	0.543
	Time-TCN	0.425	0.544	0.636	0.211	0.395	0.421	0.204	0.378	0.378	0.332	0.420	0.438	0.399
	PatchTST	0.341	0.577	0.672	0.186	0.471	0.549	0.200	0.393	0.459	0.224	0.341	0.375	0.399
	OneNet-TCN	0.374	0.511	0.543	0.191	0.319	0.371	0.221	0.345	0.356	0.411	0.513	0.534	0.391
	OneNet-TCN+Patch	0.338	0.513	0.552	0.184	0.360	0.381	0.217	0.351	0.381	0.297	0.423	0.457	0.371

Table 8: **MSE of various adaptation methods with delayed feedback**. H: forecast horizon. OneNet-TCN is the mixture of TCN and Time-TCN, and OneNet is the mixture of FSNet and Time-FSNet.

Method / H	ETTH2			ETTm1			WTH			ECL			
	1	24	48	1	24	48	1	24	48	1	24	48	Avg
OnlineTCN	0.502	5.871	11.074	0.214	0.410	0.535	0.206	0.429	0.504	3.309	9.621	24.159	4.736
ER	0.508	5.461	17.329	0.086	0.367	0.498	0.180	0.373	0.435	2.579	8.091	17.700	4.467
DER++	0.508	5.387	17.334	0.083	0.347	0.465	0.174	0.369	0.431	2.657	7.878	17.692	4.444
FSNet	0.466	5.765	11.907	0.085	0.383	0.502	0.162	0.335	0.411	3.143	8.722	27.150	4.919
OneNet-TCN	0.411	2.639	4.995	0.082	**0.287**	0.382	0.171	0.341	0.433	2.470	**4.809**	6.252	1.939
OneNet	**0.380**	**2.064**	**4.952**	**0.082**	0.332	**0.351**	**0.156**	**0.323**	**0.394**	**2.351**	4.984	**6.226**	**1.883**

instance, in Figure 6(b) and Figure 6(f), OneNet demonstrates a significantly lower MSE than baselines when the number of instances is less than 100.

C.8 Online forecasting results with delayed feedback

As illustrated in Section 2, this paper adopts the same setting as FSNet [33], where the true values of each time step are revealed to improve the performance of the model in subsequent rounds. However, in real-world applications, the true values of the forecast horizon H may not be available until H rounds later, which is known as online forecasting with delayed feedback. This setting is more challenging because the model cannot be retrained at each round and we can only train the model per H round. Tables 8 and 9 show the cumulative performance considering MSE and MAE, respectively. As expected, all methods perform worse with delayed feedback than under the traditional online forecasting setting. Notably, the state-of-the-art method FSNet is shown to be sensitive to delayed feedback, particularly when $H = 48$, where it is even inferior to a simple TCN baseline on some datasets. In contrast, our proposed method OneNet significantly outperforms all continual learning baselines across different datasets and delayed forecast horizons.

Table 9: **MAE of various adaptation methods with delayed feedback**. H: forecast horizon. OneNet-TCN is the mixture of TCN and Time-TCN, and OneNet is the mixture of FSNet and Time-FSNet.

Method / H	ETTH2			ETTm1			WTH			ECL			
	1	24	48	1	24	48	1	24	48	1	24	48	Avg
OnlineTCN	0.436	1.109	1.348	0.085	0.511	0.548	0.276	0.459	0.508	0.635	0.783	1.076	0.648
ER	0.376	0.976	1.651	0.197	0.456	0.525	0.244	0.421	0.459	0.506	0.595	0.772	0.598
DER++	0.375	0.967	1.644	0.192	0.443	0.508	0.235	0.415	0.456	0.421	0.591	0.758	0.584
FSNet	0.368	0.983	1.494	0.191	0.468	0.502	0.216	0.394	0.453	0.472	0.827	1.391	0.554
OneNet-TCN	0.374	0.772	0.951	0.191	**0.387**	**0.417**	0.221	0.389	0.461	0.411	**0.381**	0.451	0.451
OneNet	**0.348**	**0.684**	**0.916**	**0.187**	0.428	0.430	**0.201**	**0.381**	**0.436**	**0.254**	0.387	**0.444**	**0.425**

Table 10: **Results of different OneNet 's hyper-parameter configurations** on the benchmarks ($H = 48$). $lr, lr_{\mathbf{w}}, lr_{\mathbf{b}}$ are the learning rate for the two-branch framework, the learning rate for the long-term weight, and the learning rate for the short-term weight. # Layers is the number of layers of the two branches of OneNet. d_m, d_{head} is the hidden dimension and the output dimension of the encoders, respectively.

Hyper-Parameter	Value	MSE				Hyper-Parameter	Value	MSE			
		ETTh2	ETTm1	WTH	ECL			ETTh2	ETTm1	WTH	ECL
lr	1.00E-01	-	-	-	-	# Layers	6	0.632	0.114	0.203	2.402
	1.00E-02	0.585	0.152	0.171	3.128		8	0.661	0.101	0.201	2.289
	1.00E-03	0.656	0.111	0.196	2.516		10	0.609	0.108	0.200	2.201
	1.00E-04	2.994	0.464	0.331	4.949		12	0.652	0.115	0.200	2.328
$lr_{\mathbf{w}}$	1.00E-01	0.619	0.108	0.202	2.177	d_m	16	0.679	0.122	0.223	2.201
	1.00E-02	0.609	0.108	0.205	2.184		32	0.612	0.116	0.210	2.810
	1.00E-03	0.608	0.108	0.201	2.197		64	0.609	0.108	0.200	2.311
	1.00E-04	0.607	0.108	0.201	2.197		160	0.619	0.108	0.200	2.141
$lr_{\mathbf{b}}$	1.00E-01	0.899	0.134	0.221	2.499	d_{head}	80	0.741	0.136	0.219	2.468
	1.00E-02	0.876	0.112	0.197	2.372		160	0.600	0.112	0.214	2.364
	1.00E-03	0.656	0.111	0.196	2.371		320	0.609	0.108	0.201	2.184
	1.00E-04	0.643	0.111	0.196	2.362		500	0.571	0.104	0.182	2.182

Table 11: **Ablation studies of the variable independence and frequency domain augmentation**, where the metric is MSE. FEDformer-F uses frequency-enhanced blocks with Fourier transform, and FEDformer-W uses frequency-enhanced blocks with Wavelet transform. Time-TCN is the variable independence version of TCN.

Method	Online	ETTH2			ETTm1			WTH			ECL			Avg
		1	24	48	1	24	48	1	24	48	1	24	48	
FEDformer-F	✗	1.922	3.045	4.016	0.922	1.003	1.821	3.544	2.344	1.179	43.852	37.802	37.377	11.569
	✓	1.912	3.013	3.951	0.372	0.633	0.586	2.196	0.376	0.562	39.243	35.975	36.092	10.409
FEDformer-W	✗	1.816	3.070	3.996	2.275	3.784	2.662	1.220	1.211	1.431	41.791	37.236	37.210	11.475
	✓	1.798	2.993	1.623	0.235	0.451	0.516	0.717	0.962	0.372	21.387	24.600	27.640	6.941
TCN	✗	27.060	27.760	26.320	2.240	12.170	10.880	0.290	0.480	0.580	538.000	546.000	552.000	145.315
	✓	0.530	0.930	0.910	0.130	0.310	0.250	0.300	0.348	0.348	3.010	11.680	10.800	2.462
Time-TCN	✗	4.530	7.840	1.300	0.097	0.800	1.030	0.162	0.344	0.429	47.900	48.660	67.150	15.020
	✓	0.480	0.780	1.300	0.090	0.280	0.310	0.300	0.310	0.309	4.010	5.220	5.210	1.550

C.9 The effect of variable independence and frequency domain augmentation

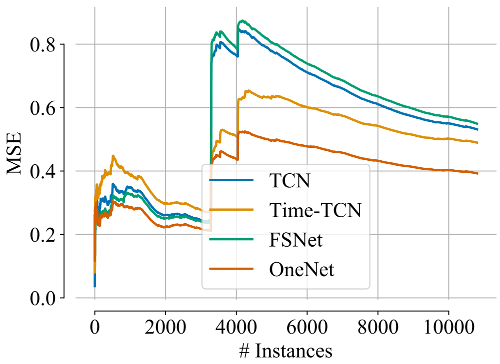
As shown in Table 11, we observe that frequency-enhanced blocks, which use the wavelet transform, offer greater robustness to the Fourier transform. FEDformer outperforms TCN in terms of generalization, but online adaptation has a limited impact on performance, similar to other transformer-based models. Notably, we find that variable independence is crucial for model robustness. By convolving solely on the time dimension, independent of the feature channel, we significantly reduce MSE error compared to convolving on the feature channel, regardless of whether online adaptation is applied.

C.10 Comparison of existing forecasting structures.

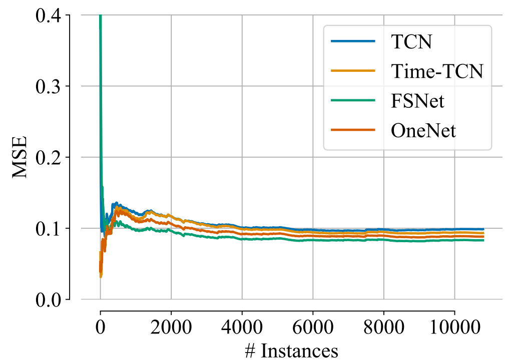
Results are shown in Table 12. Considering the average MSE on all four datasets, all transformer-based models and Dlinear are better than TCN and Time-TCN. However, with online adaptation, the forecasting error of TCN structures is reduced by a large margin and is better than DLinear and FEDformer. Specifically, we show that the current transformer-based model (PatchTST [32]) demonstrates superior generalization performance than the TCN models **even without any online adaptation**, particularly in the challenging ECL task. However, we also noticed that PatchTST remains largely unchanged after online retraining. In contrast, the TCN structure can quickly adapt to the shifted distribution, and the online updated TCN model prefers a better forecasting error than the adapted PatchTST on the first three data sets. Therefore, it is promising to combine the strengths of both structures to create a more robust and adaptable model that can handle shifting data distributions better.

Table 12: **Comparison of existing forecasting structures**, including TCN [6], FEDformer [49], PatchTST [32], Dlinear [45], Nlinear [45], TS-Mixer [13], and CrossFormer [47].

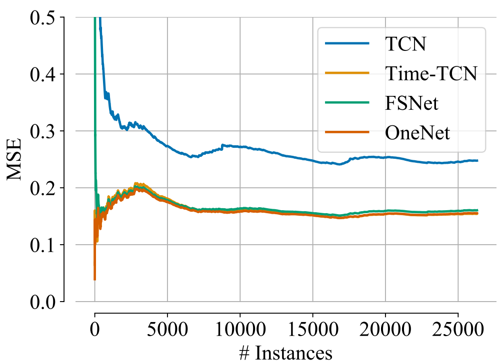
Method	Online	ETTH2			ETTm1			WTH			ECL			Avg
		1	24	48	1	24	48	1	24	48	1	24	48	
FEDformer-F	✗	1.922	3.045	4.016	0.922	1.003	1.821	3.544	2.344	1.179	43.852	37.802	37.377	11.569
	✓	1.912	3.013	3.951	0.372	0.633	0.586	2.196	0.376	0.562	39.243	35.975	36.092	10.409
FEDformer-W	✗	1.816	3.070	3.996	2.275	3.784	2.662	1.220	1.211	1.431	41.791	37.236	37.210	11.475
	✓	1.798	2.993	1.623	0.235	0.451	0.516	0.717	0.962	0.372	21.387	24.600	27.640	6.941
PatchTST	✗	0.427	2.090	3.290	0.083	0.433	0.570	0.163	0.375	0.467	2.030	4.395	5.101	1.619
	✓	0.362	1.622	2.716	0.083	0.427	0.553	0.162	0.372	0.465	2.022	4.325	5.030	1.512
Crossformer	✗	23.270	28.904	29.218	0.400	1.433	1.691	0.146	0.327	0.426	469.260	475.490	478.270	125.736
	✓	9.873	2.856	5.772	0.096	0.356	0.370	0.149	0.317	0.359	68.300	92.500	94.790	22.978
TCN	✗	27.060	27.760	26.320	2.240	12.170	10.880	0.290	0.480	0.580	538.000	546.000	552.000	145.315
	✓	0.530	0.930	0.910	0.130	0.310	0.250	0.300	0.348	0.348	3.010	11.680	10.800	2.462
Time-TCN	✗	4.530	7.840	1.300	0.097	0.800	1.030	0.162	0.344	0.429	47.900	48.660	67.150	15.020
	✓	0.480	0.780	1.300	0.090	0.280	0.310	0.300	0.310	0.309	4.010	5.220	5.210	1.550
DLinear	✗	2.91	10.25	7.53	0.538	1.461	1.233	0.266	0.462	0.542	12.03	51.28	58.46	12.247
	✓	2.44	9.24	6.91	0.46	1.3	1.12	0.262	0.459	0.541	6.69	27.82	31.54	7.399
NLinear	✗	0.424	50.15	49.52	0.09	4.02	4.13	0.171	1.07	1.08	2.14	930	929	164.316
	✓	0.369	50.24	49.6	0.089	4.035	4.141	0.171	1.053	1.064	2.135	930	930	164.408
TS-Mixer	✗	1.968	3.525	4.88	0.335	0.726	0.855	0.255	0.429	0.503	11.16	30.93	44.68	8.354
	✓	0.78	2.05	3.060	0.219	0.550	0.660	0.237	0.413	0.482	2.798	4.983	5.764	1.833



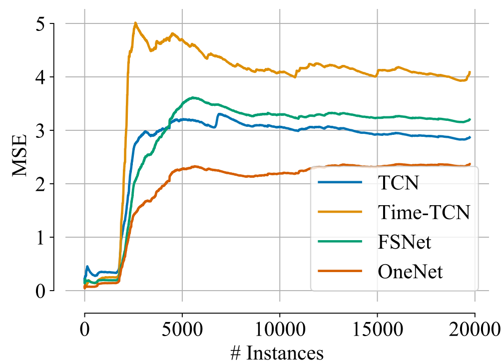
(a) ETTh2.



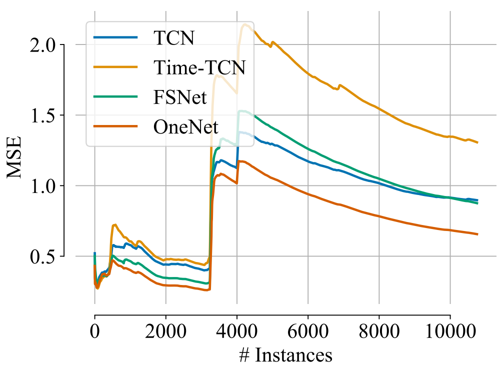
(b) ETTm1.



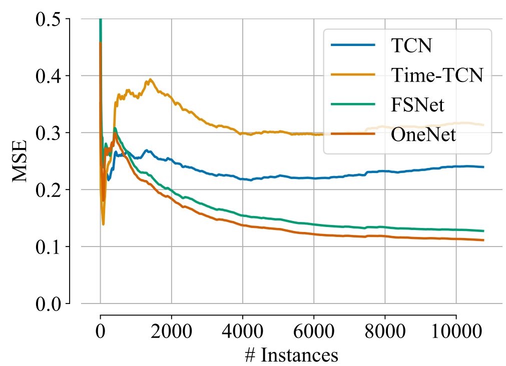
(c) WTH.



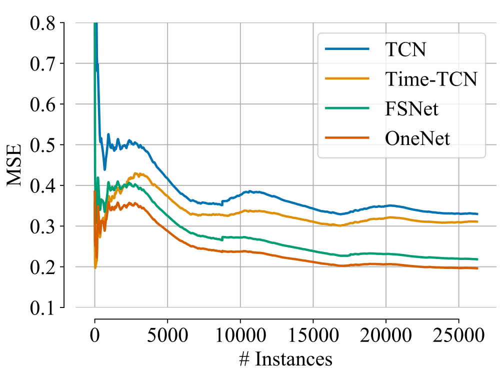
(d) ECL



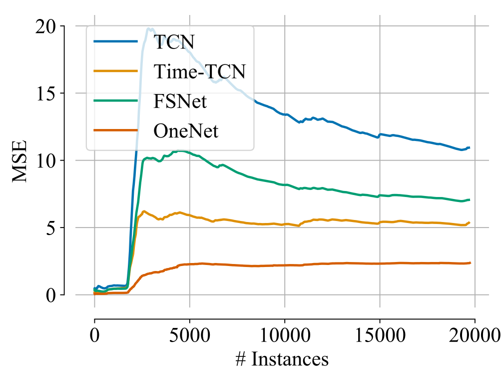
(e) ETTh2.



(f) ETTm1.

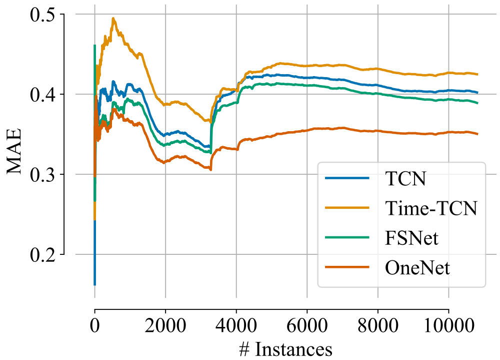


(g) WTH.

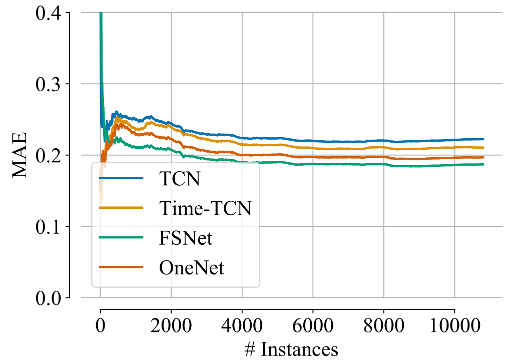


(h) ECL

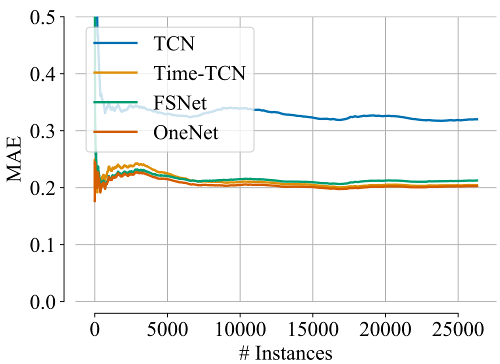
Figure 6: **Evolution of the cumulative MSE loss** during training with forecast window $H = 1$ (a, b, c, d) and $H = 48$ (e, f, g, h).



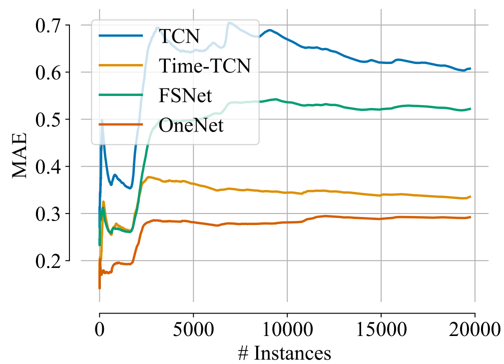
(a) ETTh2.



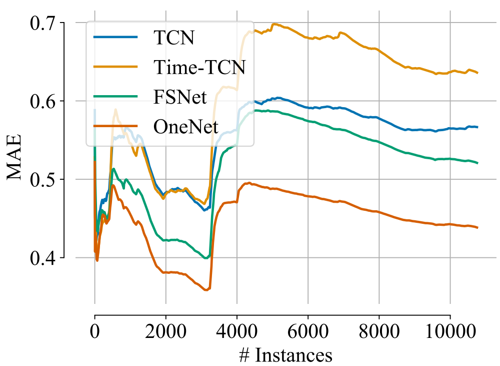
(b) ETTm1.



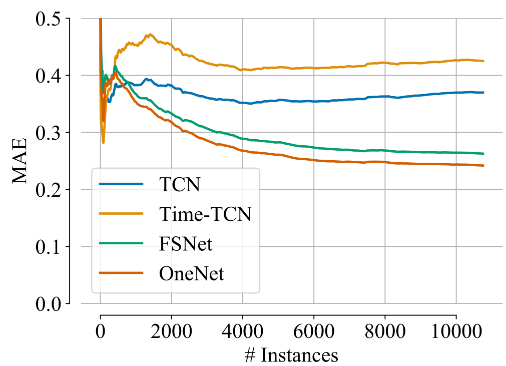
(c) WTH.



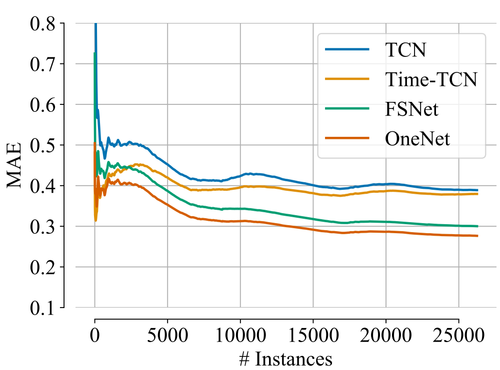
(d) ECL



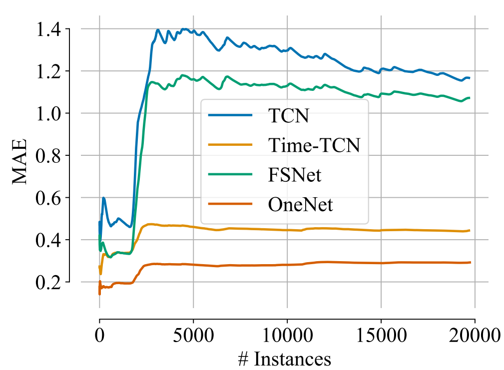
(e) ETTh2.



(f) ETTm1.

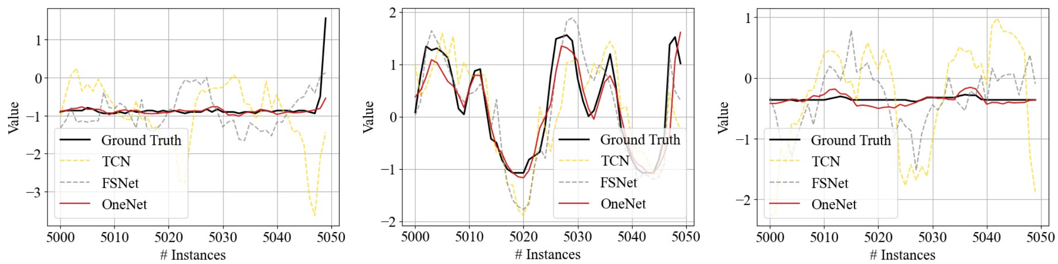


(g) WTH.

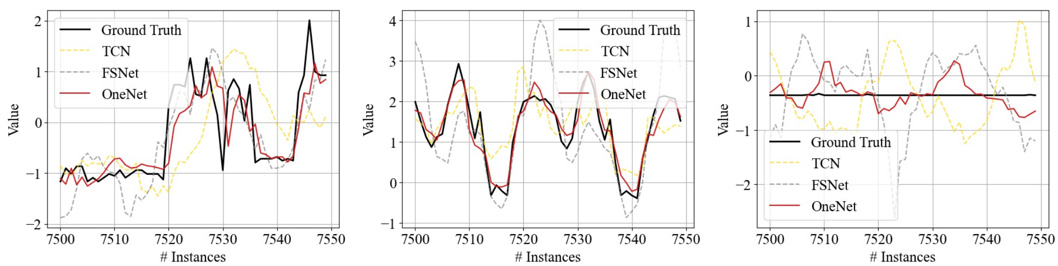


(h) ECL

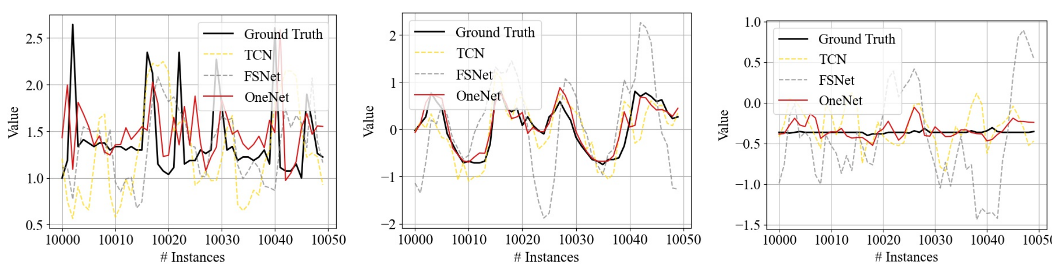
Figure 7: Evolution of the cumulative MAE loss during training with forecasting window $H = 1$ (a,b,c,d) and $H = 48$ (e,f,g,h).



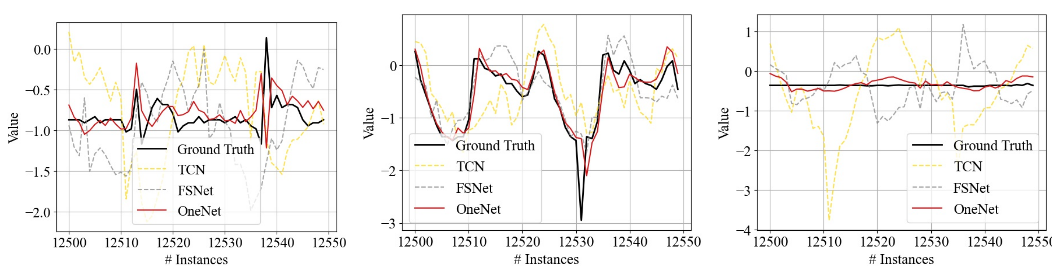
(a) Channel 1. (b) Channel 2. (c) Channel 3.



(d) Channel 1. (e) Channel 2. (f) Channel 3.

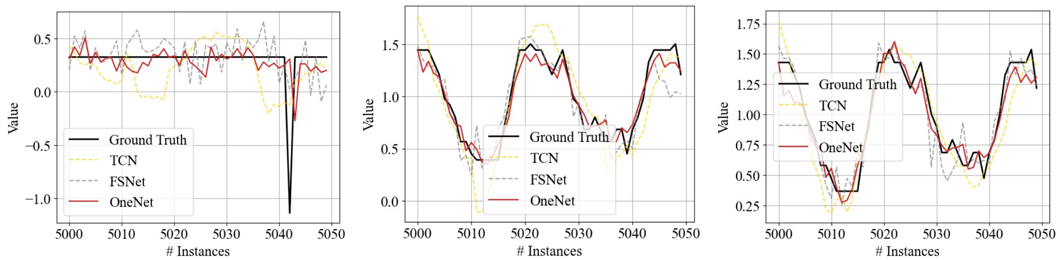


(g) Channel 1. (h) Channel 2. (i) Channel 3.

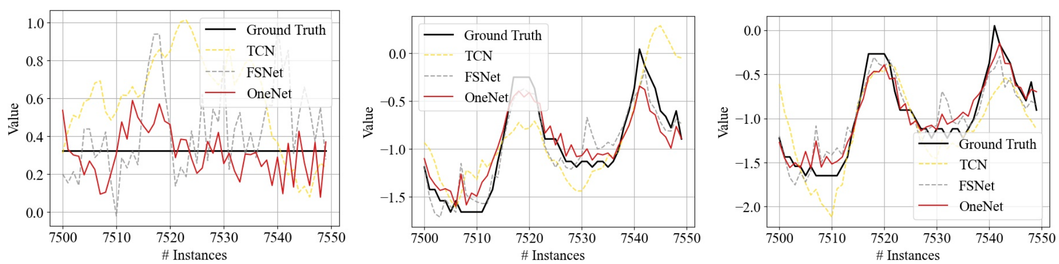


(j) Channel 1. (k) Channel 2. (l) Channel 3.

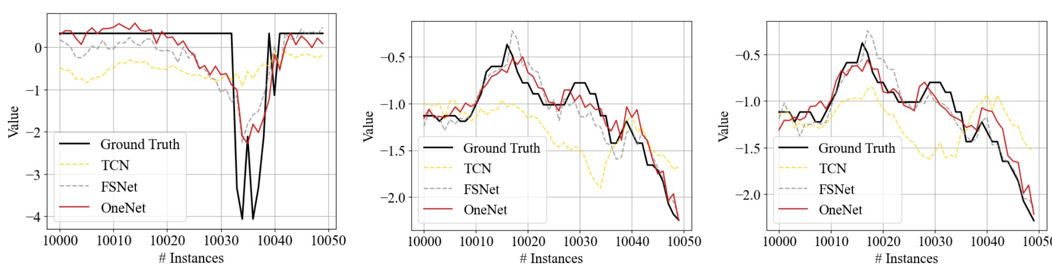
Figure 8: **Visualization of the model's prediction throughout the online learning process** in the ECL dataset. We focus on a short horizon of 50 time steps and the start prediction time is from 5000 (a,b,c), 7500 (d,e,f), 10000 (g,h,i), and 12500 (j,k,l) respectively.



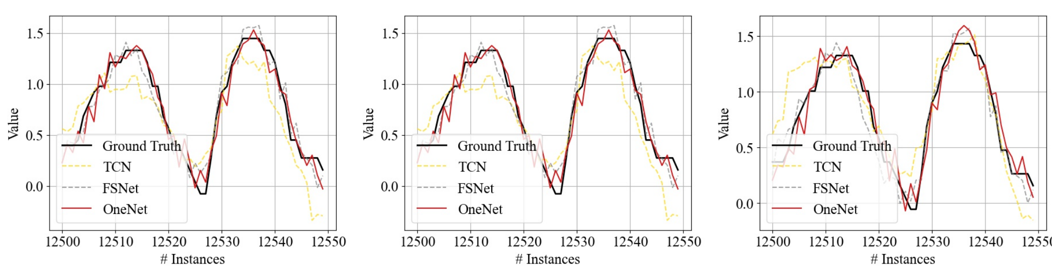
(a) Channel 1. (b) Channel 2. (c) Channel 3.



(d) Channel 1. (e) Channel 2. (f) Channel 3.

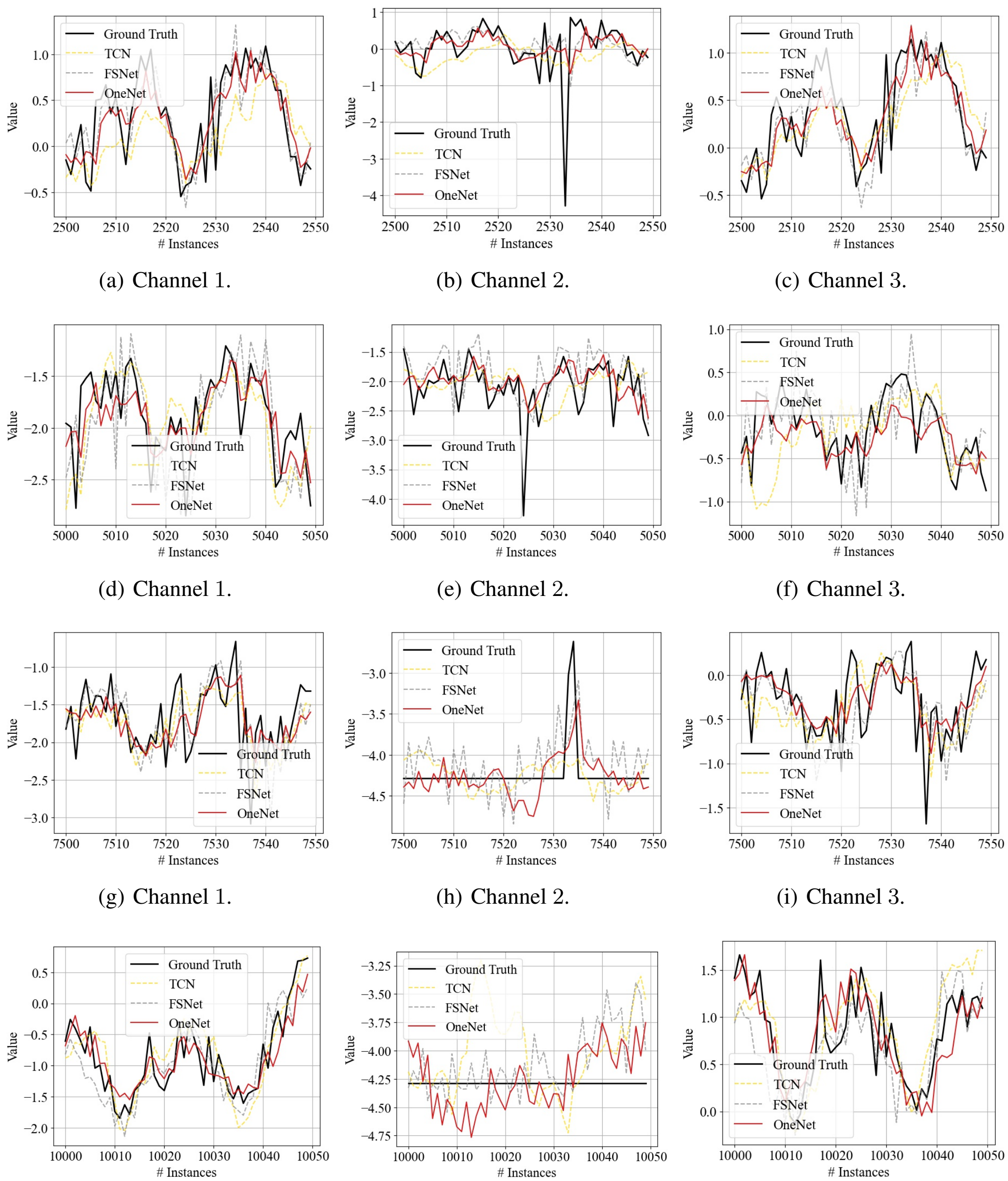


(g) Channel 1. (h) Channel 2. (i) Channel 3.



(j) Channel 1. (k) Channel 2. (l) Channel 3.

Figure 9: **Visualization of the model's prediction throughout the online learning process** on the WTH dataset. We focus on a short horizon of 50 time steps and the start prediction time is from 5000 (a,b,c), 7500 (d,e,f), 10000 (g,h,I), and 12500 (j,k,l), respectively.



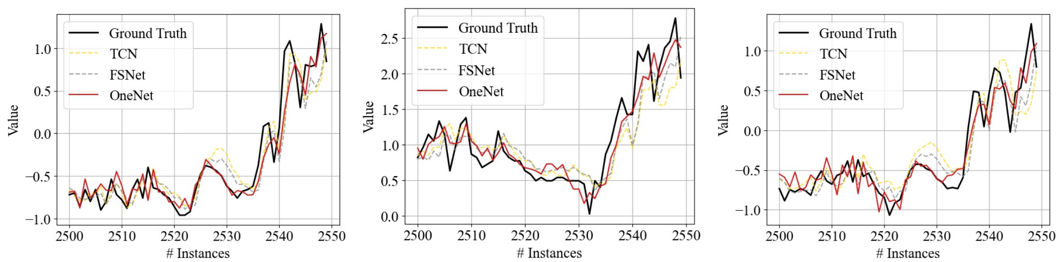
(a) Channel 1. (b) Channel 2. (c) Channel 3.

(d) Channel 1. (e) Channel 2. (f) Channel 3.

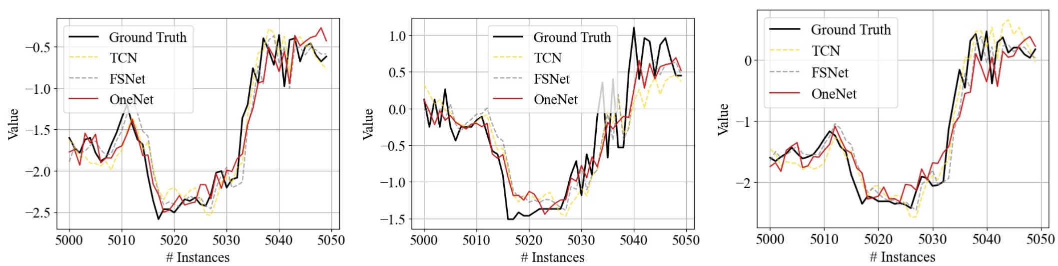
(g) Channel 1. (h) Channel 2. (i) Channel 3.

(j) Channel 1. (k) Channel 2. (l) Channel 3.

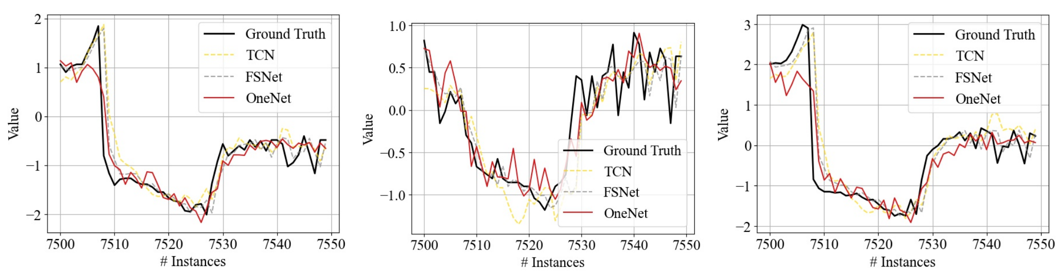
Figure 10: **Visualization of the model's prediction throughout the online learning process** in the ETTh2 data set. We focus on a short horizon of 50 time steps and the start prediction time is from 2500 (a,b,c), 5000 (d,e,f), 7500 (g,h,i), and 10000 (j,k,l) respectively.



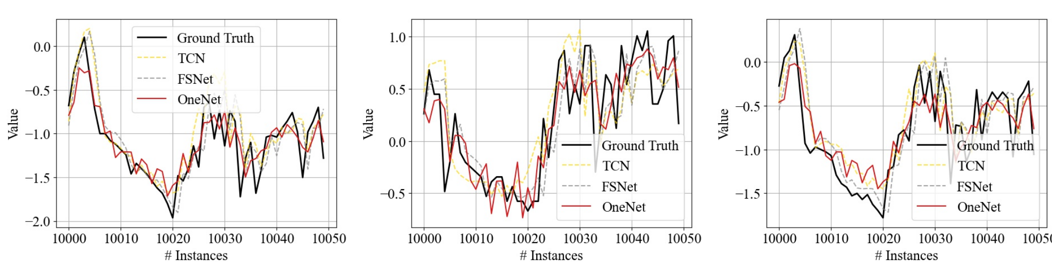
(a) Channel 1. (b) Channel 2. (c) Channel 3.



(d) Channel 1. (e) Channel 2. (f) Channel 3.



(g) Channel 1. (h) Channel 2. (i) Channel 3.



(j) Channel 1. (k) Channel 2. (l) Channel 3.

Figure 11: **Visualization of the model's prediction throughout the online learning process** on the ETTm1 dataset. We focus on a short horizon of 50 time steps and the start prediction time is from 2500 (a, b, c), 5000 (d, e, f), 7500 (g, h, i) and 10000 (j, k, l), respectively.