# DMCNET: MULTI-SCALE TREND AND PATCH COORDINATION FOR ONLINE TIME SERIES FORECASTING UNDER CONCEPT DRIFT

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

## ABSTRACT

Online time series forecasting must adapt to evolving data distributions while limiting the propagation of errors across sequential updates. This paper presents DMCnet, a multi-scale coordination framework for online multivariate time series forecasting under concept drift. DMCnet employs two complementary predictors: a trend-oriented branch that extracts multi-scale temporal patterns using smoothed moving-average representations and dilated temporal convolutions, and a patch-oriented branch that models local dynamics from segmented input patches. The forecasts are combined online through convex weight integration, allowing the model to adjust the relative contributions of long-term and local predictors as the stream evolves. To reduce the effect of unstable short-term updates, we further introduce the Online Noise Sensitivity Protector (ONSP), which uses the discrepancy between the incoming sequence and its smoothed trend as a data-dependent signal to regulate short-term weight updates. The resulting mechanism is designed as a practical stabilization strategy rather than as a direct statistical estimator of observation noise. We evaluate DMCnet on five real-world datasets, including the newly released PCpower dataset, which records personal-computer power consumption under diverse usage scenarios. Across the evaluated datasets and forecasting settings, DMCnet reduces the average cumulative mean squared error (MSE) and mean absolute error (MAE) by 24.67% and 19.32%, respectively, compared with the strongest competing methods. These results indicate that coordinating multi-scale trend representations with local patch representations can improve online forecasting robustness under distributional change.

## 1 INTRODUCTION

The rapid development of deep models has significantly enhanced the capability of time series forecasting, particularly in capturing complex temporal patterns and handling high-throughput data Benidis et al. (2022). With the increasing demand for timely decision-making in applications such as finance Cheng et al. (2022) and industrial production Wen et al. (2022), there is growing attention on the development of online forecasting algorithms. Unlike the offline batch training paradigm, where the entire training dataset is assumed to be available in advance, online forecasting involves sequentially feeding training samples into the model as streaming data over time Aljundi et al. (2019a). Consequently, data distributions and dependencies may change over time, a phenomenon known as concept drift Tsymbal (2004). While tuning the model with all available samples during each training session seems like a straightforward solution, it is time-consuming and cannot meet the timeliness requirements of online forecasting. To this end, a suitable approach is to incrementally update deep models employing new samples to capture the evolving temporal dynamics in the variables Woo et al. (2022); Pham et al. (2022); Wen et al. (2023); Zhang et al. (2024); Zhao & Shen (2024).

Despite the widespread need for online forecasting in real-world scenarios, training effective deep models remains challenging. On the one hand, the inherent dynamics of time series signals can cause dependencies between time steps to change over time. This requires online deep models to adjust promptly based on new samples to adapt to evolving temporal dependencies, thus mitigating concept drift. On the other hand, error accumulation stems from the inherent randomness or gradient noise in streaming data, and the resulting errors can continuously affect subsequent forecasting Chen et al.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

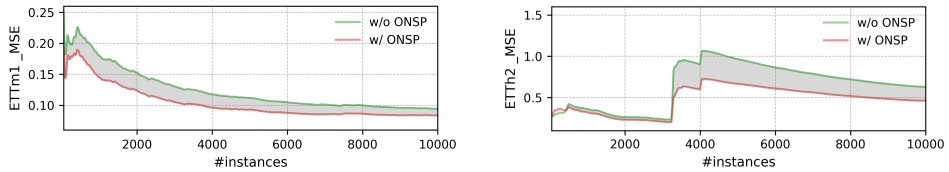



Figure 1: Examples reflecting error accumulation, where the online noise sensitivity protector (ONSP) is the module proposed in this paper to mitigate the impact of error accumulation. The comparison results of MSE show that mitigating the impact of error accumulation effectively reduces the forecasting error.

(2023); Zhang et al. (2024); Zeng et al. (2023). Figure 1 illustrates two real-world examples reflecting error accumulation, where the **O**nline **N**oise **S**ensitivity **P**rotector (ONSP) module is proposed in this paper to mitigate the impact of error accumulation. w/ and w/o mean with and without related components, respectively. The example on the ETTH2 dataset shows a dramatic case where the mean squared error (MSE) sharply increases between instances 2500-4000, affecting subsequent predictions. For the ETTm1 dataset, although the MSE steadily decreases, using the ONSP module still further reduces the prediction error, indicating that error accumulation persists even in more stable cases.

Recent efforts in the literature have focused on designing more efficient deep models for incrementally updating online time series forecasting models. FSNet Pham et al. (2022) dynamically balances the adaptation to new samples and the retention of similar old knowledge to incrementally update the model based on TCN backbones Bai et al. (2018). However, the less competitive model structure limits the capability of online models in representation learning and online updates. To this end, OneNet Wen et al. (2023) examines more powerful feature extractors, such as the Transformer Waswani et al. (2017); Nie et al. (2022); Zhou et al. (2022), to separately model temporal pattern dynamics and inter-variable dependencies, which perform better than vanilla TCN or FSNet. Additionally, ensembling FSNet and Time-TCN achieves the current most competitive baseline model. OneNet also integrates reinforcement learning-based methods into the traditional online convex optimization framework, allowing the two models to be linearly combined through dynamic weight adjustments. However, OneNet neglects the prompt updating of trends, limiting the detection of significant distribution shifts in the stream data. Furthermore, error accumulation prevents effective correction of persistent prediction errors caused by model biases. Although Zhang et al. (2024) introduces Gaussian noise into training instances and mitigates the noise gradients associated with streaming data through data augmentation, there are still challenges to addressing these limitations from the perspective of model architectures.

To address these challenges, we propose a **D**ynamic **M**ulti-scale **C**oordination **net**work (DMCnet) for online multivariate time series forecasting under concept drift. DMCnet combines a trend-oriented predictor that captures multi-scale temporal patterns with a patch-oriented predictor that models local dynamics. Their forecasts are integrated through online adaptive weighting, allowing their relative contributions to change as the data stream evolves. We further analyze how noise-induced perturbations can accumulate in short-term weight updates under simplifying assumptions, motivating additional regulation of these updates. To this end, we introduce the **O**nline **N**oise **S**ensitivity **P**rotector (ONSP), which uses the discrepancy between the incoming sequence and its smoothed trend as a data-dependent signal to modulate short-term weight offsets. ONSP serves as a practical stabilization mechanism rather than a direct statistical estimator of observation noise. We also introduce PCpower, a dataset of personal-computer power consumption under diverse usage scenarios, to support the evaluation of online forecasting in realistic computing workloads. Our contributions are summarized as follows:

- We propose DMCnet, an online forecasting framework that adaptively combines multi-scale trend forecasts and local patch-based forecasts to accommodate evolving temporal patterns.

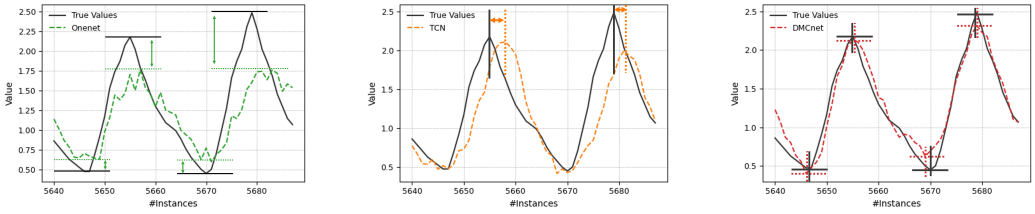



Figure 2: Forecasting examples on ETTH2 dataset. (a) Example of energy misalignment on OneNet where predictions follow the overall trend of true values but perform energy misalignment (*e.g.*, Instance: 5640-5650, 5650-5660, 5675-5680). (b) Example of rhythm misalignment on TCN, which suffers energy misalignment (*e.g.*, Instance: 5650-5660, 5675-5685). (c) Example of improvement on DMCnet where the rhythm and energy of the predicted values are aligned with the true values.

- We analyze the accumulation of noise-induced perturbations in short-term weight updates under simplifying assumptions and introduce ONSP, a discrepancy-based mechanism designed to stabilize short-term weight adaptation.
- We introduce PCpower, a dataset containing component-level power consumption measurements across diverse laptop configurations and usage scenarios, providing a benchmark for power consumption forecasting.
- Experiments on five real-world datasets, including PCpower, show that DMCnet achieves lower MSE and MAE than the evaluated baselines across the tested forecasting horizons.

## 2 Related Work

### 2.1 Concept Drift

In the real world, due to the dynamic nature of environments, the underlying concepts of observed values may change over time. For example, social interests may shift due to changes in social relationships Bai et al. (2022), and stock investment behaviors can be influenced by market sentiment Li et al. (2022). This phenomenon is known as concept drift Tsymbal (2004), which increases the complexity of model learning for the distribution of test data different from the historical training data. In this paper, we focus on online time series forecasting. Unlike offline batch training paradigms that assume the entire training dataset is available in advance, online forecasting involves training samples being incrementally fed into the model as streaming data over time Aljundi et al. (2019a). Current research focuses on designing more effective deep models or frameworks to mitigate the impact of concept drift Woo et al. (2022); Pham et al. (2022); Wen et al. (2023); Zhao & Shen (2024). A representative example is OneNet Wen et al. (2023), which separately models temporal pattern dynamics and inter-variable dependencies. However, insufficient attention to trend modeling limits the ability to promptly perceive streaming data and adjust model preferences. To this end, we advocate for strengthening the perception of global rhythm and local energy in online data through multi-scale decoupling and segmented response strategies to address concept drift.

### 2.2 Error Accumulation

In online learning, the knowledge acquired from previous steps can have a lasting impact on model updates. A similar concept is often discussed in recursive forecasting, where multi-step forecasting is made by recursively applying single-step forecasting. The forecasting error at each step accumulates, influencing subsequent predictions Chen et al. (2023). Similarly, when the model is updated based on the observation of a single new sample, the model bias can continuously affect the subsequent learning and updates Zhang et al. (2024); Zeng et al. (2023). Furthermore, streaming data often exhibits significant randomness or gradient noise, leading to error accumulation over time. Although Zhang et al. (2024) mitigates noise gradients in streaming data by introducing Gaussian noise into the training instances for data augmentation, addressing error accumulation from the perspective of the

model architecture remains a challenge. In this paper, we propose the ONSP module that quantifies the noise level in real-time and integrates this information into the dynamic adjustment process of the online convex optimization framework, effectively mitigating the impact of error accumulation.

### 2.3 Deep Models for Online Time Series Forecasting

Deep models for online time series forecasting require both expressive temporal representations and mechanisms for sequential adaptation. FSNet combines fast adaptation with associative memory to capture new and recurring patterns Pham et al. (2022), while OneNet dynamically combines predictors of temporal and cross-variable dependencies **?**. Meanwhile, forecasting architectures provide useful representation strategies: PatchTST encodes local subseries as patches Nie et al. (2022), and TimeMixer combines temporal information across multiple scales Wang et al. (2024). These representation strategies motivate their use in online forecasting, where their contributions must adapt as the data distribution evolves. Building on these ideas, DMCnet separately models multi-scale trends and local patch dynamics and adaptively combines their forecasts. It further introduces ONSP to regulate short-term weight offsets using the discrepancy between incoming observations and their smoothed trend, providing a practical mechanism for stabilizing online adaptation.

## 3 Notations and Preliminaries

**Online Time Series Forecasting.** For traditional offline time series forecasting, given a collection of time series $X = \{x_1, x_2, \ldots, x_L\}$ with a look-back window $L$, where $x_t \in \mathbb{R}^d$ represents the vectors collected by one (e.g., $d = 1$) or multiple sensors (e.g., $d > 1$) at time step $t$. The target is training a deep neural network $f$ to forecast future vectors $Y = \{y_{L+1}, y_{L+2}, \ldots, y_{L+P}\}$ with horizon $P$, where $y_{t'} \in \mathbb{R}^m$ denotes the predicted value at time step $t'$, and $m$ is the dimension of the forecasting features. Note that $d$ and $m$ may be different, such as when predicting a univariate time series based on a multivariate time series. In online time series forecasting, the data is streamed. The iterative process is as follows: $f$ receives a look-back window $L$ and updates the parameters of $f$, then predicts the forecast window $P$. The true values corresponding to the forecast window are then assigned to the look-back window for $f$ updating.

**Concept Drift and Error Accumulation.** In the real world, due to the dynamic nature of environments, the underlying concepts of observed values may change over time. This phenomenon is known as concept drift Tsymbal (2004), which increases the complexity of model learning for the distribution of test data different from the historical training data. The formal definition of concept drift can be described as the change over time of the joint distribution $P(X, Y)$ of input time series $X$ and forecasting values $Y$, like $P_t(X_t, Y_t) \neq P_{t+1}(X_{t+1}, Y_{t+1})$. Furthermore, in online learning, the knowledge acquired from previous steps can have a lasting impact on $f$ updates. To this end, when the model is updated based on the observation of a single new sample, the model bias can continuously affect the subsequent learning Zhang et al. (2024); Zeng et al. (2023). Streaming data often exhibits significant randomness or gradient noise, which may exacerbate error accumulation over time.

## 4 Method

In this section, we first introduce *Dynamic Trend Coordination for Concept Drift* (DTC), which combines multi-scale trend modeling and local patch encoding through adaptive forecast integration. We then analyze the accumulation of noise-induced perturbations in short-term weight updates under simplifying assumptions and present the *Online Noise Sensitivity Protector* (ONSP). ONSP uses the discrepancy between the incoming sequence and its smoothed trend as a data-dependent signal to regulate short-term weight offsets, providing a practical stabilization mechanism for online adaptation. Finally, we describe the overall architecture and optimization procedure of DMCnet.

### 4.1 DTC: Dynamic Trend coordination for Concept drift

**Temporal and Amplitude Misalignment.** Online forecasting requires capturing both broader temporal patterns and local variations as the data stream evolves. Figure 2(a) illustrates an example on ETTH2 where OneNet follows the overall trend but exhibits amplitude discrepancies around the

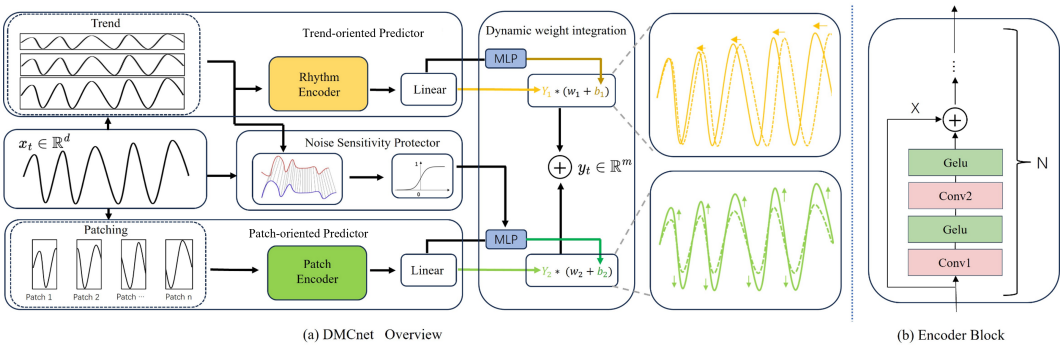



Figure 3: Overview of DMCnet including the core modules as *Trend-oriented Predictor*, *Patch-oriented Predictor*, *Online Noise Sensitivity Protector*, and *Dynamic Weight Integration*.

valley (instances 5640–5650) and peaks (instances 5650–5660 and 5675–5680). Figure 2(b) shows temporal misalignment in TCN predictions, with local changes lagging behind the observations at instances 5650–5660 and 5675–5685. These examples motivate complementary representations of multi-scale temporal patterns and local dynamics. Accordingly, we introduce a *Trend-oriented Predictor* and a *Patch-oriented Predictor*, whose forecasts are adaptively combined to adjust their relative contributions as the stream evolves.

**Trend-oriented Predictor.** Online time series exhibit temporal patterns at multiple scales, whose relative importance may change as the data stream evolves. To capture these patterns within the look-back window, we construct a trend-oriented representation using moving-average pooling with kernel sizes $[k_1, k_2, \ldots, k_H]$. The smoothed sequences are combined using softmax-normalized weights:

$$X_r = \mathrm{Softmax}\left(\mathrm{AvgP}\left(X\right)_{k_1}, \mathrm{AvgP}\left(X\right)_{k_2}, \ldots, \mathrm{AvgP}\left(X\right)_{k_H}\right), \tag{1}$$

where $\mathrm{AvgP}\left(X\right)_{k_i}$ denotes the moving average pooling with kernel scale $k_i$, and $H$ is the total number of kernels. The pooling operations preserve the temporal length, yielding $X_r \in \mathbb{R}^{L\times d}$. This weighted representation combines temporal patterns captured at different smoothing scales. Inspired by TCN, we stack $N$ layers of 1D convolutions interspersed with Gelu activation functions Hendrycks & Gimpel (2016), and finally output the result with residual connections to prevent overfitting. After passing through a linear layer, the initial prediction $Y_1 \in \mathbb{R}^{P\times m}$ is then fed into the online convex optimization framework *Dynamic Weight Integration*, which will be introduced in subsequent sections.

**Patch-oriented Predictor.** To capture local dynamics complementary to the smoothed multi-scale representation, we segment the original input sequence into $n$ temporal patches:

$$X_e = (X_1, X_2, \ldots, X_n). \tag{2}$$

The patch encoder processes $X_e$ with channel independence to model local temporal patterns and amplitude variations. A linear projection maps the encoded representation to a forecast $Y_2 \in \mathbb{R}^{P\times m}$. This forecast is combined with the trend-oriented forecast through *Dynamic Weight Integration*, allowing their relative contributions to adapt as the data stream evolves.

**Dynamic weight integration.** To integrate the above components, we integrate the Trend-oriented Predictor and Patch-oriented Predictor into a convex optimization combination. In the context of online learning for time series forecasting, the Exponential Gradient Descent (EGD) Hill & Williamson (2001) is employed to dynamically adjust the weights of the models. Specifically, the weight $w_{t,i}$ of the $i$-th expert at time $t$ is updated as:

$$w_{t+1,i} = \frac{w_{t,i}\exp\left(-\eta\left\|f_i(\mathbf{x}) - \mathbf{y}\right\|^2\right)}{S_t}, \tag{3}$$

where $\eta$ is the learning rate, $f_i(\mathbf{x})$ is the prediction made by the $i$-th expert, $\mathbf{y}$ is the true label, and $S_t$ is the normalization factor ensuring that the weights sum to 1. The term $\ell_{t,i} = \left\|f_i(\mathbf{x}) - \mathbf{y}\right\|^2$ represents the loss incurred by the $i$-th expert at time $t$.

Although EGD demonstrates stability in long-term forecasting, meaning that the upper bound of the external regret for the exponentially weighted average forecast can be maintained within a small range Wen et al. (2023). This ensures that the results of convex optimization combinations do not significantly deteriorate compared to the optimal forecast. However, the response of EGD is relatively slow in short-term forecasts, i.e., the *slow switching phenomenon* means that the change in $w_{t+1,i}$ may not be significant after a limited number of iterations Wen et al. (2023). This limits the timely response to rapidly changing time patterns. To address this issue, we assume that the long-term weighted prediction combination represents the current state, using a parameterized policy $\pi_{\theta_{rl}}\left(\mathbf{b}_t \mid \left\{\{w_{t,i}\tilde{y}_i\}_{i=1}^{d}\right\}_{t\in I}; Y_i\right)$ as the basis for agent action selection, like reinforcement learning. An efficient implementation scheme of short-term weight is:

$$b_t = f_{rl}\left(w_{t,1}Y_1 \otimes \cdots \otimes w_{t,d}Y_d \otimes Y\right), \tag{4}$$

where $f_{rl}$ is a two-layer MLPs, followed as RvS Emmons et al. (2021) and OneNet Wen et al. (2023). Another challenge is that error accumulation may cumulatively affect the update of short-term weights $b_t$. We propose the ONSP module to address this issue, as detailed in the following section.

### 4.2 ONSP: Online Noise Sensitivity Protector

The learning of short-term weights $b_t$ in the online integrated prediction model depends on the real-time supervision signals, which makes $b_t$ sensitive to noise and causes errors to accumulate over time.

**Proposition 1.** *The short-term weight $b_t$ is sensitive to model gradients, and let $G_t := \|\nabla_{b_t}\hat{y}_t\left(b_t\right)\|_F^2$ be the Frobenius norm, so the covariance of $\delta b_T$ accumulates over time, as follows:*

$$\mathrm{Var}\left[\delta b_T\right] = 4\eta_b^2\sigma^2\sum_{t=1}^{T-1} G_t, \tag{5}$$

*where $\eta_b$ denotes the learning rate of $b_t$, and $\sigma^2$ denotes the variance.*

That is, $b_t$ is continuously affected by gradient noise, which increases the risk of deviating from the optimal $b_t$, especially in online predictions where real-time updates reduce the stability of $b_t$. To mitigate the online forecasting instability caused by error accumulation, we first quantify the noise level of the online time series. This noise level is then used as a correction term for dynamic weight adjustment. The ONSP module evaluates the sequence variations by calculating the Dynamic Time Warping (DTW) distance between the input time series $X$ and time flow rhythm $X_r$ as:

$$C_{i,j} = D_{i,j} + \min\left(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}\right), \tag{6}$$

where $D_{i,j} = \left\|X^i - X_r^j\right\|$ represents the distance between $X$ at the *i*-th time step and $X_r$ at the *j*-th time step. The minimum cumulative distance $C$ is represented as the noise level of the current time series. In practice, we employ the more efficient computation method FastDTW Salvador & Chan (2007) to accelerate the calculation. We normalize $C$ using the sigmoid function and incorporate it as a confidence parameter into the dynamic weight integration process. The corresponding proof of Proposition 1 can be found in A.1.

To accelerate the response efficiency of the exponentially weighted moving average predictor to drastic distribution changes and instantaneous environmental shifts, we learn the short-term weights based on the long-term weights $w$ and the performance of the expert during the short-term history. We concatenate each predicted value, weighted by the expert weight $(w_{t,i}\cdot\tilde{Y}_i)$, with the results $Y_1, Y_2$ of two predictors, and introduce short-term weight offsets $b_{t,1}$ and $b_{t,2}$ associated with each predictor. The integrated weight is constructed as:

$$\widetilde{w}_{t,i} = \frac{w_{t,i} + b_{t,i}}{\sum_{i=1}^{d}(w_{t,i} + b_{t,i})}, \tag{7}$$

where $b_{t,i}$ represents the short-term weight offset for the $i$-th expert at time $t$. This approach allows independent short-term weights to better respond to their respective instantaneous impacts, and the obtained local confidence parameters are further combined.

### 4.3 OVERVIEW OF DMCNET

The overview of DMCnet is shown in Figure 3. DCT includes two components, the Time Flow Rhythm Predictor and the Energy Alignment Predictor, which are used to align the time flow rhythm and energy, respectively. The ONSP component detects the noise level of the time series and integrates it into the Dynamic Weight Integration module to eliminate error accumulation. The Dynamic Weight Integration component combines the predictions of the sub-models with their confidence parameters and dynamically updates the weights within the convex optimization framework. Specifically, DMCnet is an integrated model trained by $\mathcal{L}\left(\tilde{Y_1}, Y\right) + \mathcal{L}\left(\tilde{Y_2}, Y\right)$ and $\mathcal{L}\left(w_1 * \tilde{Y_1} + w_2 * \tilde{Y_2}, Y\right)$, with MSE as the loss function $\mathcal{L}$. We also provide the pseudocode of DMCnet in A.2.

## 5 EXPERIMENTS

In this section, we conduct experiments to study the performance of our proposed DMCnet method on five real-world datasets. Here we first introduce the experimental settings including General Datasets, the PCpower Dataset, Comparison Methods, and Implementation Details. Then the performance of the proposed method is compared with baseline methods and an ablation study is conducted to validate the effectiveness of individual modules. The is available at https://github.com/xiaojintesha2019/DMCNET. Particularly, the PCpower Dataset as a novel dataset is introduced and integrated into online time series forecasting, enriching the scope of our research. The dataset is available at https://github.com/xiaojintesha2019/DMCNET/tree/main/data.

### 5.1 EXPERIMENTAL SETTINGS

We delve into a variety of time series forecasting datasets to comprehensively evaluate model performance. The ETT datasets, as documented by Zhou et al. (2021), meticulously log the *oil temperature* target value along with six power load features over a span of two years. We further examine the ETTH2 and ETTm1 datasets, which capture observations at hourly and 15-minute intervals, respectively. The Weather dataset encompasses 11 climate features from nearly 1,600 locations across the United States, recorded hourly from 2010 to 2013. Additionally, the WTH dataset offers detailed weather records from multiple stations, significantly enhancing the depth and breadth of our analysis. We have released the first publicly available PCpower dataset of laptop power consumption, details of which can be found in A.3. We provide a concise overview of the comparison methods utilized in our experiments as baselines including **Informer** Zhou et al. (2021), **TCN** Woo et al. (2022), **ER** Chaudhry et al. (2019), **MIR** Aljundi et al. (2019a), **DER++** Buzzega et al. (2020), **TFCL** Aljundi et al. (2019b), **FSNet** Pham et al. (2022), **PatchTST** Nie et al. (2022), **Timemixer** Wang et al. (2024), **OneNet** Wen et al. (2023).

**Implementation Details.** In the experiment, we segmented the data into warm-up and online training phases with a 1:3 ratio. During the warm-up phase, 20% of the data served as the validation set. Aligning with online learning protocols, we set both the number of epochs and the batch size to 1. For all test models, we fixed the lookback window length at 60 and varied the prediction lengths across $\{24, 48\}$. We adhered to the optimization details in Zhou et al. (2021) and utilized the AdamW optimizer Loshchilov & Hutter (2018) to minimize the MSE loss. These settings ensured a fair comparison between all baselines and our method.

### 5.2 ONLINE FORECASTING RESULTS

As shown in Table 1 and Table 2, DMCnet demonstrates significant advantages in time series prediction tasks, particularly in terms of the MSE and MAE metrics. Through comparative experiments, DMCnet outperforms existing methods, including the previous best method OneNet, across all tested datasets (ETTH2, ETTm1, WTH, Weather, and PCpower) and prediction window lengths (H=24 and H=48).

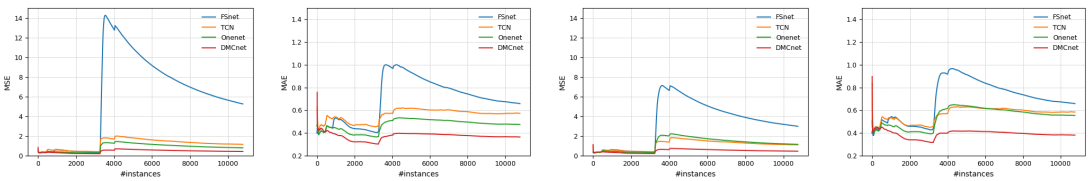
In terms of MSE, DMCnet achieves substantial improvements, consistently reducing forecasting errors across all datasets and forecasting horizons. The average MSE improvement across all datasets and horizons is $24.67\%$, highlighting its robust predictive accuracy. DMCnet shows an MSE improvement of over 20% on the ETTh2, ETTm1, and WTH datasets at $H = 24$ and $H = 48$ prediction horizons.

Table 1: MSE of Adaptation Methods. H: Forecast Horizon. **Bold**: Best results. Underlined: Second best results. Improvement: The improvement compared to the second-best method.

| MSE | ETTH2 | | ETTm1 | | WTH | | Weather | | PCpower | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method/H | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | Avg |
| Informer | 4.629 | 5.692 | 0.478 | 0.388 | 0.38 | 0.367 | - | - | - | - | 1.99 |
| TCN | 0.83 | 1.183 | 0.258 | 0.283 | 0.308 | 0.302 | 0.506 | 0.522 | 0.62 | 0.644 | 0.55 |
| TFCL | 0.846 | 1.208 | 0.211 | 0.236 | 0.301 | 0.323 | - | - | - | - | 0.52 |
| ER | 0.808 | 1.136 | 0.202 | 0.22 | 0.293 | 0.297 | 0.441 | 0.442 | 0.613 | 0.649 | 0.51 |
| MIR | 0.812 | 1.103 | 0.192 | 0.21 | 0.291 | 0.297 | - | - | - | - | 0.48 |
| DER++ | 0.828 | 1.157 | 0.196 | 0.208 | 0.287 | 0.294 | 0.434 | 0.435 | 0.612 | 0.648 | 0.51 |
| FSNet | 0.687 | 0.846 | 0.115 | 0.127 | 0.188 | 0.223 | 0.357 | 0.399 | 0.542 | 0.578 | 0.41 |
| Time-TCN | 0.779 | 1.307 | 0.281 | 0.308 | 0.311 | 0.308 | 0.62 | 0.72 | 0.577 | 0.627 | 0.58 |
| PatchTST | 1.622 | 2.716 | 0.427 | 0.553 | 0.372 | 0.465 | 1.52 | 2.42 | 0.601 | 0.653 | 1.13 |
| Timemixer | 1.664 | 2.776 | 0.403 | 0.517 | 0.401 | 0.455 | 0.929 | 1.282 | 1.283 | 1.791 | 1.15 |
| OneNet-TCN | 0.772 | 0.806 | 0.212 | 0.223 | 0.293 | 0.31 | 0.476 | 0.524 | 0.546 | 0.575 | 0.47 |
| OneNet | 0.532 | 0.609 | 0.098 | 0.108 | 0.175 | 0.2 | 0.314 | 0.428 | 0.513 | 0.553 | 0.35 |
| DMCnet(ours) | **0.413** | **0.444** | **0.078** | **0.079** | **0.132** | **0.147** | **0.292** | **0.313** | **0.375** | **0.386** | **0.27** |
| Improvement | 22.37%↑ | 27.09%↑ | 20.41%↑ | 26.85%↑ | 24.57%↑ | 26.50%↑ | 7.01%↑ | 26.87%↑ | 26.90%↑ | 30.20%↑ | 24.67%↑ |

Table 2: MAE of Adaptation Methods. H: Forecast Horizon. **Bold**: Best results. Underlined: Second best results. Improvement: The improvement compared to the second-best method.

| MAE | ETTH2 | | ETTm1 | | WTH | | Weather | | PCpower | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method/H | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | Avg |
| Informer | 0.668 | 0.752 | 0.525 | 0.46 | 0.417 | 0.419 | - | - | - | - | 0.54 |
| TCN | 0.547 | 0.589 | 0.381 | 0.403 | 0.367 | 0.362 | 0.312 | 0.326 | 0.521 | 0.535 | 0.43 |
| TFCL | 0.548 | 0.592 | 0.341 | 0.363 | 0.363 | 0.382 | - | - | - | - | 0.43 |
| ER | 0.543 | 0.571 | 0.333 | 0.351 | 0.356 | 0.363 | 0.264 | 0.281 | 0.513 | 0.531 | 0.41 |
| MIR | 0.541 | 0.565 | 0.325 | 0.342 | 0.355 | 0.361 | - | - | - | - | 0.41 |
| DER++ | 0.54 | 0.577 | 0.326 | 0.34 | 0.351 | 0.359 | 0.256 | 0.273 | 0.511 | 0.53 | 0.41 |
| FSNet | 0.467 | 0.515 | 0.249 | 0.263 | 0.276 | 0.301 | 0.244 | 0.292 | 0.462 | 0.485 | 0.36 |
| Time-TCN | 0.544 | 0.636 | 0.395 | 0.421 | 0.378 | 0.378 | 0.367 | 0.447 | 0.479 | 0.508 | 0.46 |
| PatchTST | 0.577 | 0.672 | 0.471 | 0.549 | 0.393 | 0.459 | 0.659 | 0.842 | 0.482 | 0.512 | 0.56 |
| Timemixer | 0.578 | 0.68 | 0.459 | 0.528 | 0.417 | 0.453 | 0.436 | 0.562 | 0.568 | 0.609 | 0.53 |
| OneNet-TCN | 0.511 | 0.543 | 0.319 | 0.371 | 0.345 | 0.356 | 0.278 | 0.319 | 0.463 | 0.483 | 0.40 |
| OneNet | 0.407 | 0.436 | 0.225 | 0.238 | 0.255 | 0.279 | 0.269 | 0.395 | 0.455 | 0.478 | 0.34 |
| DMCnet(ours) | **0.361** | **0.381** | **0.204** | **0.211** | **0.221** | **0.227** | **0.207** | **0.212** | **0.374** | **0.382** | **0.28** |
| Improvement | 11.30%↑ | 12.61%↑ | 9.33%↑ | 11.34%↑ | 13.33%↑ | 18.64%↑ | 15.16%↑ | 22.34%↑ | 17.80%↑ | 20.08%↑ | 19.32%↑ |




(a) (b) (c) (d)

Figure 4: Cumulative loss evolution during training for the four primary methods on ETTH2 dataset with forecasting window $H = 24$ and $H = 48$. (a) Mean Squared Error (MSE) with $H = 24$. (b) Mean Absolute Error (MAE) with $H = 24$. (c) Mean Squared Error (MSE) with $H = 48$. (d) Mean Absolute Error (MAE) with $H = 48$.

While the prediction accuracy improvement on the Weather dataset is slightly lower, it still achieves an MSE improvement of $7.01\%$ at $H = 24$ . Particularly encouraging are the test results on the

PCpower dataset, where it achieves an MSE improvement of $26.9\%$ at $H = 24$ and $30.2\%$ at H=48, significantly outperforming the current competitive method OneNet. For MAE, DMCnet also performs excellently, consistently improving across most datasets and prediction horizons. The average MAE improvement across all datasets and horizons is $19.32\%$. Particularly, on the PCpower dataset, MAE improves by $17.8\%$ at $H = 24$ and $20.08\%$ at $H = 48$. the improvements brought by DMCnet are more pronounced, indicating its superior performance on longer forecasting windows.

### 5.3 Visualization and Convergence Analysis

To validate the effectiveness of DMCnet in addressing trend and patch misalignment, we randomly visualized online forecasting samples, as shown in Figure 2. Figures 2(a) and 2(b) demonstrate the trend and patch misalignment within forecasting results of the competitive baseline methods OneNet and the classical method TCN on the ETTH2 dataset. Figure 2(c) shows the forecasting results of DMCnet on the same test samples, clearly showing excellent amplitude alignment at the peaks and valleys, and good matching of the time flow rhythm. Tigure 4(d) further highlight this finding and the baseline models, influenced by anomalous samples, struggle to restore the model to its initial low MAE level. In contrast, DMCnet can rapidly adjust, effectively avoiding the sustained impact of anomalous samples or accumulation errors.

### 5.4 Ablation Studies

In this research, we systematically investigated the effectiveness of each design component through a comprehensive ablation study. The results of these ablation studies are presented in Table 3 and Table 5, which report the MSE and MAE metrics, respectively. These tables compare the performance of three configurations: the standalone Trend-oriented Predictor (Only_Y1), the standalone Patch-oriented Predictor (Only_Y2), and the model with the ONSP module removed (Remove_ONSP), against our proposed DMCnet framework. As shown in Table 3 (MSE results), the standalone predictors achieved average MSE values of 0.53 (Only_Y1) and 0.60 (Only_Y2), while the Remove_ONSP configuration achieved an average MSE of 0.28. In contrast, our DMCnet framework achieved a significantly lower average MSE of 0.27, demonstrating a relative improvement of 3.57% over the Remove_ONSP configuration. The corresponding results regarding MAE can be found in A.

Table 3: Ablation studies of the individual results of Only_Y1, the individual results of Only_Y2, and the ablation results with Remove_ONSP, where the metric is MSE.

| Ablation_MSE | ETTH2 | | ETTm1 | | WTH | | Weather | | PCpower | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method/H | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | Avg |
| Only_Y1 | 0.799 | 0.901 | 0.231 | 0.243 | 0.277 | 0.289 | 0.54 | 0.557 | 0.669 | 0.747 | 0.53 |
| Only_Y2 | 0.835 | 0.896 | 0.184 | 0.17 | 0.346 | 0.328 | 0.52 | 0.529 | 1.094 | 1.091 | 0.60 |
| Remove_ONSP | 0.461 | 0.508 | 0.083 | 0.084 | 0.15 | 0.167 | 0.292 | 0.315 | 0.376 | 0.398 | 0.28 |
| DMCnet(ours) | **0.413** | **0.444** | **0.078** | **0.079** | **0.132** | **0.147** | **0.292** | **0.313** | **0.375** | **0.386** | 0.27 |

## 6 Conclusion

This paper proposes an online time series forecasting model DMCnet, which reduces online forecasting errors by enhancing the perception of trend and patch in online time series. We introduce an ONSP that can quantify the noise level in real time and integrate it into the dynamic adjustment process of the online convex optimization framework to mitigate the impact of accumulation errors. Additionally, this paper open-sourced the first PCpower dataset, which comprehensively reflects the power consumption characteristics of personal computers in various usage scenarios, providing a data standard for power forecasting and control in PC product development. And the dynamic adjustment process integrates multiple components, which inevitably reduces computation efficiency. Developing more efficient and simplified alternatives could be an interesting direction for future work.

## AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for [Article revision]. We have not used generative AI tools for [other tasks with required disclosure], and [the rest of the required disclosure tasks] are not applicable to this work. Additionally, we used generative AI tools for [tasks with recommended disclosure]. We have reviewed all AI-assisted work. [Elaborate. For example, "we checked LLM-generated research ideas for potential plagiarism through a manual literature survey", "LLM-generated code was verified and tested for correctness by 2 authors", etc.]. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

See the ICLR 2027 AI Policy for Authors for more details. This statement should not be more than 1 page.

## 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.

Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664*, 2022.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36, 2022.

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.

Zonglei Chen, Minbo Ma, Tianrui Li, Hongjun Wang, and Chongshou Li. Long sequence time-series forecasting with deep learning: A survey. *Information Fusion*, 97:101819, 2023.

Dawei Cheng, Fangzhou Yang, Sheng Xiang, and Jin Liu. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*, 121:108218, 2022.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Simon I Hill and Robert C Williamson. Convergence of exponentiated gradient algorithms. *IEEE Transactions on Signal Processing*, 49(6):1208–1215, 2001.

Wendi Li, Xiao Yang, Weiqing Liu, Yingce Xia, and Jiang Bian. Ddg-da: Data distribution generation for predictable concept drift adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4092–4100, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

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

Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106, 2004.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024.

A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *NIPS*, 2017.

Qingsong Wen, Linxiao Yang, Tian Zhou, and Liang Sun. Robust time series analysis and applications: An industrial perspective. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4836–4837, 2022.

Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, 36:69949–69980, 2023.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*, 2022.

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

YiFan Zhang, Weiqi Chen, Zhaoyang Zhu, Dalin Qin, Liang Sun, Xue Wang, Qingsong Wen, Zhang Zhang, Liang Wang, and Rong Jin. Addressing concept shift in online time series forecasting: Detect-then-adapt. *arXiv preprint arXiv:2403.14949*, 2024.

Lifan Zhao and Yanyan Shen. Proactive model adaptation against concept drift for online time series forecasting. *arXiv preprint arXiv:2412.08435*, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

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

# A Appendix

## A.1 Proof of Proposition 1

*Proof.* The short-term weight $b_t$ is updated by minimizing the loss function using a gradient descent strategy:

$$b_{t+1} = b_t - \eta_b \cdot \nabla_{b_t} \mathcal{L}_t \left( b_t \right). \tag{8}$$

Using the chain rule, the derivative of L is $\nabla_{b_t}\mathcal{L}_t = 2\left(\hat{y}_t\left(b_t\right) - y_t\right)^\top \cdot \nabla_{b_t}\hat{y}_t\left(b_t\right)$. Substituting $y_t = y_t^* + \varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2 I_d\right)$ can be regarded as the real-time noise and $y_t^*$ is the ideal signal, we can get:

$$b_{t+1} = b_t - 2\eta_b\left(\hat{y}_t\left(b_t\right) - y_t^*\right)^\top \nabla_{b_t}\hat{y}_t\left(b_t\right) + 2\eta_b\varepsilon_t^\top \nabla_{b_t}\hat{y}_t\left(b_t\right). \tag{9}$$

Let $\Delta b_t^{\text{signal}} := -2\eta_b\left(\hat{y}_t\left(b_t\right) - y_t^*\right)^\top \nabla_{b_t}\hat{y}_t\left(b_t\right)$ denote the update direction guided by the ideal signal, and let $\Delta b_t^{\text{noise}} := 2\eta_b\varepsilon_t^\top \nabla_{b_t}\hat{y}_t\left(b_t\right)$ denote the disturbance to the update caused by the observed noise. So Equation 9 can be expressed as:

$$b_{t+1} = b_t + \Delta b_t^{\text{signal}} + \Delta b_t^{\text{noise}}. \tag{10}$$

Let the initial error be $\delta b_1 := b_1 - b_1^*$, and the error at any step be $\delta b_{t+1} = \delta b_t + \Delta b_t^{\text{noise}} + \epsilon_t^{(\text{h.o.t.})}$. Neglecting higher-order terms (assuming small perturbations at each step), we have:

$$\delta b_T = \sum_{t=1}^{T-1} \Delta b_t^{\text{noise}} + \delta b_1. \tag{11}$$

Since each $\Delta b_t^{\text{noise}}$ is independent, the covariance is:

$$\text{Var}\left[\delta b_T\right] = 4\eta_b^2\sigma^2 \sum_{t=1}^{T-1} \left\|\nabla_{b_t}\hat{y}_t\left(b_t\right)\right\|_F^2 = 4\eta_b^2\sigma^2 \sum_{t=1}^{T-1} G_t. \tag{12}$$

□

### A.2 Algorithm of DMCnet

**Algorithm 1** Training and Inference Algorithm of DMCnet

**Input:** Historical multivariate time series $x \in \mathbb{R}^{L\times d}$, forecast target $y \in \mathbb{R}^{P\times m}$, learning rate $\eta$;
**Output:** Final prediction $\hat{y}$;

1: Initialize time flow rhythm predictor $f_1$, energy alignment predictor $f_2$;
2: Initialize weights $w_1 = 0.5, w_2 = 0.5$, confidence coefficient $C$, and short-term offsets $b_1, b_2$;
3: $X_r, X_e \leftarrow$ TemporalDecoupling$(x)$; {Decouple input into time-related and energy-related components}
4: $\hat{y}_1 \leftarrow f_1(X_r)$; {Time flow rhythm prediction}
5: $\hat{y}_2 \leftarrow f_2(X_e)$; {Energy alignment prediction}
6: $C \leftarrow$ CalculateConfidence$(x, X_r)$; {Calculate confidence coefficient using ONS module}
7: $b_1 \leftarrow \text{bias}_1$, $b_2 \leftarrow C \times \text{bias}_2$; {Update short-term weight offsets}
8: Compute dynamic weights:

$$\tilde{w}_1 = \frac{w_1 + b_1}{(w_1 + b_1) + (w_2 + b_2)}, \quad \tilde{w}_2 = 1 - \tilde{w}_1$$

9: $\hat{y} = \tilde{w}_1\hat{y}_1 + \tilde{w}_2\hat{y}_2$;
10: Update long-term weights (EGD)
11: Update $f_1$, $f_2$ using Adam on $\mathcal{L}(\hat{y}, y)$;
12: **return** $\hat{y}$

### A.3 PCpower Dataset

In recent years, personal computers have become ubiquitous, and understanding their power consumption characteristics is crucial for enhancing system energy efficiency, performance, and user experience. The PCpower dataset is designed to collect power consumption data from various components (Figure 5) under diverse usage scenarios. By analyzing the relationships among components and correlating them with user behaviors, the dataset aims to achieve objectives such as detecting usage scenarios, identifying abnormal power consumption patterns, and predicting power consumption trends. These insights can help optimize control strategies for computer systems, ultimately improving energy management and operational efficiency. To the best of our knowledge, this is the

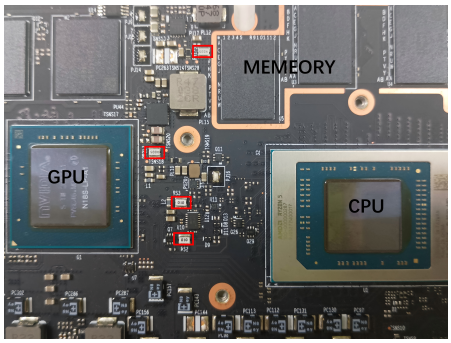



Figure 5: The hardware motherboard schematic of a notebook personal computer illustrates the apparatus utilized during the dataset collection process. Some key parts are annotated in black, while the precision resistors employed for data acquisition are highlighted in red.

first open-source dataset for personal computer usage scenarios. Refer to the Section A.3 for more information about the PCpower dataset.

The PCpower dataset focuses on the power consumption of mobile laptops. Its data comes from strict power consumption monitoring and collection of various laptop components in real user scenarios. The goal is to create a resource that accurately reflects the power consumption characteristics during actual operation. The dataset provides in-depth analysis of detailed energy consumption data, revealing the interaction mechanisms of components under different function calls and operational states. It builds quantitative models with different usage scenarios and behavior patterns. This dataset provides crucial data for understanding the theoretical aspects of computer system energy consumption, formulating subsequent data-driven optimization strategies, developing anomaly detection algorithms, and constructing power consumption prediction models.

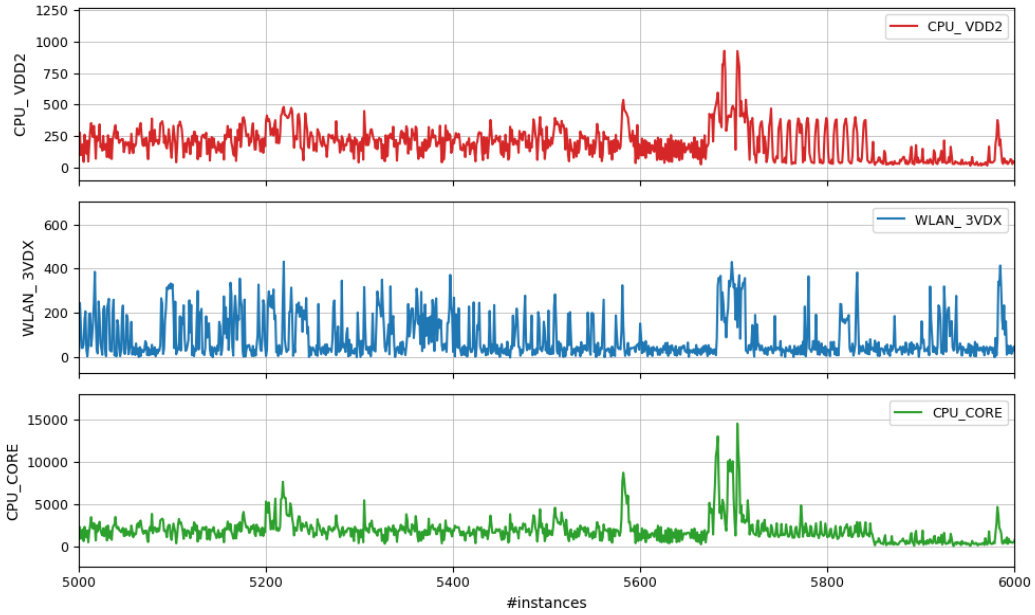



Figure 6: Visual diagram of the core variables of PCpower dataset, with Instance: 5000-6000

### A.3.1 Collection Method of PCpower Dataset

The data collection method of the PCpower dataset is based on rigorous electrical principles and advanced technologies, ensuring the scientific, accurate, and complete nature of the data. Based on

Table 4: Parameter of PCpower Dataset

| Parameter | Description | Details |
|---|---|---|
| **Dataset Name** | PCpower dataset | Focuses on mobile notebook power consumption. |
| **Data Source & Scenarios** | Real - user scenarios of mobile notebook computers | Idle & sleep, office work(word excel ppt file edition), professional creation( video file edition&Import and Export), webpage browsing (20 different categories of websites) , play games(Dota2&CSGO) |
| **Time Range & Sampling Frequency** | Collect data samples in real time | 2 hours per usage scenario & 5 times per second _36000 sampl points |
| **Number of Data Points** | 20+ components, 5 usage scenarios, 5 notebook models | Large - scale coverage across hardware and scenarios. |
| **Components** | CPU, GPU, memory, SSD, camera, display, audio, battery, south bridge and.. | details in datasheets |
| **Notebook Models** | Low - power (15W), performance - type (28W) performance (30W) performance (30W) + DGPU high - performance (80W) + DGPU | config1:Intel Core i7-1265U +16GB memory<br>config2:Intel Core i7-1360P +32GB memory<br>config3:AMD Ryzen 7 8845H+16GB memory<br>config4:Intel Core Ultra 7-175H +32GB memory + NV RTX 4050<br>config5:Intel Core i9-14900HX +32GB memory+NV RTX4070 |
| **Application Scenarios** | Time series prediction, classification, and behavior annotations detection | Use cases include optimization, anomaly detection, and efficiency management. |

electrical principles, high-precision real hardware precision resistors are employed to monitor the power usage of each channel in real time. Specifically, the potential difference across the precision resistors in the power path is precisely measured, as shown in Figure 7. Combined with the known and calibrated resistance value, the current flowing through each component is accurately calculated using **Ohm's Law**:

$$I = \frac{V}{R}, \tag{13}$$

where $I$ represents current, $V$ represents voltage difference, and $R$ represents resistance. Subsequently, the actual power consumption of each component at different times is determined using the **power calculation formula**:

$$P = VI, \tag{14}$$

where $P$ represents power, $V$ represents voltage, and $I$ represents current. This is achieved by integrating the measured voltage and calculated current values.

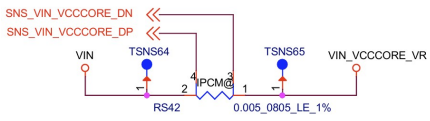



Figure 7: The schematic diagram demonstrates the circuit of the precision resistors in key components power rail.

Advanced analog to digital conversion technology (ADC) is used to convert the collected analog signals into digital signals, facilitating efficient data transmission and collection while ensuring processability and stability. Specifically, the original analog signals collected carry the real-time power consumption change information of each component. After being processed by ADC according to the preset sampling precision and conversion rate, they are stably output to external professional collection devices. The collection parameters are shown in Table 4.

Considering the dynamic change characteristics of the power consumption of each component of the notebook computer in different usage scenarios, the sampling rate is set to 5 times per second. This relatively high sampling frequency can finely capture the instantaneous power consumption changes of each component in the time dimension, effectively avoiding data information loss caused by too large sampling intervals. At the same time, for the various complex scenarios that users may encounter in actual use, a collection duration of 2 hours is set for each usage scenario, which can cover the complete cycle from a single-operation behavior to a complex task combination.

In the comprehensive consideration of the collection coverage range, the collection work covers the power supply channels corresponding to more than 20 key components such as CPU, GPU, memory, SSD, camera, display, audio, battery, and south bridge, ensuring that the power consumption contributions of each core component in different operating states are captured without any omission. In addition, the dataset collection covers 5 typical and representative usage scenarios: standby, office work, professional creation, entertainment webpages, and games. It also includes 5 types of notebook computers with low-power (15W), performance (30W), and high-performance (80W) CPU series, as well as integrated graphics cards and independent graphics cards. This extensive model coverage strategy aims to ensure that the dataset can reflect the power consumption characteristics of notebook computers with different hardware configurations and functional positions in actual use. Figure 6 shows the visualization of key dimensions based on the data collected from actual measurements.

### A.3.2 Annotation of PCpower Dataset

The annotation of the PCpower dataset follows the principle of accurately depicting the association between user behavior and component power consumption, and constructs a clear, layered, and logically rigorous annotation system for different usage scenarios. For example, in the entertainment web page scenario, key information such as the URL of each newly opened web page, the time of access, and the page-staying duration is recorded in detail. This aims to outline the user's behavior trajectory in the web browsing process and the corresponding power consumption changes of each component through these fine-grained annotations.

In the office work scenario, for the key operations of commonly used office software (such as Word, Excel, PPT, etc.), including inserting pictures, editing text, creating and saving files, the specific timestamps, operation sequences, and related file attributes are all meticulously annotated. In this way, the mapping relationship between office-operation behavior and the power consumption of each component is established.

In the professional creation scenario, for the core links involved in video editing, such as importing videos, editing videos, previewing videos, and exporting and rendering, not only the start and end times and operation parameters of each link are recorded, but also the power consumption status changes of each component of the notebook computer corresponding to them are annotated.

### A.3.3 Application Scenarios of PCpower Dataset

The PCpower dataset has a wide range of applications in the field of time series analysis. **In time series forecasting**, the future trend of component power consumption is predicted, which provides forward-looking support for system resource allocation, energy consumption optimization, and fault early warning. **In the time series classification**, the data of multi-scene, multi-model and multi-component are used to accurately judge the current usage scenario of the notebook, help the system to automatically adjust the performance parameters, and provide intelligent services based on scene awareness. **In time series detection**, time series analysis technology and behavior knowledge graph are combined to detect the key operation behaviors or specific intervals of users, realize intelligent monitoring, fault diagnosis and user behavior guidance, and improve user experience and system reliability.

## A.4 Comparison Methods

We provide a concise overview of the comparison methods utilized in our experiments as baselines including

- **Informer** Zhou et al. (2021): Informer is a Transformer-based model for long-range time series forecasting, leveraging probabilistic sparse self-attention to reduce computational complexity and memory usage, making it suitable for long sequence data.
- **TCN** Woo et al. (2022): This method employs a conventional Temporal Convolutional Network (TCN) architecture, which consists of ten hidden layers. Each layer comprises two stacks of residual convolutional filters.
- **ER** Chaudhry et al. (2019): Experience Replay (ER) extends the TCN baseline by incorporating an episodic memory. This memory stores previous samples and interleaves them with newer ones during the learning process.
- **MIR** Aljundi et al. (2019a): Memory Importance Reweighting (MIR) improves upon ER by replacing its random sampling technique with a more strategic approach. MIR selects samples from the memory that cause the highest forgetting and applies ER to these samples.
- **DER++** Buzzega et al. (2020): Dynamic Episodic Replay with Knowledge Distillation (DER++) enhances the standard ER method. It incorporates a knowledge distillation loss on the previous logits, thereby improving the learning process.
- **TFCL** Aljundi et al. (2019b): Task-Free Continual Learning (TFCL) is an online continual learning method that starts with an ER process. It includes a task-free MAS-styled regularization, which helps in maintaining the stability of the learning process.
- **FSNet** Pham et al. (2022): FSNet is an online time series forecasting framework inspired by the Complementary Learning Systems theory, balancing fast adaptation and knowledge retrieval through per-layer adapters and associative memory.
- **PatchTST** Nie et al. (2022): PatchTST is a Transformer-based model that employs variable-independent modeling for time series forecasting, processing each variable separately to avoid overfitting and enhance robustness.
- **Timemixer** Wang et al. (2024): Timemixer uses a fully MLP-based architecture to take full advantage of disentangled multiscale series in both past extraction and future prediction phases.
- **OneNet** Wen et al. (2023): OneNet is an online time series forecasting model with a dual-stream architecture for modeling temporal and variable dependencies. Time-TCN, a convolutional neural network, captures temporal dependencies via causal and dilated convolutions. OneNet-TCN integrates TCN and Time-TCN, addressing concept drift through an online ensemble mechanism. OneNet combines FSNet and Time-FSNet to handle complex patterns in multivariate time series.

### A.5 Supplementary results

#### A.5.1 Ablation experiment

For 5, the standalone predictors achieved average MAE values of 0.41 (Only Y1) and 0.41 (Only Y2), while the Remove ONSP configuration achieved an average MAE of 0.30. Our DMCnet framework further reduced the average MAE to 0.28, representing a relative improvement of 6.67% over the Remove ONSP configuration.

Table 5: Ablation studies of the individual results of Only_Y1, the individual results of Only_Y2, and the ablation results with Remove_ONSP, where the metric is MAE.

| Ablation_MAE | ETTH2 | | ETTm1 | | WTH | | Weather | | PCpower | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method/H | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | 24 | 48 | Avg |
| Only_Y1 | 0.567 | 0.591 | 0.362 | 0.374 | 0.353 | 0.362 | 0.339 | 0.353 | 0.375 | 0.386 | 0.41 |
| Only_Y2 | 0.57 | 0.581 | 0.322 | 0.309 | 0.373 | 0.361 | 0.338 | 0.342 | 0.462 | 0.46 | 0.41 |
| Remove_ONSP | 0.376 | 0.398 | 0.205 | 0.209 | 0.223 | 0.241 | 0.196 | 0.214 | 0.446 | 0.444 | 0.30 |
| DMCnet(ours) | **0.361** | **0.381** | **0.204** | **0.204** | **0.221** | **0.227** | **0.207** | **0.212** | **0.374** | **0.382** | 0.28 |