Applied Soft Computing 144 (2023) 110467

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Fast sharpness-aware training for periodic time series classification and forecasting

Jinseong Park [a], Hoki Kim [a,b], Yujin Choi [a], Woojin Lee [c], Jaewook Lee [a,*]

[a] Department of Industrial Engineering, Seoul National University, Gwanakro 1, Seoul, Republic of Korea
[b] Institute of Engineering Research, Seoul National University, Gwanakro 1, Seoul, Republic of Korea
[c] School of AI Convergence, Dongguk University-Seoul, Pildong-ro 30, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 26 September 2022
Received in revised form 3 April 2023
Accepted 20 May 2023
Available online 30 May 2023

Keywords:
Sharpness-aware minimization
Time series classification
Time series forecasting
Periodic time series
Computational efficiency

ABSTRACT

Various deep learning architectures have been developed to capture long-term dependencies in time series data, but challenges such as overfitting and computational time still exist. The recently proposed optimization strategy called Sharpness-Aware Minimization (SAM) optimization prevents overfitting by minimizing a perturbed loss within the nearby parameter space. However, SAM requires doubled training time to calculate two gradients per iteration, hindering its practical application in time series modeling such as real-time assessment. In this study, we demonstrate that sharpness-aware training improves generalization performance by capturing trend and seasonal components of time series data. To avoid the computational burden of SAM, we leverage the periodic characteristics of time series data and propose a new fast sharpness-aware training method called Periodic Sharpness-Aware Time series Training (PSATT) that reuses gradient information from past iterations. Empirically, the proposed method achieves both generalization and time efficiency in time series classification and forecasting without requiring additional computations compared to vanilla optimizers.

1. Introduction

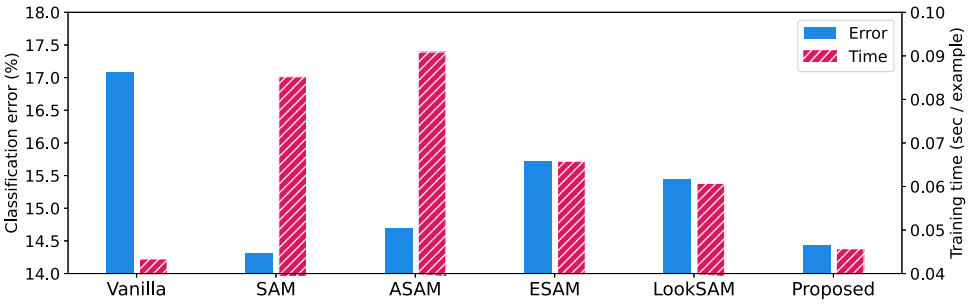
Time series classification and forecasting have demonstrated their effectiveness in various decision-making processes such as financial prediction [1,2], risk management [3,4], energy consumption [5], and traffic [6]. Even though deep learning has been successful in a wide variety of tasks, traditional deep learning models have limitations in distinguishing overall trends and are susceptible to overfitting short-term patterns and noise [7]. This is because modeling time series data differs from building image or language models. Time series data usually consists of low-dimensional data, where each historical data shows continuous patterns over time. To mitigate this limitation, recent deep learning models [5,8,9] emphasize building unique structures to capture long-term dependencies by connecting different time periods.

Optimization methods to enhance generalization ability are important but often overlooked aspects of time series modeling, in addition to architecture design. Specifically, finding flat minima is important in time series modeling to capture the overall trend and cyclic seasonality and be robust against irregular noises. To prevent overfitting, various studies [10–12] have investigated geometric properties to find well-generalizing minima. Recently, Foret et al. [12] proposed a novel training method called Sharpness-Aware Minimization (SAM) that makes the loss landscape flatter by minimizing the maximum loss in the nearby parameter space. This method has achieved state-of-the-art generalization performance in various domains [13–15]. However, sharpness-aware training requires **doubled** ($2\times$) computational time compared to SGD or Adam to find the maximum loss. This computation overhead hinders time series models from updating frequently and quickly to keep up-to-date with new data, such as online learning and real-time assessment.

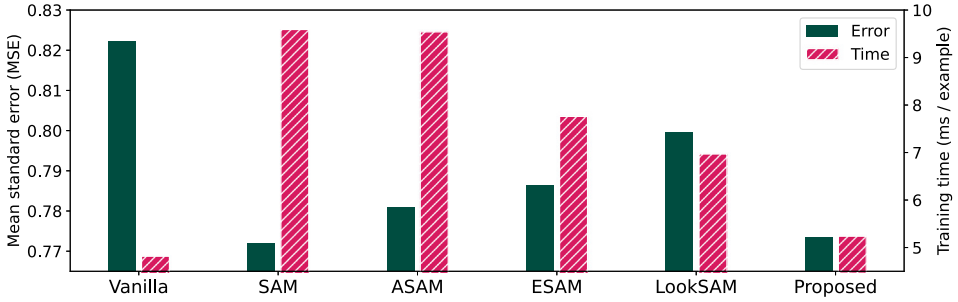
In this paper, we aim to achieve both flatness and speed during training by mitigating the computational inefficiency of sharpness-aware training in time series classification and forecasting. We first demonstrate that periodic characteristics can produce similar effects on training loss among data from different time periods in both toy examples and real data. By utilizing the periodicity, we propose a new sharpness-aware training method called Periodic Sharpness-Aware Time series Training (PSATT) that reuses gradient information from past iterations. By skipping gradient calculation for inner maximization, our method achieves better generalization with minimal computational burden compared to vanilla optimizers. Our experiments demonstrate the effectiveness and efficiency of PSATT in time series classification and forecasting. The main results of classification and forecasting are summarized in Fig. 1. With lower errors and faster training

* Corresponding author.
E-mail address: jaewook@snu.ac.kr (J. Lee).

https://doi.org/10.1016/j.asoc.2023.110467
1568-4946/



(a) Classification tasks: classification error vs. training time



(b) Forecasting tasks: mean standard error vs. training time

Fig. 1. Test error (lower is better) and training time (lower is better) of vanilla, SAM [12], ASAM [16], ESAM [17], LookSAM [18], and the proposed PSATT. PSATT achieves both generalization performance and training efficiency.

time, the proposed PSATT outperforms vanilla training and any variants of SAM in time series modeling.

The main contributions are summarized as follows:

- We demonstrate the effectiveness of finding flat minima to improve the generalization performance of existing time series architectures. To our knowledge, this is the first attempt to investigate the sharpness-aware training in time series data.
- We propose a new fast sharpness-aware training method called Periodic Sharpness-Aware Time series Training (PSATT) that leverages the periodic characteristics of time series data to avoid the additional gradient calculation required by SAM.
- The experimental results show that our approach achieves both generalization and time efficiency in time series classification and forecasting. Specifically, our method achieves over 3% accuracy gain on classification models and about 5% error reduction rate on state-of-the-art forecasting models without the need for additional computations.

2. Related work

2.1. Network architecture design for time series data

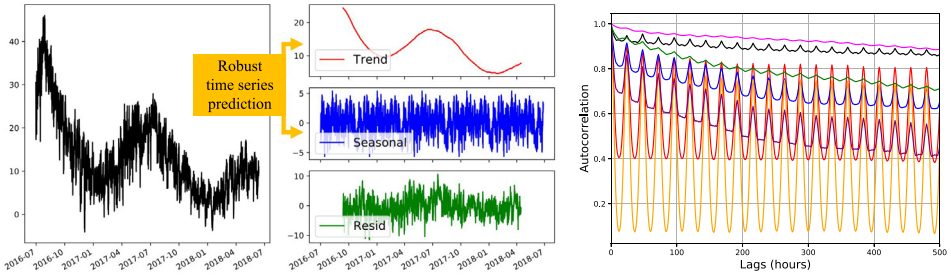
Classical statistical methods [19,20] have explored the detection of intrinsic patterns within data, by decomposing the data into three fundamental components: trend, seasonal, and remainder irregular components. The effectiveness of this approach has been demonstrated through datasets like ETTh1 [5], as illustrated in Fig. 2(a). Furthermore, the data shows strong autocorrelation and repeated patterns over time, as depicted in Fig. 2(b).

Recent years have seen a growing interest in time series deep learning models to overcome the difficulty of statistical approaches on capturing complicated short-term and long-term dependencies. Following [7], we divide time series architectures into three categories: (i) recurrent neural networks (RNN), (ii) convolutional neural networks (CNN), and (iii) Transformer networks [21].

RNN-based methods, the most traditional neural network for time series, developed sequence-to-sequence prediction by recursively propagating the fast information to the future inputs. RNN and its variants, such as LSTM and GRU, successfully captured temporal dependencies in classification [22,23] and forecasting [24,25]. However, RNN-based models suffer from capturing long-term patterns because of exploding/vanishing gradients [26] and recursively accumulated errors [27].

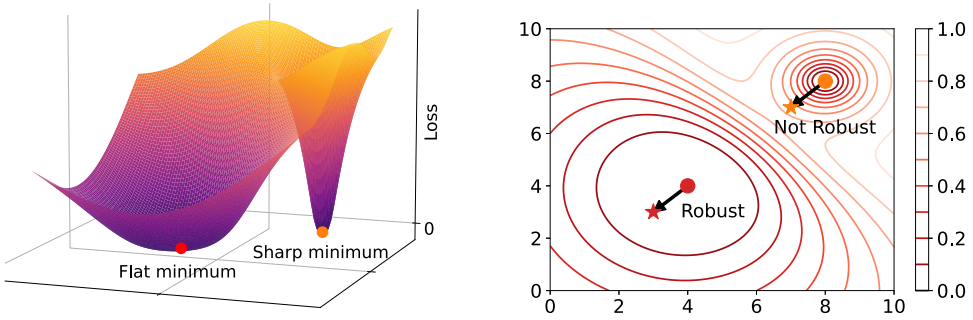
CNN-based models are designed to capture local correlation of time series data using convolutions filters for classification [28,29] and forecasting [30]. Especially, Bai et al. [31] suggested a Temporal Convolutional Network (TCN) that learns the local and repeated patterns of time series data through dilated convolutions. A unique structure connecting convolutions discontinuously allows TCN-based models to capture long-term repeating patterns [6,32]. One of the state-of-the-art forecasting models is a hierarchical TCN-based model called Sample Convolution and Interaction Networks (SCINet) [9]. Using multiple binary tree structures, SCINet progressively downsamples the input data and interactively learns from the coarser sequences.

By contrast, Transformer-based [21] methods capture the long-term dependency using attention mechanisms. However, those methods [33,34] entail high computational burden. Thus, Zhou et al. [5] suggested Informer, a new forecasting model with sparse self-attention, to mitigate the computations in time series forecasting. Liu et al. [8] adopted pyramidal tree structures to efficiently capture long-term dependencies, which can reduce the number of parameters and calculations while enhancing forecasting abilities and recent models [35,36] are designed to capture long-term and cross-dimensional dependencies.



(a) Seasonal and Trend decomposition of ETTh1 (b) Autocorrelation of ETTh1 features

Fig. 2. Unique characteristics and decomposition of time series data (ETTh1).



(a) Loss landscape (b) Contour map

Fig. 3. Illustration of flat and sharp minima. The flat minimum shows the better robustness to the same size of perturbation than the sharp one.

2.2. Optimization strategies to alleviate overfitting

Generalization is a central topic in improving the performance of overfitted deep learning architectures. An overparameterized network trained with a large number of parameters can converge to zero training error and even fit random noise [37]. To avoid this overfitting, various studies [11,38,39] investigated the geometrical property of loss landscape of model parameters to find well-generalizing minima.

Flat minima, described by having small eigenvalues of the Hessian matrix, usually generalize better than sharp minima [11,12]. As shown in Fig. 3, the flatness of a model is directly related to the robustness because the flat minimum shows a lower loss increase with irregular noise of time series data. Zhang et al. [37] proved that SGD, the primary key to the success of neural networks, can implicitly converge to well-generalizing minima, especially with the small batch size and large learning rate by searching flat minima [10]. Optimization techniques, such as batch normalization [40] and residual connection [38], are also revealed to find flatter loss.

Recently, Foret et al. [12] proposed a new optimization method called Sharpness-Aware Minimization (SAM). SAM updates a weight to minimize the worst loss in its neighborhood to avoid sharp minima in two steps. With this simple idea, SAM could effectively find flat minima and achieve state-of-the-art performance on a wide range of domains, such as vision [13] and language [15] models.

3. Preliminaries

3.1. Notations

- $\boldsymbol{x} \in \mathbb{R}^d$: A time series data sample where $d = 1$ indicates univariate time series inputs and $d > 1$ denotes multivariate time series inputs.
- $X := \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p, \ldots, \boldsymbol{x}_L\} \in \mathbb{R}^{d \times L}$: time series data with length L with $\boldsymbol{x}_p$ in a time period p, where $\boldsymbol{x}_{p_1:p_2} := \{\boldsymbol{x}_{p_1}, \boldsymbol{x}_{p_1+1}, \ldots, \boldsymbol{x}_{p_2}\}$ is an ordered set of $\boldsymbol{x}_p$ between time periods p_1 and p_2.
- $B^{(i)}$: Mini-batch time series set of randomly selected $\boldsymbol{x}_{p_1:p_2}$. A superscript is used to denote the index for data samples i.
- $k, k' \in \mathbb{R}$: Look-forward and look-back window sizes of a time series model.
- $f(\boldsymbol{x}) := f(\boldsymbol{w}; \boldsymbol{x})$: A prediction function f of input $\boldsymbol{x}$ with parameter $\boldsymbol{w} \in \mathbb{R}^k$.
- $\nabla \ell(\boldsymbol{w}_t) := \nabla_{\boldsymbol{w}_t} \ell(\boldsymbol{w}_t; \boldsymbol{x}) \in \mathbb{R}^k$: Gradient with respect to the weights $\boldsymbol{w}_t$ of loss ℓ for input $\boldsymbol{x}_t$ of time step t. A subscript is used to denote the time step t. We denote the time step t with a subscript. $\nabla_{\boldsymbol{w}_t} \ell(\boldsymbol{w}_t; X)$ and $\nabla_{\boldsymbol{w}_t} \ell(\boldsymbol{w}_t; B)$ can also be represented in this manner.
- $\rho \in \mathbb{R}$: A positive scalar value of the perturbation radius.
- $\boldsymbol{v}_t := \frac{\nabla \ell(\boldsymbol{w}_t)}{\|\nabla \ell(\boldsymbol{w}_t)\|} \in \mathbb{R}^k$: A normalized vector with $\|\boldsymbol{v}_t\|_2 = 1$ that indicates the loss-maximization direction at step t.
- $\boldsymbol{w}_t^p := \boldsymbol{w}_t + \rho \boldsymbol{v}_t$: Perturbed weight along the direction of $\boldsymbol{v}_t$ at step t.

3.2. Time series modeling

The objective of time series classification is to classify the time series data $\hat{\boldsymbol{y}} = f(\boldsymbol{x}_{1:L})$ into the correct one-hot label $\boldsymbol{y}$. By contrast, the objective of time series forecasting is to predict and minimize the error of unseen target variables $\boldsymbol{z}$. The model forecasts $\hat{\boldsymbol{z}}_{p+1:p+k} = f(\boldsymbol{z}_{p-k':p}, \boldsymbol{x}_{p-k':p})$, where k and k' are the look-forward and look-back windows of a model, respectively.

3.3. Overview of sharpness-aware minimization (SAM)

SAM [12] is a state-of-the-art optimization method that seeks flat minima to improve generalization performance. Given a radius ρ, the objective of SAM is minimizing sharpness in addition

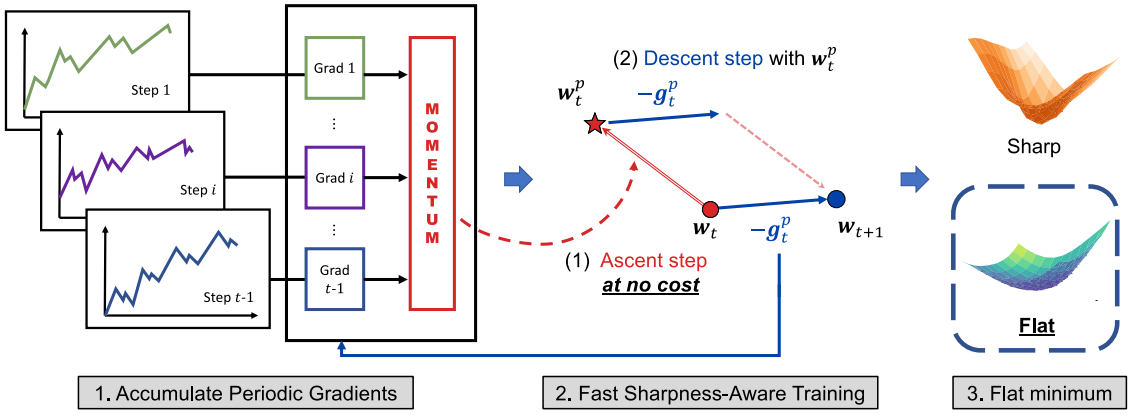


Fig. 4. The overview of the proposed framework. By utilizing accumulated gradients with periodic information, the proposed method can approximate SAM's ascent direction without the need for gradient computation. As a result, we can achieve flat minima without incurring any additional computational overhead.

to loss by solving the following problem:

$$\min_{\boldsymbol{w}} \max_{\|\boldsymbol{v}\|=1} \ell(\boldsymbol{w}+\rho\boldsymbol{v}). \tag{1}$$

Foret et al. [12] first approximated the inner-maximization problem in Eq. (1) using first-order Taylor expansion with radius size of ρ. At time t, the inner step can be formulated as follows:

$$\boldsymbol{w}_t^p = \boldsymbol{w}_t + \rho \frac{\nabla \ell(\boldsymbol{w}_t)}{\|\nabla \ell(\boldsymbol{w}_t)\|}. \tag{2}$$

For outer-minimization of *perturbed loss* $\boldsymbol{w}_t^p$ in Eq. (2), $\boldsymbol{w}_t$ is updated toward $\nabla \ell(\boldsymbol{w}_t^p)$ as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla \ell(\boldsymbol{w}_t^p). \tag{3}$$

For convenience, we denote Eq. (2) as the *ascent step* and Eq. (3) as the *descent step*. By minimizing both loss $\ell(\boldsymbol{w})$ and *sharpness* $\ell(\boldsymbol{w}_t^p) - \ell(\boldsymbol{w})$, SAM successfully obtains well-generalizing minima.

While SAM has its benefits, it also has two main issues that must be considered. First, SAM requires **twice** (2×) computational costs compared to vanilla optimizers (e.g. SGD and Adam) for propagating each gradient in the ascent and descent steps, which may hinder the quick real-time assessment. The second issue is that identifying the best direction for the ascent step is still unknown because achieving higher loss in the inner maximization does not guarantee better generalization [41,42].

To address these issues, several papers have proposed selecting a better-ascending direction [16,43] or investigating the learning dynamics of SAM [44] to minimize the generalization gap. Moreover, various studies [17,18,45] have attempted to reduce training time by approximating or skipping the ascent direction. However, these methods still require more training time than SGD, even though it is less than SAM.

4. Methodology

4.1. Overall framework

Our goal is to achieve both generalization and computational efficiency in time series modeling. To accomplish this goal, identifying flat minima is essential for accurately capturing robust trend and seasonal components. Therefore, we will utilize sharpness-aware training to make the loss flatter while addressing the main limitation of SAM, i.e., the computational burden of calculating $\boldsymbol{w}_p$. To overcome this drawback, we propose a new algorithm called *Periodic Sharpness-Aware Time series Training (PSATT)*. A brief overview of the proposed method can be found in Fig. 4. This method approximates the ascent direction using the periodic momentum of time series data in the ascent step. By eliminating the need for additional forward and backward steps, the proposed method only requires the same computational time as vanilla optimizers. A comparison of the vanilla optimizer (SGD or Adam), SAM, and the proposed method is provided in Table 1, where the proposed method achieves both flat minima and computational efficiency.

Table 1
A comparison of the sharpness of minima and computation burden among vanilla optimizer, SAM, and the proposed method. The number in parenthesis indicates the number of gradient calculation in each step.

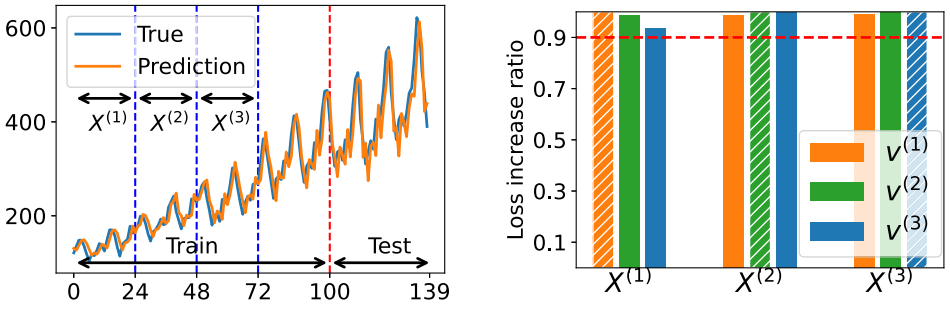
	Vanilla	SAM	Proposed
Converged minima	Sharp	Flat	**Flat**
Computational burden	Low (1×)	High (2×)	**Low (1×)**

In Section 4.2, we first analyze the impact of data periodicity on loss increase. In Section 4.3, we demonstrate that this periodicity induces batch-wise similarities in loss calculation, a phenomenon that is not usual in other domains except for periodic time series. Finally, we explain the details of the proposed algorithm in Section 4.4.

4.2. Toy example: Analyzing loss and periodicity in time series data

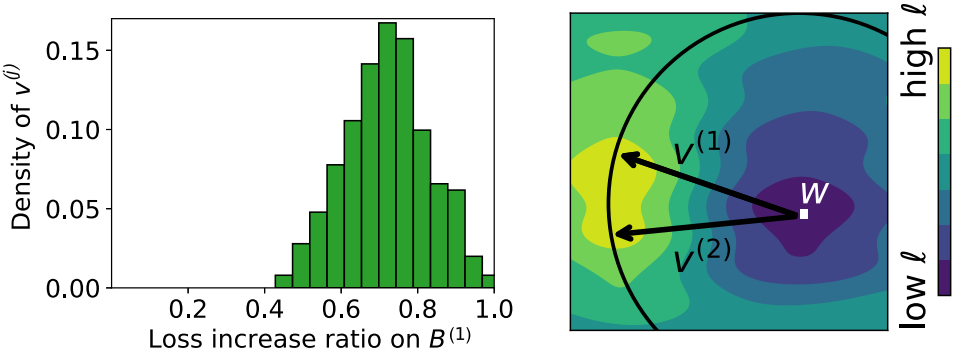
This subsection presents a toy example using monthly airline passenger data [46], as shown in Fig. 5. Our goal is to understand the influence of periodicity, a fundamental property of time series data, on loss increase and SAM's ascent direction. We train a simple one-layer LSTM model with a hidden size of 2 and a fully connected layer. After training, we choose three training periods $X^{(1)}, X^{(2)}$, and $X^{(3)}$ that exhibit similar seasonality in Fig. 5(a).

Given that all periods have similar trend and seasonality, we hypothesize that we could utilize information from one period in another. Therefore, we calculate the loss increase of each period using the gradients of different periods. We first calculate the ascent directions with respect to different periods $\boldsymbol{v}^{(j)} := \frac{\nabla \ell(\boldsymbol{w};X^{(j)})}{\|\nabla \ell(\boldsymbol{w};X^{(j)})\|}$ for $j \in \{1,2,3\}$ using the trained weight $\boldsymbol{w}$. Then, we measure the perturbed loss using different periods by calculating each $\boldsymbol{w}+\rho\boldsymbol{v}^{(j)}$. Finally, we show the effectiveness of the perturbed weights for different periods. In Fig. 5(b), we measure the loss increase ratio along different ascent directions $\frac{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(j)};X^{(i)})-\ell(\boldsymbol{w};X^{(i)})}{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(i)};X^{(i)})-\ell(\boldsymbol{w};X^{(i)})}$ for $(i,j) \in \{1,2,3\}^2$ and $\rho = 0.01$. The results indicate that the



(a) Airline passenger data and model's prediction (b) Loss increase ratio

Fig. 5. Toy experiment on airline passenger data. (a) True data and prediction of the trained model. Three periods $X^{(1)}, X^{(2)}$, and $X^{(3)}$ show similar trend and seasonality. (b) Loss increase ratio $\frac{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(j)};X^{(i)})-\ell(\boldsymbol{w};X^{(i)})}{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(i)};X^{(i)})-\ell(\boldsymbol{w};X^{(i)})}$ on $(i,j) \in \{1,2,3\}^2$. The results of using other periods also show over 90% loss increase compared to the hatched bars ($i = j$).



(a) Loss increase ratio of ETTm1 (b) Illustration of the loss landscape with two loss-increasing direction

Fig. 6. (a) Loss increase ratio $\frac{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(j)};B^{(1)})-\ell(\boldsymbol{w};B^{(1)})}{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(1)};B^{(1)})-\ell(\boldsymbol{w};B^{(1)})}$ of ETTm1. Even in batch learning, all the $\boldsymbol{v}^{(j)}$ for $j \in \{2,\dots,N\}$ effectively increase the loss of $B^{(1)}$. (b) Illustration of the loss landscape with two loss-increasing directions, $\boldsymbol{v}^{(1)}$ and $\boldsymbol{v}^{(2)}$. Both directions show a similar loss increase on $\boldsymbol{w}$ in a given radius.

loss increase is highly affected by periodicity regardless of the ascending direction $\boldsymbol{v}^{(j)}$, for $i \neq j$. Furthermore, the loss increase using the gradient of a different period is almost similar (over 90%) to the loss increase using its gradient. Thus, even though all periods have different numerical values, we conclude that trend and seasonality have a strong influence on training.

4.3. Batch-wise periodicity and loss increase

To push further into the impact of periodicity on time series data, we now explore mini-batch training scenarios. Generally, it is not possible to use information from different mini-batches to approximate SAM's ascent step because the loss $\ell(\boldsymbol{w}; B)$ highly fluctuates depending on mini-batch B in other domains [43,45], such as vision or language data. Thus, we now measure batch-wise loss increase as $\boldsymbol{v}^{(j)} := \frac{\nabla\ell(\boldsymbol{w};B^{(j)})}{\|\nabla\ell(\boldsymbol{w};B^{(j)})\|}$, which is similar to the analysis in Section 4.2 but using random batches B instead of fixed periods X. For modeling, we train Pyraformer [8] on the ETTm1 dataset [5] with a radius of $\rho = 0.1$. Using the trained weights $\boldsymbol{w}$, we measure the batch-wise loss increase to a fixed mini-batch $B^{(1)}$ by utilizing the ascent directions of other batches.

The results of the loss increase ratio $\frac{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(j)};B^{(1)})-\ell(\boldsymbol{w};B^{(1)})}{\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(1)};B^{(1)})-\ell(\boldsymbol{w};B^{(1)})}$ are illustrated in Fig. 6(a). All ratios are positive, suggesting that gradients from different batches can increase the loss on $B^{(1)}$ and approximate the ascent direction $\boldsymbol{v}^{(1)}$ during sharpness-aware training. Specifically, other batch gradient directions show an average loss increase of over 75% compared to $\boldsymbol{v}^{(1)}$. Note that even when directions are not aligned, we see a similar loss increase as illustrated in Fig. 6(b). This observation can be summarized in the time series domain as follows:

$$\begin{aligned}&\rho\nabla\ell(\boldsymbol{w};B^{(1)})^T\mathbb{E}_{j\in\{1,\dots,N\}}[\boldsymbol{v}^{(j)}]\\&=\mathbb{E}_{j\in\{1,\dots,N\}}[\ell(\boldsymbol{w}+\rho\boldsymbol{v}^{(j)};B^{(1)})-\ell(\boldsymbol{w};B^{(1)})]>0.\end{aligned}\tag{4}$$

Therefore, $\mathbb{E}_{j\in1,\dots,N}[\boldsymbol{v}^{(j)}]$ can be viewed as the direction that increases the loss of $\boldsymbol{w}$. It is important to note again that the loss increase among different batches is small due to batch-wise disagreement in other domains.

4.4. Periodic sharpness-aware time series training (PSATT)

Based on the analysis in Section 4.3, we alter the ascent direction of SAM by considering gradients computed from multiple batches. We first employ Eq. (4) to approximate the inner-maximization as follows:

$$\min_{\boldsymbol{w}}\max_{\|\boldsymbol{v}\|=1}\ell(\boldsymbol{w}+\rho\mathbb{E}_{j\in\{1,\dots,N\}}[\boldsymbol{v}^{(j)}]).\tag{5}$$

To reduce the computational burden of the expectation $\mathbb{E}_{j\in\{1,\dots,N\}}[\boldsymbol{v}^{(j)}]$ at each iteration t, we suggest leveraging previous

gradient information for calculating the ascent step. Fortunately, at every step t, we already possess the gradients of various mini-batches $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{t-1}\}$ from prior training steps. As current iterations contain gradient information to current step than old gradients, we introduce momentum to establish the ascent direction at step $t > 1$ as follows:

$$\boldsymbol{m}_t = \boldsymbol{g}_t + \beta \boldsymbol{m}_{t-1}, \tag{6}$$

where $\boldsymbol{m}_1 = \boldsymbol{g}_1$ and $\beta < 1$. As the momentum method assigns exponentially decreasing weights over time, the ascent momentum of Eq. (6) can effectively accumulate gradient information and achieve better convergence. Thus, we reformulate Eq. (6) as follows:

$$\boldsymbol{m}_t = \boldsymbol{g}_t + \beta \boldsymbol{g}_{t-1} + \beta^2 \boldsymbol{g}_{t-2} + \cdots + \beta^{t-1} \boldsymbol{g}_1. \tag{7}$$

Building on this intuition, we propose a novel optimization method called *Periodic Sharpness-Aware Time series Training (PSATT)* for finding flat minima in time series data. This method uses a weighted sum of diverse previous batches to approximate the ascent step $\boldsymbol{v}_t^{PSATT}$ by utilizing the previous ascent momentum $\boldsymbol{m}_{t-1}$, without requiring additional gradient calculation at step t:

$$\boldsymbol{v}_t^{PSATT} = \frac{\boldsymbol{m}_{t-1}}{\|\boldsymbol{m}_{t-1}\|} = \frac{\boldsymbol{g}_{t-1} + \beta \boldsymbol{g}_{t-2} + \cdots + \beta^{t-2} \boldsymbol{g}_1}{\|\boldsymbol{g}_{t-1} + \beta \boldsymbol{g}_{t-2} + \cdots + \beta^{t-2} \boldsymbol{g}_1\|}. \tag{8}$$

Then, the perturbed weight $\boldsymbol{w}_t^p$ of PSATT can be calculated as follows:

$$\boldsymbol{w}_t^p = \boldsymbol{w}_t + \rho \boldsymbol{v}_t^{PSATT} = \boldsymbol{w}_t + \rho \frac{\boldsymbol{g}_{t-1} + \beta \boldsymbol{g}_{t-2} + \cdots + \beta^{t-2} \boldsymbol{g}_1}{\|\boldsymbol{g}_{t-1} + \beta \boldsymbol{g}_{t-2} + \cdots + \beta^{t-2} \boldsymbol{g}_1\|}. \tag{9}$$

The full algorithm of the proposed PSATT using Eq. (9) is shown in Algorithm 1.

Algorithm 1: Periodic Sharpness-Aware Time series Training

Input: Initial weight $\boldsymbol{w}_1$, learning rate η, radius ρ, momentum coefficient β, and small ε to prevent zero division.
Output: Flat weight $\boldsymbol{w}_{T+1}$.
Initialize: Ascent momentum $\boldsymbol{m}_0 = \boldsymbol{0}$.
for $t = 1$ *to* T **do**
  Sample a mini-batch $B^{(t)}$.
  Compute ascent direction at no cost $\boldsymbol{v}_t^{PSATT} \leftarrow \boldsymbol{m}_{t-1}/(\|\boldsymbol{m}_{t-1}\| + \varepsilon)$.
  Compute descent gradient $\boldsymbol{g}_t \leftarrow \nabla \ell(\boldsymbol{w}_t + \rho \boldsymbol{v}_t^{PSATT}; B^{(t)})$.
  Update ascent momentum with periodicity $\boldsymbol{m}_t \leftarrow \boldsymbol{g}_t + \beta \boldsymbol{m}_{t-1}$.
  Update weights $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \boldsymbol{g}_t$.
end

Please refer to Appendix B for the specific update algorithms of vanilla and SAM. We now prove that the proposed method can approximate the perturbed loss of SAM under high batch-wise similarity.

Theorem 1. *Let a continuous loss function $\ell(\cdot)$ is a differentiable Lipschitz function with L. Given $\boldsymbol{v}_t = \frac{\nabla \ell(\boldsymbol{w}_t; B^{(t)})}{\|\nabla \ell(\boldsymbol{w}_t; B^{(t)})\|}$ and $\boldsymbol{v}_t^{PSATT}$ in Eq. (8), the difference between the perturbed losses using $\boldsymbol{v}_t$ and $\boldsymbol{v}_t^{PSATT}$ can be bounded:*

$$\left|\ell(\boldsymbol{w}_t + \rho \boldsymbol{v}_t^{PSATT}; B^{(t)}) - \ell(\boldsymbol{w}_t + \rho \boldsymbol{v}_t; B^{(t)})\right| < 2\rho L(1 + \epsilon), \tag{10}$$

where ρ is the radius in the parameter space and sufficiently small $\epsilon > 0$. Higher batch-wise similarity can guarantee lower ϵ.

Please refer to Appendix A for the proof. Note that we can simply calculate the ascent direction $\boldsymbol{m}_t = \frac{1}{\eta_t}(\boldsymbol{w}_{t-1} - \boldsymbol{w}_t)$ and $\boldsymbol{v}_t^{PSATT} = \frac{\boldsymbol{w}_{t-1} - \boldsymbol{w}_t}{\|\boldsymbol{w}_{t-1} - \boldsymbol{w}_t\|}$ by setting the ascent momentum of Eq. (7) equal to the momentum of the base optimizer.

4.5. Computational efficiency of PSATT

We provide a naive analysis of the complexity in a single iteration to explain the computational difference among vanilla, SAM, and PSATT. As mentioned earlier, computational efficiency is a significant limitation of sharpness-aware training in time series models. Each forward and backward propagation requires $O(ndw)$ operations for training one iteration in vanilla optimizers, given n data examples with d-dimensional features and the number of total weights w in the network [47]. SAM requires $O(ndw)$ operations for both the ascent and descent steps, resulting in doubled computations in a single iteration compared to vanilla optimizers. In contrast, because reusing the information of previous model parameters $\boldsymbol{m}_{t-1}$ only needs $O(w)$ operations, PSATT requires negligible additional operations with respect to vanilla training.

5. Experiments and analysis

We now demonstrate the advantages of the proposed method in terms of performance and time complexity. We consider both classification and forecasting tasks to verify the effectiveness.

5.1. Experimental setup

Baselines. For the baselines, we choose vanilla optimizer, SAM, and various variants of SAM: ASAM [16], LookSAM [18], and ESAM [17]. ASAM is designed to find better ascent direction with adaptive sharpness requiring the same calculation as SAM. For computational efficiency, LookSAM calculates the ascent gradient only once out of several iterations and ESAM selects subsets and reduces computations during both the ascent and descent steps. We follow the best settings mentioned in the papers.

Classification. For multivariate classification, we conduct experiments on the UCR (University of California, Riverside) time series classification datasets [48] and evaluate the accuracy. For the architecture, we use a CNN model with three convolution layers with a kernel size of [8,5,3] with batch normalization and Rectified Linear Units (ReLU) in every layer. We select the hyperparameters of training via grid searches. As a result, all models are trained with 100 epochs with batch size 128 and Adam optimizer with momentum $\beta_1 = 0.9$, $\beta_2 = 0.99$ by decaying the learning rate with cosine learning rate annealing from 0.1 initially. We select the best radius ρ from the set {0.001, 0.01, 0.1, 1, 2, 3, 5}.

Forecasting. For multivariate forecasting, we use the Electricity Transformer Temperature (ETT) [5] datasets, i.e., ETTh1, ETTh2, and ETTm1, by varying the look-forward window size (forecast horizon) k. For the architectures, we select two state-of-the-art models: Transformer-based Pyraformer [8], and TCN-based SCINet [9]. Both models achieve the best performance in time series forecasting tasks using tree-based architectures to capture long-term dependencies. By default, we re-implement experiments using the official GitHub repository of the papers, following the hyperparameters used in the original papers or GitHub implementations.[1] We use Adam optimizer with momentum $\beta_1 = 0.9$, $\beta_2 = 0.99$ and report the hyperparameters of Pyraformer and SCINet in Table 2. We choose the radius ρ from the set {0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.4}.

All the experiments are implemented using PyTorch-based libraries [50,51] on NVIDIA TITAN V.

[1] https://github.com/alipay/Pyraformer [49], https://github.com/cure-lab/SCINet [9].

Table 2
Experimental settings of the forecasting models. We report the best batch size (bs), learning rate (lr), and radius ρ for the proposed PSATT. k and k' indicate look-forward and look-back window sizes, respectively.

Method		Pyraformer					SCINet				
Dataset	k	k'	bs	lr	epoch	ρ	k	bs	lr	epoch	ρ
ETTh1	168	168	32	1.00E−05	5	0.05	336	32	5.00E−04	100	0.01
	336	168	32	1.00E−05	5	0.1	336	512	1.00E−04	100	0.001
	720	336	32	1.00E−05	5	0.01	736	256	1.00E−05	100	0.01
ETTh2	168	168	32	1.00E−05	5	0.05	336	16	5.00E−05	100	0.2
	336	168	32	1.00E−05	5	0.2	336	128	5.00E−05	100	0.4
	720	336	32	1.00E−05	5	0.001	736	128	1.00E−05	100	0.2
ETTm1	96	384	32	1.00E−05	5	0.1	384	32	5.00E−05	100	0.01
	288	672	32	1.00E−05	5	0.1	672	32	1.00E−05	100	0.1
	672	672	16	1.00E−05	5	0.4	672	32	1.00E−05	100	0.4

Table 3
Classification accuracy and computational time to process a data example in seconds of vanilla, SAM, and PSATT on UCR time series datasets. We calculate the Gain and Time Overhead Ratio ($\frac{\Delta \text{Time}}{\text{Time}}$) of SAM and PSATT w.r.t. vanilla training. Please refer to Appendix C for the classification results of ASAM [16], ESAM [17], and LookSAM [18].

Datasets	Vanilla		SAM				PSATT			
	Acc (%)	Time (sec)	Acc (%)	Gain (%)	Time (sec)	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)	Acc (%)	Gain (%)	Time (sec)	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)
ACSF1	95.00	0.0778	96.00	1.05	0.1542	98.20	95.00	0.00	0.0792	1.80
ArrowHead	88.00	0.0456	89.71	1.94	0.0883	93.64	90.29	2.60	0.0508	11.40
BirdChicken	99.33	0.1445	100.00	0.67	0.2826	95.57	100.00	0.67	0.1533	6.09
CBF	96.67	0.0314	100.00	3.44	0.0587	86.94	100.00	3.44	0.0376	19.75
Chinatown	97.96	0.0250	99.13	1.19	0.0462	84.80	99.13	1.19	0.0272	8.80
CinCECGTorso	83.55	0.2172	84.35	0.96	0.4317	98.76	86.09	3.04	0.2216	2.03
CricketZ	79.49	0.0052	81.03	1.94	0.0101	94.23	81.79	2.89	0.0056	7.69
Crop	76.86	0.0001	77.61	0.98	0.0002	100.00	77.15	0.38	0.0001	0.00
DistalPhalanx-OutlineCorrect	81.88	0.0011	82.97	1.33	0.0021	90.91	82.97	1.33	0.0014	27.27
Earthquakes	66.30	0.0091	79.86	20.45	0.0178	95.60	79.14	19.37	0.0097	6.59
ECG5000	92.00	0.0020	94.71	2.95	0.0038	90.00	94.56	2.78	0.0024	20.00
ECGFiveDays	94.38	0.0431	100.00	5.95	0.0816	89.33	100.00	5.95	0.0515	19.49
EthanolLevel	83.40	0.0184	84.40	1.20	0.0365	98.37	84.80	1.68	0.0187	1.63
FaceAll	81.30	0.0017	82.66	1.67	0.0032	88.24	85.38	5.02	0.0020	17.65
FacesUCR	91.41	0.0047	93.37	2.14	0.0089	89.36	92.39	1.07	0.0057	21.28
FiftyWords	63.74	0.0039	65.49	2.75	0.0077	97.44	65.27	2.40	0.0044	12.82
Fish	96.00	0.0150	97.71	1.78	0.0293	95.33	97.14	1.19	0.0161	7.33
FreezerSmallTrain	86.07	0.0713	95.02	10.40	0.1391	95.09	93.47	8.60	0.0779	9.26
Ham	76.19	0.0246	82.86	8.75	0.0482	95.93	81.90	7.49	0.0262	6.50
Haptics	49.68	0.0401	52.92	6.52	0.0794	98.00	52.27	5.21	0.0413	2.99
HouseTwenty	97.48	0.2758	99.16	1.72	0.5479	98.66	99.16	1.72	0.2795	1.34
InsectWing-beatSound	41.41	0.0077	43.03	3.91	0.0150	94.81	42.88	3.55	0.0086	11.69
ItalyPower-Demand	96.70	0.0075	97.47	0.80	0.0139	85.33	97.28	0.60	0.0082	9.33
LargeKitchen-Appliances	91.20	0.0104	94.40	3.51	0.0205	97.12	92.80	1.75	0.0109	4.81
MiddlePhalanx-OutlineAgeGroup	66.88	0.0017	68.18	1.94	0.0031	82.35	68.18	1.94	0.0022	29.41
Average	82.92	0.0434	85.68	3.60	0.0852	93.36	85.56	3.44	0.0457	10.68

5.2. Classification results

Generalization performance. We first experimentally prove the effectiveness of the sharpness-ware training in time series classification. We report the best test accuracy and its corresponding time to process a data example in seconds, obtained by vanilla, SAM, and PSATT in Table 3. We also note *Gain*, the percentage improvement of accuracy. To verify the effectiveness of the proposed PSATT, we select 25 datasets that show the most improved accuracy with SAM compared to vanilla training. Similar to experiments in other classification tasks [12,43], the experimental results in Table 3 indicate that sharpness-aware training improves the classification accuracy in various time series data. Please refer to Appendix C for the classification results of ASAM, ESAM, and LookSAM. Overall, both SAM and PSATT show a performance gain of over 3% on average.

Computational overhead. In Table 3, we note the computational time of vanilla, SAM, and PSATT for processing one example while training and Time Overhead Ratio ($\frac{\Delta \text{Time}}{\text{Time}}$), the percentage overhead of computational time of SAM and PSATT compared to the vanilla optimizer. The experimental results show that our proposed PSATT improves the time efficiency of SAM. Specifically, the proposed method only requires 10.68% of the additional computation burden. By contrast, SAM needs 93.36% computation more than vanilla optimizer. Please refer to Appendix C for the classification times of ASAM, ESAM, and LookSAM. To summarize, the proposed PSATT shows the lowest computational overhead.

5.3. Forecasting results

We further conduct experiments on forecasting to evaluate the effectiveness of sharpness-aware training in regression tasks, particularly multivariate forecasting: test error in Table 4 and computational time in Table 5. The proposed method also achieves both performance and time efficiency in forecasting tasks.

Table 4
Forecasting performance of vanilla, SAM [12], ASAM [16], ESAM [17], LookSAM [18], and the proposed PSATT on ETTh1, ETTh2, and ETTm1 with Pyraformer [8] and SCI-Net [9]. We calculate the error reduction rate (ERR) w.r.t. vanilla training.

Network	Optimizer	Metric	ETTh1			ETTh2			ETTm1			Average
			168	336	720	168	336	720	96	288	672	
Pyraformer	Vanilla	MSE	0.7846	0.9153	0.9703	0.7704	0.9262	0.9792	0.5017	0.6960	0.8570	0.8223
		MAE	0.6822	0.7530	0.7727	0.6671	0.7548	0.7798	0.4980	0.6378	0.7070	0.6947
	SAM	MSE	0.7488	0.8697	0.9563	0.7519	0.8987	0.9726	0.4694	0.5989	0.6830	0.7721
		MAE	0.6599	0.7232	0.7782	0.6561	0.7364	0.7775	0.4867	0.5786	0.6390	0.6706
		ERR(%)	3.96	4.52	0.49	2.05	2.73	0.51	4.36	11.72	15.47	5.09
	ASAM	MSE	0.7920	0.8916	0.9432	0.7495	0.8578	0.9678	0.4851	0.6232	0.7180	0.7809
		MAE	0.6740	0.7358	0.7691	0.6637	0.7113	0.7821	0.4933	0.5932	0.6560	0.6754
		ERR(%)	0.05	2.45	1.76	1.69	6.66	0.52	2.12	8.80	12.15	4.02
	ESAM	MSE	0.7590	0.9191	0.9588	0.7760	0.9060	0.9363	0.5074	0.6020	0.7119	0.7863
		MAE	0.6672	0.7492	0.7791	0.6772	0.7343	0.7620	0.5122	0.5805	0.6309	0.6770
		ERR(%)	2.76	0.00	0.29	−1.09	2.42	3.45	−1.99	11.34	14.14	3.48
	LookSAM	MSE	0.7837	0.8976	0.9951	0.8179	0.9012	0.9812	0.5052	0.6174	0.6968	0.7996
		MAE	0.6620	0.7360	0.7877	0.6888	0.7311	0.7793	0.4976	0.5840	0.6389	0.6784
		ERR(%)	1.43	2.08	−2.29	−4.81	2.90	−0.09	−0.31	9.93	14.60	2.60
	PSATT	MSE	0.7562	0.8882	0.9519	0.7549	0.8658	0.9388	0.4950	0.5983	0.7111	0.7734
		MAE	0.6564	0.7334	0.7738	0.6540	0.7110	0.7656	0.4942	0.5799	0.6449	0.6681
		ERR(%)	3.69	2.80	0.99	1.99	6.20	3.10	1.05	11.67	13.30	4.98
SCI-Net	Vanilla	MSE	0.4515	0.5858[a]	0.5833	0.5051	0.6183	1.0749	0.1915	0.3654	1.1520[a]	0.6142
		MAE	0.4566	0.5478[a]	0.5607	0.5044	0.5605	0.7617	0.2912	0.4150	0.8162[a]	0.5460
	SAM	MSE	0.4354	0.5099	0.5741	0.4583	0.5586	1.0749	0.1889	0.2964	1.1396	0.5819
		MAE	0.4404	0.5017	0.5529	0.4797	0.5331	0.7617	0.2882	0.3541	0.8081	0.5245
		ERR(%)	3.56	10.76	1.49	7.08	7.39	0.00	1.16	16.65	1.04	5.45
	ASAM	MSE	0.4372	0.5169	0.5660	0.5303	0.6163	1.0749	0.1944	0.3476	0.7187	0.5558
		MAE	0.4423	0.5061	0.5483	0.5106	0.5593	0.7617	0.2923	0.4000	0.6126	0.5148
		ERR(%)	3.15	9.76	2.60	−3.11	0.27	0.00	−0.83	4.20	32.36	5.38
	ESAM	MSE	0.4688	0.5305	0.5815	0.3934	0.5277	1.0862	0.1904	0.2835	0.8961	0.5509
		MAE	0.4705	0.5143	0.5567	0.4291	0.5198	0.7684	0.2881	0.3450	0.7063	0.5109
		ERR(%)	−3.44	7.83	0.51	18.52	11.14	−0.98	0.87	19.46	18.59	8.06
	LookSAM	MSE	0.4653	0.5508	0.5820	0.4731	0.5654	1.0918	0.1963	0.3123	1.1004	0.5930
		MAE	0.4641	0.5275	0.5588	0.4854	0.5375	0.7693	0.2930	0.3715	0.8067	0.5349
		ERR(%)	−2.35	4.88	0.28	5.05	6.44	−1.33	−1.37	12.38	3.10	3.01
	PSATT	MSE	0.4435	0.5098	0.5831	0.4244	0.5166	1.0224	0.1804	0.2981	0.6819	0.5178
		MAE	0.4483	0.5017	0.5605	0.4571	0.5134	0.7448	0.2759	0.3583	0.5841	0.4938
		ERR(%)	1.79	10.77	0.03	12.68	12.62	3.78	5.47	15.89	35.68	10.97

[a]We cannot reproduce the reported results of SCINet on ETTh1 336 (MSE 0.502 and MAE 0.497) and ETTm1 672 (MSE 0.713 and MAE 0.604). However, SAM and PSATT show improvement in both experiments.

Generalization performance. In Table 4, we evaluate the performance of vanilla, SAM, ASAM, ESAM, LookSAM, and proposed PSATT for long-range forecasting with two state-of-the-art models: Pyraformer [8] and SCINet [9]. We calculate the *error reduction rate (ERR)*, the percentage reduction in error, of the sum of mean standard error (MSE) and mean absolute error (MAE), compared to the vanilla optimizer.

All sharpness-aware training methods improve MSE and MAE in all tasks, even though we set the base architectures as the state-of-the-art models in Transformer and TCN. For Pyraformer, both SAM and PSATT show about 5% of the overall ERR. For SCINet, SAM shows 5.45% of ERR, and PSATT further reduces the regression error with 10.97% on average. While ASAM achieves similar performance to SAM, both ESAM and LookSAM show lower performance than SAM.

To investigate how the sharpness-aware training affects forecasting, we plot the prediction value of variates from the ETTh1 dataset with vanilla and PSATT, in Fig. 7. Interestingly, the Transformer-based Pyraformer (Figs. 7(a) and 7(b)) shows smooth forecasting results. Simultaneously, TCN-based SCINet (Figs. 7(c) and 7(d)) predicts each data point more precisely. Although all figures have different variate and time steps, PSATT shows accurate regression to the ground truth by maximizing the advantages of each method.

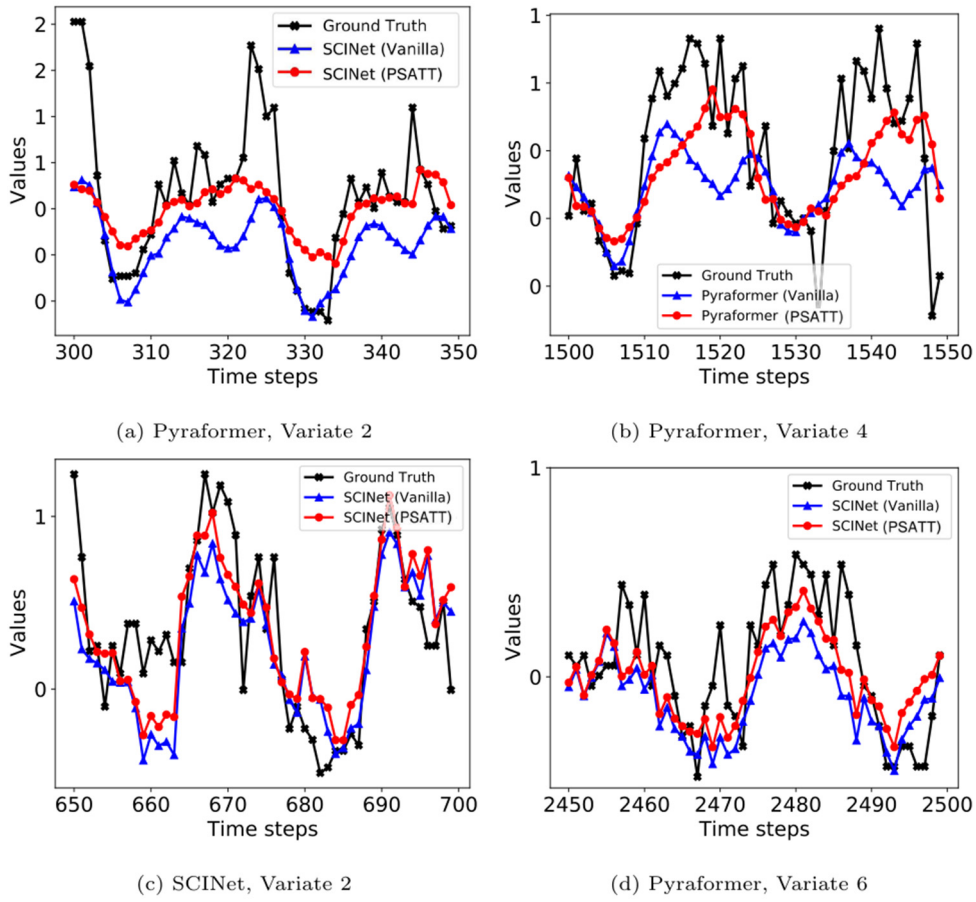
Computational overhead. Furthermore, Table 5 shows the computational time for processing one forecasting example in seconds using Pyraformer. Similar to the classification time in Table 3, SAM and ASAM require the twice computation as a vanilla optimizer. However, the proposed method only further requires 10.51% computation. Although ESAM and LookSAM are designed to increase the speed of SAM, they still have a computational burden of 66.81% and 46.34%, respectively. This is because both methods reduce the level of gradient calculation but cannot fully replace the gradient calculation of the ascent step.

5.4. Visualizing the local minima of the time series model

As the sharp minima can cause overfitting problems in time series data, we visualize the loss landscape of the trained neural networks of vanilla training and sharpness-aware training using filter normalization [38]. We perturb the trained weight into two random orthogonal directions and calculate the sharpness of each point in grids. The results are illustrated in Fig. 8 using CNN on classification and Pyraformer [8] on forecasting. In both tasks, the vanilla optimizer shows sharper minima than sharpness-aware training. Since the sharp local minima can be easily affected by noisy irregular components, sharpness-aware training helps robust prediction in time series data.

5.5. Sensitivity analysis to radius ρ

As the radius ρ is a hyperparameter, we now give the sensitivity analysis on selecting the best ρ. It is known that selecting proper ρ significantly affects the generalization performance [12]. Fig. 9 shows the difference in MSE between models trained with PSATT with given radius $\rho \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.4\}$ and vanilla, with Pyraformer (solid line) and SCINet (dash-dotted line) on three forecasting datasets. Lower MSE indicates a



(a) Pyraformer, Variate 2 (b) Pyraformer, Variate 4

(c) SCINet, Variate 2 (d) Pyraformer, Variate 6

Fig. 7. The prediction values of vanilla and PSATT on randomly-selected time steps and variates from the ETTh1 dataset.

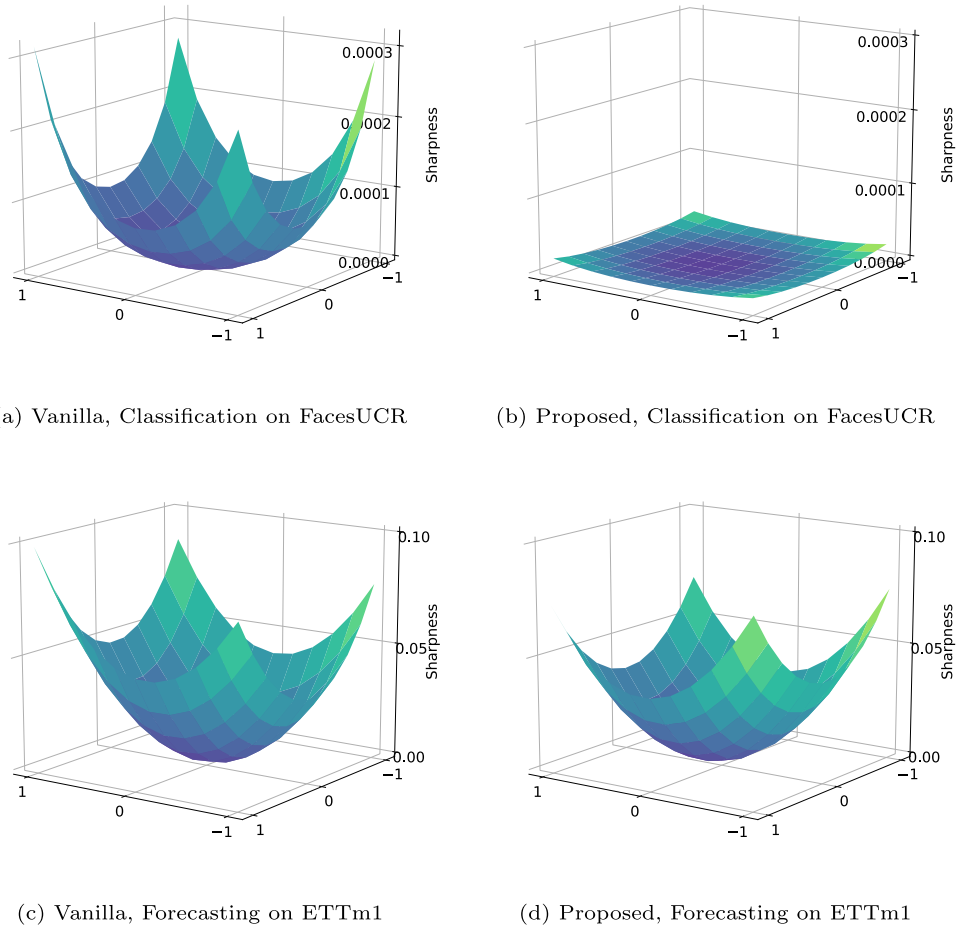
Table 5
Computational efficiency of forecasting to process a data example in milliseconds (ms) of vanilla, SAM [12], ASAM [16], ESAM [17], LookSAM [18], and the proposed PSATT. We also calculate the Time Overhead Ratio ($\frac{\Delta \text{Time}}{\text{Time}}$) w.r.t. vanilla training.

Optimizer	Metric	ETTh1			ETTh2			ETTm1			Average
		168	336	720	168	336	720	96	288	672	
Vanilla	Time (ms)	2.49	2.52	5.68	2.50	2.52	5.68	4.07	9.12	8.85	4.82
SAM	Time (ms)	4.94	4.99	11.35	4.96	5.00	11.34	8.05	18.15	17.57	9.59
	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)	98.59	98.02	99.89	98.44	98.41	99.58	98.11	99.07	98.62	98.75
ASAM	Time (ms)	4.81	5.00	11.38	4.94	5.01	11.35	7.92	17.85	17.57	9.54
	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)	93.59	98.15	100.44	97.99	98.78	99.87	94.83	94.83	94.83	97.04
ESAM	Time (ms)	4.08	4.24	9.63	4.37	4.64	9.52	6.40	12.89	14.08	7.76
	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)	64.21	68.04	69.64	75.00	84.23	67.62	57.53	57.53	57.53	66.81
LookSAM	Time (ms)	3.31	3.63	8.34	3.96	4.03	8.26	6.08	12.21	12.89	6.97
	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)	33.17	43.82	46.88	58.40	60.11	45.42	49.62	33.91	45.77	46.34
PSATT	Time (ms)	2.86	2.88	6.09	2.87	2.90	6.08	4.42	9.48	9.58	5.24
	$\frac{\Delta \text{Time}}{\text{Time}}$ (%)	14.96	14.32	7.17	14.94	15.04	7.08	8.81	3.99	8.26	10.51

better forecasting model in Fig. 9. ETTh1 ($k = 336$) shows lower MSE with a smaller radius ρ, while ETTm1 ($k = 288$) performs better on large ρ. In contrast, ETTh2 ($k = 336$) performs well under various radius ρ. Similar to other hyperparameters, such as batch size and learning rate, the best radius ρ should be searched depending on look-forward window k.

6. Conclusion

In this study, we explore the effectiveness of sharpness-aware training for time series data. We demonstrate that time series models are prone to overfitting and can become trapped in sharp local minima. To overcome this challenge, we introduce Sharpness-Aware Minimization, which has the drawback of requiring twice the computational time. To address this issue, we propose a sharpness-aware training method that incorporates the periodic characteristics of time series data and results in negligible additional computational overhead compared to vanilla optimizers. Our experimental results show that our proposed method improves the performance of time series data while maintaining computational efficiency. However, our work has



(a) Vanilla, Classification on FacesUCR (b) Proposed, Classification on FacesUCR

(c) Vanilla, Forecasting on ETTm1 (d) Proposed, Forecasting on ETTm1

Fig. 8. Landscape of sharpness in two random directions with vanilla and the proposed method on classification and forecasting models.

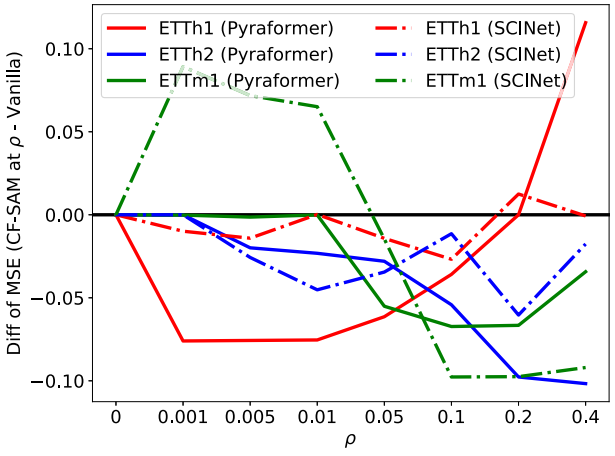


Fig. 9. Sensitivity analysis to radius ρ of PSATT with Pyraformer (solid line) and SCINet (dash-dotted line) on ETTh1, ETTh2, and ETTm1. We plot the difference of MSE between models with PSATT at given ρ and vanilla (lower is better).

the limitation of not providing an adaptive way to select the best radius ρ, which can be influenced by the dataset and look-forward window size. In future work, we plan to develop an adaptive method for selecting radius ρ during training and online learning situations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have cited the sources for the datasets and they are publicly available.

Acknowledgement

This work was partly supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation) and the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) (No. 2019R1A2C2002358; No. 2022R1A5A6000840; No. 2022R1F1A1074393).

Table C.6
Classification accuracy and computational time to process a data example in seconds of ASAM, ESAM, and LookSAM on UCR time series datasets. We calculate the Gain and Time Overhead Ratio ($\frac{\Delta\text{Time}}{\text{Time}}$) of SAM and PSATT w.r.t. vanilla training in Table 3.

Datasets	ASAM				ESAM				LookSAM			
	Acc (%)	Gain (%)	Time (sec)	$\frac{\Delta\text{Time}}{\text{Time}}$ (%)	Acc (%)	Gain (%)	Time (sec)	$\frac{\Delta\text{Time}}{\text{Time}}$ (%)	Acc (%)	Gain (%)	Time (sec)	$\frac{\Delta\text{Time}}{\text{Time}}$ (%)
ACSF1	96.00	1.00	0.1668	114.42	95.00	0.00	0.1035	33.02	95.00	0.00	0.0975	25.32
ArrowHead	90.86	2.86	0.1059	132.32	88.00	0.00	0.0914	100.50	88.57	0.57	0.0774	69.66
BirdChicken	100.00	0.67	0.2754	90.55	95.00	−4.33	0.2731	88.98	100.00	0.67	0.2296	58.87
CBF	98.33	1.66	0.0619	97.25	96.67	0.00	0.0668	112.62	98.33	1.66	0.0513	63.53
Chinatown	99.13	1.17	0.0508	103.21	99.13	1.17	0.0551	120.47	99.13	1.17	0.0413	65.06
CinCECGTorso	81.88	−1.67	0.4499	107.14	81.01	−2.54	0.2634	21.29	81.01	−2.54	0.2729	25.63
CricketZ	80.51	1.02	0.0123	136.59	78.97	−0.52	0.0118	126.82	80.26	0.77	0.0096	85.04
Crop	77.86	1.00	0.0002	108.95	77.12	0.26	0.0002	126.97	77.26	0.40	0.0002	62.33
DistalPhalanx-OutlineCorrect	82.25	0.37	0.0022	100.85	81.88	0.00	0.0024	118.51	82.25	0.37	0.0017	53.15
Earthquakes	92.00	25.70	0.0172	88.95	94.00	27.70	0.0148	62.52	94.00	27.70	0.0143	56.64
ECG5000	100.00	8.00	0.0040	100.29	100.00	8.00	0.0043	114.08	100.00	8.00	0.0031	57.39
EthanolLevel	80.40	−3.00	0.0345	87.45	78.20	−5.20	0.0216	17.46	78.20	−5.20	0.0230	24.78
FaceAll	86.21	4.91	0.0034	97.11	86.98	5.68	0.0037	118.97	83.14	1.84	0.0026	53.31
FacesUCR	93.22	1.81	0.0092	96.70	92.34	0.93	0.0096	104.84	92.63	1.22	0.0072	54.20
FiftyWords	64.18	0.44	0.0077	96.27	64.62	0.88	0.0084	115.37	64.62	0.88	0.0066	68.17
Fish	97.14	1.14	0.0309	105.73	97.14	1.14	0.0265	76.89	97.14	1.14	0.0247	64.63
FreezerSmallTrain	94.00	7.93	0.1523	113.61	84.11	−1.96	0.1513	112.27	90.81	4.74	0.1228	72.22
Ham	79.05	2.86	0.0518	110.68	81.90	5.71	0.0465	89.13	79.05	2.86	0.0409	66.28
Haptics	53.90	4.22	0.0813	102.82	50.97	1.29	0.0534	33.11	51.62	1.94	0.0533	32.90
HouseTwenty	99.16	1.68	0.6157	123.24	99.16	1.68	0.3215	16.57	97.48	0.00	0.3361	21.88
InsectWing-beatSound	43.33	1.92	0.0145	88.03	42.73	1.32	0.0159	106.74	42.73	1.32	0.0125	62.65
ItalyPower-Demand	97.47	0.77	0.0130	73.42	96.99	0.29	0.0148	97.04	96.89	0.19	0.0111	48.48
LargeKitchen-Appliances	92.80	1.60	0.0185	77.79	93.33	2.13	0.0152	46.08	91.47	0.27	0.0150	43.98
MiddlePhalanx-OutlineAgeGroup	67.53	0.65	0.0032	85.87	67.53	0.65	0.0037	116.24	67.53	0.65	0.0027	60.50
Average	85.30	2.86	0.0909	101.64	84.28	1.85	0.0658	86.52	84.55	2.11	0.0607	54.03

Appendix A. Proof of Theorem 1

We here provide the proof of Theorem 1.

Proof. First of all, by the batch-wise similarity of the perturbed loss assumption, the equation can be transformed as follows:

$$
\begin{aligned}
&\left|\ell(\boldsymbol{w}_t+\rho\boldsymbol{v}_t^{PSATT};B^{(t)})-\ell(\boldsymbol{w}_t+\rho\boldsymbol{v}_t;B^{(t)})\right| \\
&\le \left|\ell(\boldsymbol{w}_t+\rho\boldsymbol{v}_t^{PSATT};B^{(t)})-\mathbb{E}_B\left[\ell(\boldsymbol{w}_t+\rho\frac{\nabla\ell(\boldsymbol{w}_t;B)}{\|\nabla\ell(\boldsymbol{w}_t;B)\|};B^{(t)})\right]\right| \\
&\quad+\left|\mathbb{E}_B\left[\ell(\boldsymbol{w}_t+\rho\frac{\nabla\ell(\boldsymbol{w}_t;B)}{\|\nabla\ell(\boldsymbol{w}_t;B)\|};B^{(t)})\right]-\ell(\boldsymbol{w}_t+\rho\boldsymbol{v}_t;B^{(t)})\right| \\
&=(1+\epsilon)\left|\ell(\boldsymbol{w}_t+\rho\boldsymbol{v}_t^{PSATT};B^{(t)})-\mathbb{E}_B\left[\ell(\boldsymbol{w}_t+\rho\frac{\nabla\ell(\boldsymbol{w}_t;B)}{\|\nabla\ell(\boldsymbol{w}_t;B)\|};B^{(t)})\right]\right| \\
&\qquad\qquad\because \textit{Batch-wise similarity} \\
&=(1+\epsilon)\left|\rho\nabla\ell(\boldsymbol{w}_t;B^{(t)})^T\left[\boldsymbol{v}_t^{PSATT}-\mathbb{E}_B\frac{\nabla\ell(\boldsymbol{w}_t;B)}{\|\nabla\ell(\boldsymbol{w}_t;B)\|}\right]\right| \\
&\quad\because \textit{Taylor expansion} \\
&\le(1+\epsilon)\rho L\left|\boldsymbol{v}_t^{PSATT}-\mathbb{E}_B\frac{\nabla\ell(\boldsymbol{w}_t;B)}{\|\nabla\ell(\boldsymbol{w}_t;B)\|}\right| \quad\because \textit{Lipschitz condition} \\
&\le 2\rho L(1+\epsilon). \quad\square
\end{aligned}
$$

Appendix B. Algorithms of vanilla training and SAM

We illustrate the algorithms of vanilla training and SAM in Algorithm 2 similar to Algorithm 1. Because the gradient calculation $\nabla\ell(\boldsymbol{w})$ takes dominant time in training, SAM requires twice the computations than vanilla or the proposed PSATT.

Algorithm 2: Vanilla and SAM

Input: Initial parameter $\boldsymbol{w}_1$, learning rate η, radius ρ, momentum coefficient β, and small ε to prevent zero division.
Output: Final parameter $\boldsymbol{w}_{T+1}$.
for $t = 1$ *to* T **do**
  Sample a mini-batch $B^{(t)}$.
  if *Vanilla* **then**
    Update weights $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta\nabla\ell(\boldsymbol{w}_t;B^{(t)})$.
  if *SAM* **then**
    Compute ascent direction $\boldsymbol{v}_t^{SAM} \leftarrow \frac{\nabla\ell(\boldsymbol{w}_t)}{\|\nabla\ell(\boldsymbol{w}_t)\|+\varepsilon}$.
      `// Computational burden of extra gradient calculation`
    Compute descent gradient $\boldsymbol{g}_t \leftarrow \nabla\ell(\boldsymbol{w}_t+\rho\boldsymbol{v}_t^{SAM};B^{(t)})$.
    Update weights $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta\boldsymbol{g}_t$.
end

Appendix C. Classification results of variants of SAM

We report the classification results of variants of SAM in Table C.6. The experimental settings are equal to Table 3. The variants of SAM also improved performance, indicating the usefulness of sharpness-aware training. Note that ESAM is designed to boost the computational efficiency of SAM. However, due to small datasets and additional computations for calculating the difference in the loss in ESAM, it sometimes takes longer than SAM or ASAM.

References

[1] O.B. Sezer, M.U. Gudelek, A.M. Ozbayoglu, Financial time series forecasting with deep learning: A systematic literature review: 2005–2019, Appl. Soft Comput. 90 (2020) 106181.

[2] H. Jang, J. Lee, An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information, IEEE Access 6 (2017) 5427–5437.

[3] B. Son, J. Lee, Graph-based multi-factor asset pricing model, Finance Res. Lett. 44 (2022) 102032.

[4] H. Jang, J. Lee, Generative Bayesian neural network model for risk-neutral pricing of American index options, Quant. Finance 19 (4) (2019) 587–603.

[5] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 12, 2021, pp. 11106–11115.

[6] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, Adv. Neural Inf. Process. Syst. 33 (2020) 17804–17815.

[7] B. Lim, S. Zohren, Time-series forecasting with deep learning: a survey, Phil. Trans. R. Soc. A 379 (2194) (2021) 20200209.

[8] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A.X. Liu, S. Dustdar, Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting, in: International Conference on Learning Representations, 2021.

[9] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, Q. Xu, SCINet: Time series modeling and forecasting with sample convolution and interaction, in: Thirty-Sixth Conference on Neural Information Processing Systems, NeurIPS, 2022, 2022.

[10] C. Xing, D. Arpit, C. Tsirigotis, Y. Bengio, A walk with SGD, 2018, arXiv preprint arXiv:1802.08770.

[11] N.S. Keskar, J. Nocedal, P.T.P. Tang, D. Mudigere, M. Smelyanskiy, On large-batch training for deep learning: Generalization gap and sharp minima, in: International Conference on Learning Representations, 2017.

[12] P. Foret, A. Kleiner, H. Mobahi, B. Neyshabur, Sharpness-aware minimization for efficiently improving generalization, in: International Conference on Learning Representations, 2020.

[13] X. Chen, C.-J. Hsieh, B. Gong, When vision transformers outperform ResNets without pre-training or strong data augmentations, in: International Conference on Learning Representations, 2022.

[14] Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, Z. Lu, Generalized federated learning via sharpness aware minimization, in: International Conference on Machine Learning, PMLR, 2022.

[15] D. Bahri, H. Mobahi, Y. Tay, Sharpness-aware minimization improves language model generalization, 2021, arXiv preprint arXiv:2110.08529.

[16] J. Kwon, J. Kim, H. Park, I.K. Choi, Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks, in: International Conference on Machine Learning, PMLR, 2021, pp. 5905–5914.

[17] J. Du, H. Yan, J. Feng, J.T. Zhou, L. Zhen, R.S.M. Goh, V. Tan, Efficient sharpness-aware minimization for improved training of neural networks, in: International Conference on Learning Representations, 2022.

[18] Y. Liu, S. Mai, X. Chen, C.-J. Hsieh, Y. You, Towards efficient and scalable sharpness-aware minimization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12360–12370.

[19] R.B. Cleveland, W.S. Cleveland, J.E. McRae, I. Terpenning, STL: A seasonal-trend decomposition, J. Off. Stat 6 (1) (1990) 3–73.

[20] J.D. Cryer, K.-S. Chan, Time Series Analysis: With Applications in R, Vol. 2, Springer, 2008.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[22] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, P. Loos, Time series classification using deep learning for process planning: A case from the process industry, Procedia Comput. Sci. 114 (2017) 242–249.

[23] J. Azar, A. Makhoul, R. Couturier, J. Demerjian, Robust IoT time series classification with data compression and deep learning, Neurocomputing 398 (2020) 222–234.

[24] J. Chung, S. Ahn, Y. Bengio, Hierarchical multiscale recurrent neural networks, in: International Conference on Learning Representations, 2017.

[25] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: Probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191.

[26] J.F. Kolen, S.C. Kremer, Gradient flow in recurrent nets: The difficulty of learning LongTerm dependencies, in: A Field Guide to Dynamical Recurrent Networks, 2001, pp. 237–243.

[27] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, G.W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 2627–2633.

[28] B. Zhao, H. Lu, S. Chen, J. Liu, D. Wu, Convolutional neural networks for time series classification, J. Syst. Eng. Electron. 28 (1) (2017) 162–169.

[29] C. Ji, M. Du, Y. Hu, S. Liu, L. Pan, X. Zheng, Time series classification based on temporal features, Appl. Soft Comput. 128 (2022) 109494.

[30] I. Koprinska, D. Wu, Z. Wang, Convolutional neural networks for energy time series forecasting, in: 2018 International Joint Conference on Neural Networks, IJCNN, IEEE, 2018, pp. 1–8.

[31] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, arXiv preprint arXiv:1803.01271.

[32] M. Thill, W. Konen, H. Wang, T. Bäck, Temporal convolutional autoencoder for unsupervised anomaly detection in time series, Appl. Soft Comput. 112 (2021) 107751.

[33] N. Kitaev, L. Kaiser, A. Levskaya, Reformer: The efficient transformer, in: International Conference on Learning Representations, 2019.

[34] R. Bala, R.P. Singh, et al., A dual-stage advanced deep learning algorithm for long-term and long-sequence prediction for multivariate financial time series, Appl. Soft Comput. 126 (2022) 109317.

[35] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, PMLR, 2022, pp. 27268–27286.

[36] Y. Zhang, J. Yan, Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting, in: The Eleventh International Conference on Learning Representations, 2023.

[37] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning requires rethinking generalization, in: International Conference on Learning Representations, 2017.

[38] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, Visualizing the loss landscape of neural nets, Adv. Neural Inf. Process. Syst. 31 (2018).

[39] S. Lee, W. Lee, J. Park, J. Lee, Towards better understanding of training certifiably robust models against adversarial examples, Adv. Neural Inf. Process. Syst. 34 (2021) 953–964.

[40] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? Adv. Neural Inf. Process. Syst. 31 (2018).

[41] M. Andriushchenko, N. Flammarion, Towards understanding sharpness-aware minimization, in: International Conference on Machine Learning, PMLR, 2022, pp. 639–668.

[42] H. Kim, J. Park, Y. Choi, W. Lee, J. Lee, Exploring the effect of multi-step ascent in sharpness-aware minimization, 2023, arXiv preprint arXiv:2302.10181.

[43] J. Zhuang, B. Gong, L. Yuan, Y. Cui, H. Adam, N.C. Dvornek, J. s Duncan, T. Liu, et al., Surrogate gap minimization improves sharpness-aware training, in: International Conference on Learning Representations, 2021.

[44] H. Kim, J. Park, Y. Choi, J. Lee, Stability analysis of sharpness-aware minimization, 2023, arXiv preprint arXiv:2301.06308.

[45] J. Du, Z. Daquan, J. Feng, V. Tan, J.T. Zhou, Sharpness-aware training for free, in: Advances in Neural Information Processing Systems, 2022.

[46] G.E. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, San Francisco, Calif.(USA) Holden-Day, 1976.

[47] C.M. Bishop, N.M. Nasrabadi, Pattern Recognition and Machine Learning, Vol. 4, Springer, 2006.

[48] H.A. Dau, A. Bagnall, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/CAA J. Autom. Sin. 6 (6) (2019) 1293–1305.

[49] F. Liu, B. Han, T. Liu, C. Gong, G. Niu, M. Zhou, M. Sugiyama, et al., Probabilistic margins for instance reweighting in adversarial training, Adv. Neural Inf. Process. Syst. 34 (2021).

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8026–8037.

[51] H. Kim, Torchattacks: A pytorch repository for adversarial attacks, 2020, arXiv preprint arXiv:2010.01950.