Dataset Condensation for Time Series Classification via Dual Domain Matching

Zhanyu Liu
Shanghai Jiao Tong University
Shanghai, China
zhyliu00@sjtu.edu.cn

Ke Hao
Shanghai Jiao Tong University
Shanghai, China
ke_hao2002@outlook.com

Guanjie Zheng*
Shanghai Jiao Tong University
Shanghai, China
gjzheng@sjtu.edu.cn

Yanwei Yu
Ocean University of China
Qingdao, China
yuyanwei@ouc.edu.cn

ABSTRACT

Time series data has been demonstrated to be crucial in various research fields. The management of large quantities of time series data presents challenges in terms of deep learning tasks, particularly for training a deep neural network. Recently, a technique named *Dataset Condensation* has emerged as a solution to this problem. This technique generates a smaller synthetic dataset that has comparable performance to the full real dataset in downstream tasks such as classification. However, previous methods are primarily designed for image and graph datasets, and directly adapting them to the time series dataset leads to suboptimal performance due to their inability to effectively leverage the rich information inherent in time series data, particularly in the frequency domain. In this paper, we propose a novel framework named Dataset **Cond**ensation for **T**ime **S**eries **C**lassification via Dual Domain Matching (**CondTSC**) which focuses on the time series classification dataset condensation task. Different from previous methods, our proposed framework aims to generate a condensed dataset that matches the surrogate objectives in both the time and frequency domains. Specifically, CondTSC incorporates multi-view data augmentation, dual domain training, and dual surrogate objectives to enhance the dataset condensation process in the time and frequency domains. Through extensive experiments, we demonstrate the effectiveness of our proposed framework, which outperforms other baselines and learns a condensed synthetic dataset that exhibits desirable characteristics such as conforming to the distribution of the original data.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Dataset Condensation; Data Mining; Time Series Classification
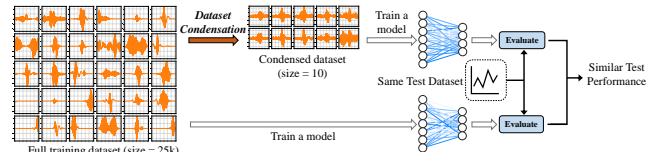
---

*Corresponding Author

Figure 1: The diagram illustrates the concept of time series data condensation, which aims to learn a small dataset that achieves comparable performance to the full dataset.

1 INTRODUCTION

The exponential growth of time series data across various domains, such as traffic forecasting [33, 34, 37–40], clinical diagnosis [20, 21], and financial analysis [6, 28], has presented opportunities for researchers and practitioners. However, the sheer volume of data has imposed burdens in terms of storage, processing, and analysis in terms of deep learning context. In the context of some deep learning downstream tasks such as neural architecture search [47] and continual learning [19], the utilization of the full dataset for training has the potential to yield bad efficiency. Consequently, there is an urgent need to develop effective strategies for condensing time series datasets while preserving their essential information to alleviate the efficiency challenge of training a deep model.

To address this issue, *Dataset Condensation* [54, 63] has emerged as a powerful and efficient approach. The core idea is learning a small synthetic dataset that achieves comparable performance to the original full dataset when training the identical model, as depicted in Fig. 1. Unlike traditional data compression techniques, the primary objective of dataset condensation is to distill a reduced-sized synthetic dataset that could efficiently train a model and effectively generalize to unseen test data rather than just reduce the size without information loss. Notably, the fundamental principles of data condensation encompass several strategies aimed at addressing specific surrogate objectives. These strategies include maximizing the testing performance [13, 36, 41, 45, 46, 54, 65], matching the training gradient [4, 9, 14, 25–27, 30, 32, 58, 61, 63], and matching the hidden state distribution when training [29, 53, 62].

Currently, most of the data condensation research focuses on image [13, 30, 46, 53, 65], and graph [25, 26, 35, 48] dataset. However, the dataset condensation research for time series data is not well explored. Different from the other modalities, time series data exhibits distinct characteristics, such as periodicity and seasonality,

which are closely related to its frequency domain. The frequency domain plays a crucial role, offering valuable insights and aiding in time series analysis [56, 57, 59, 60, 64]. In the frequency domain, the key patterns and characteristics of time series data that might be difficult to discern in the time domain alone could be identified. Consequently, there is a pressing need to develop dataset condensation methods specifically tailored for time series data, capitalizing on the rich information available in the frequency domain to enhance the efficiency and effectiveness of condensation techniques.

In this paper, we propose a framework named Dataset **Cond**ensation for **T**ime **S**eries **C**lassification via Dual Domain Matching (**CondTSC**). The proposed framework aims to generate a condensed synthetic dataset by matching the surrogate objectives between synthetic dataset and full real dataset in both the time domain and frequency domain. Specifically, we first introduce **Multi-view Data Augmentation** module, which utilizes the data augmentation techniques to project the synthetic data into different frequency-enhanced spaces. This approach enriches data samples and strengthens the matching process of surrogate objectives, resulting in synthetic data that is more robust and representative. Then, we propose **Dual Domain Training** module, which leverages both the time domain and frequency domain of the synthetic data. By encoding information from both domains, the downstream surrogate matching module benefits from the rich dual-domain information. Furthermore, we introduce **Dual Objectives Matching** module. By matching the surrogate objectives, training these networks with the condensed synthetic dataset yields similar gradient and hidden state distributions as training with the full real dataset. This enables the synthetic data to learn the dynamics of real data when training the networks such as CNN or Transformer. Overall, the CondTSC framework generates a condensed synthetic dataset that keeps the dynamics of training a network for time series analysis.

Extensive experiments show that CondTSC achieves outstanding performance in the data condensation task in many scenarios such as human activity recognition (HAR) and insect audio classification. For example, we achieve 61.38% accuracy with 0.1% of the original size and 86.64% accuracy with 1% of the original size, compared with 93.14% accuracy with full original size in HAR dataset.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to systematically complete the data condensation problem for time series classification. We highlight the importance of the frequency domain in time series analysis and propose to integrate the frequency domain view into the dataset condensation process. This allows for a comprehensive understanding and analysis of time series patterns and behaviors.
- We propose a novel framework named CondTSC, which incorporates multi-view data augmentation, dual domain training, and dual surrogate objectives to enhance the dataset condensation process in the time and frequency domains.
- We validate the effectiveness of CondTSC through extensive experiments. Results show our proposed framework not only outperforms other baselines in the condensed training setting but also shows good characteristics such as conforming to the distribution of the original data.

2 RELATED WORK

2.1 Time Series Compression

Time series compression aims to compress the time series data without losing the important information, which benefits the process of storage, analysis, and transmission [7]. There are many works that focus on this problem by using dictionary compression [42], Huffman coding [43], and so on. However, the compressed dataset does not work efficiently in training a deep network. In order to utilize the compressed data, it must first be decoded to its original size, and subsequently, the network iterates through the original-sized dataset to generate loss for back-propagation. Both of these processes are time-consuming. Consequently, time series compression is unsuitable for tasks demanding high training efficiency, such as neural architecture search [15]. To handle this problem, a newly emerging area *Dataset Condensation* gives its solution.

2.2 Dataset Condensation

Dataset Condensation aims to learn a small synthetic dataset that could achieve comparable performance to the original full dataset when training a deep neural network. Currently, there are three main approaches to Dataset Condensation as follows. (1) The testing performance maximization approaches focus on maximizing the performance on the real-world test-split dataset of the network trained by the condensed synthetic dataset [13, 36, 41, 45, 46, 54, 65]. These approaches would face the challenge of the high cost of computing high-order derivatives and thus some of the works utilize the kernel ridge regression [45, 46] or neural feature regression [65] to reduce the computational complexity. (2) The training gradient matching approach aims to match the training gradient of the model trained on the condensed dataset with that of the model trained on the full dataset [4, 9, 14, 25–27, 30, 32, 58, 61, 63]. By matching the gradients, the condensed dataset captures the essential full-data training dynamics and enables efficient model training. Moreover, there are two main surrogate matching objectives including single-step gradient [9, 25–27, 30, 32, 61, 63] and multi-step gradients [4, 14, 58]. (3) The hidden state distribution matching approach focuses on matching the distribution of hidden states during the training process [29, 49, 53, 62]. These approaches have high efficiency due to their low computational complexity. However, all approaches of these three types focus on images or graphs, lacking detailed analysis for techniques for the time series datasets.

2.3 Frequency-enhanced Time series analysis

Recently, several studies have focused on enhancing time series analysis by leveraging the frequency domain. MCNN [10] incorporates multi-frequency augmentation of time series data and utilizes a simple convolutional network. Frequency-CNN [1] utilizes a simple convolutional network on the frequency domain and gets good results on automatic early detection. BTSF [57] proposes iterative bilinear spectral-time mutual fusion and trains the encoder based on triplet loss. Recently, many studies have focused on developing frequency-based self-supervised architectures specifically designed for time series analysis. CoST [56] and TF-C [60] construct contrastive losses based on the frequency domain pattern to learn more robust representations. FEDFormer [64] proposes
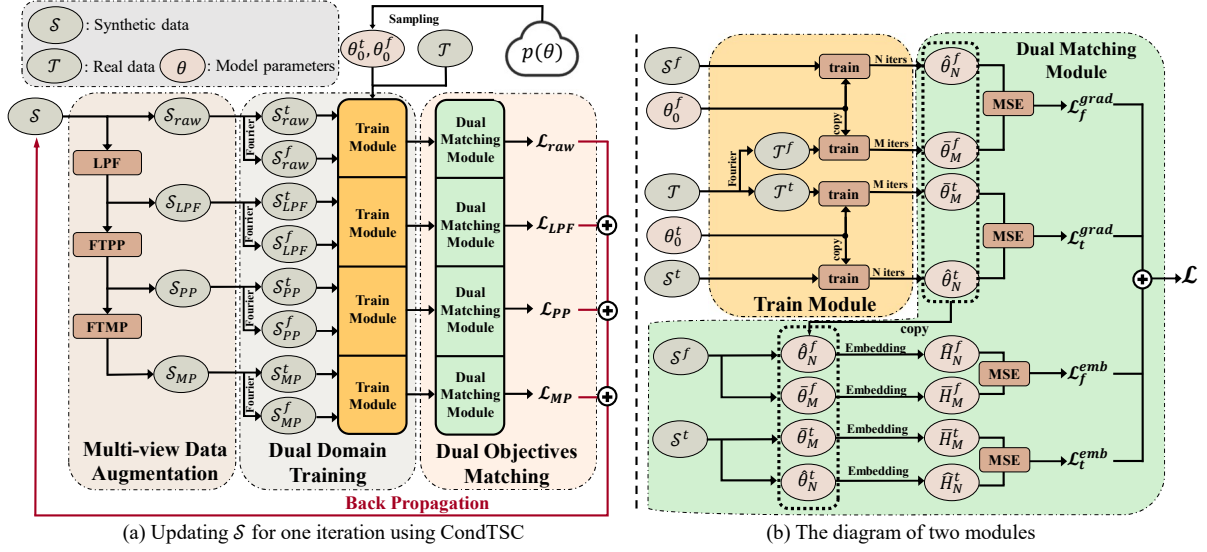
(a) Updating $\mathcal{S}$ for one iteration using CondTSC   (b) The diagram of two modules

Figure 2: The diagram of CondTSC. LPF indicates low pass filter. FTPP indicates Fourier transform phase perturbation and FTMP indicates Fourier transform magnitude perturbation.

a frequency-based representation learning to improve the effectiveness and efficiency of the Transformer. The outstanding results achieved by these methods have provided evidence for the effectiveness of incorporating frequency information in time series analysis.

3 PRELIMINARY

3.1 Problem Overview

The goal of Data Condensation is to learn a synthetic dataset $\mathcal{S} = \{(s_i, y_i)\}_{i=1}^{|\mathcal{S}|}$ from the real training dataset $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{T}|}$, where each data $s_i, x_i \in \mathbb{R}^d$, and the label $y_i \in \mathcal{Y} = \{0, 1, \ldots, C - 1\}$. We want the size of the synthetic dataset to be much smaller than the size of the real training dataset, i.e., $|\mathcal{S}| \ll |\mathcal{T}|$ while keeping the network trained by synthetic data comparable to the network trained by the real training dataset.

Formally, by denoting $\widehat{\theta}$ as the network parameter trained by $\mathcal{S}$ and $\bar{\theta}$ as the network parameter trained by $\mathcal{T}$, we could write as follows:

$$\widehat{\theta} = \arg\min_{\theta} \sum_{(s_i, y_i) \in \mathcal{S}} \ell(s_i, y_i, f_\theta), \quad (1)$$

$$\bar{\theta} = \arg\min_{\theta} \sum_{(x_i, y_i) \in \mathcal{T}} \ell(x_i, y_i, f_\theta). \quad (2)$$

Then, the aim is to ensure that both networks exhibit comparable generalization capabilities when evaluated on the test dataset $\mathcal{T}_{test}$ as follows:

$$\mathbb{E}_{(x,y) \in \mathcal{T}_{test}} [Eval(f_{\widehat{\theta}}(x), y)] \simeq \mathbb{E}_{(x,y) \in \mathcal{T}_{test}} [Eval(f_{\bar{\theta}}(x), y)]. \quad (3)$$

Here, $\ell(\cdot, \cdot, \cdot)$ indicates training loss, the f_θ indicates the network parameterized by θ and $Eval(\cdot, \cdot)$ indicates arbitrary evaluation metrics such as accuracy and recall.

4 METHOD

In this section, we present our proposed framework CondTSC, which comprises three key modules. The framework's architecture is depicted in Fig. 2, and the detailed pseudo code is shown in Alg. 1. The first module, termed Multi-view Data Augmentation, projects the synthetic dataset $\mathcal{S}$ into spaces of multi-view to enrich the data samples. Next, the Dual Domain Encoding module applies the Fourier transform and trains separate networks on each domain to obtain network parameters specific to $\mathcal{S}$ and $\mathcal{T}$. Lastly, the Dual Objectives Matching module leverages the trained network parameters to construct two types of surrogate objectives: parameter matching and embedding space matching. By matching these surrogate objectives, the losses computed across all views can be back-propagated to update the synthetic dataset $\mathcal{S}$ accordingly.

4.1 Initializing $\mathcal{S}$

In previous research, pixel-level random initialization has been commonly utilized to initialize $\mathcal{S}$, as evidenced by numerous studies [4, 25–27, 61–63]. However, these studies have focused primarily on condensing image or graph datasets. In contrast, empirical evidence suggests that random initialization results in poor performance when condensing time-series datasets. Therefore, to address this issue, we propose using K-means to cluster the data samples of each class from the full dataset $\mathcal{T}$, and then selecting the clustering centroids of each class as the initial data samples of $\mathcal{S}$. Formally, it could be formulated as follows.

$$\mathcal{S}_i = Kmeans(\mathcal{T}_i, spc) \quad (4)$$

In this context, $\mathcal{S}_i$ represents the samples of the i-th class within $\mathcal{S}$, while $\mathcal{T}_i$ represents the samples of the i-th class within $\mathcal{T}$. The function $Kmeans(X, K)$ employs K-means algorithm on data X with K clustering centroids and outputs the centroids to initialize $\mathcal{S}_i$.

Data augmentation squeezes the boundaries → Harder to classify → Learning a more effective synthetic dataset
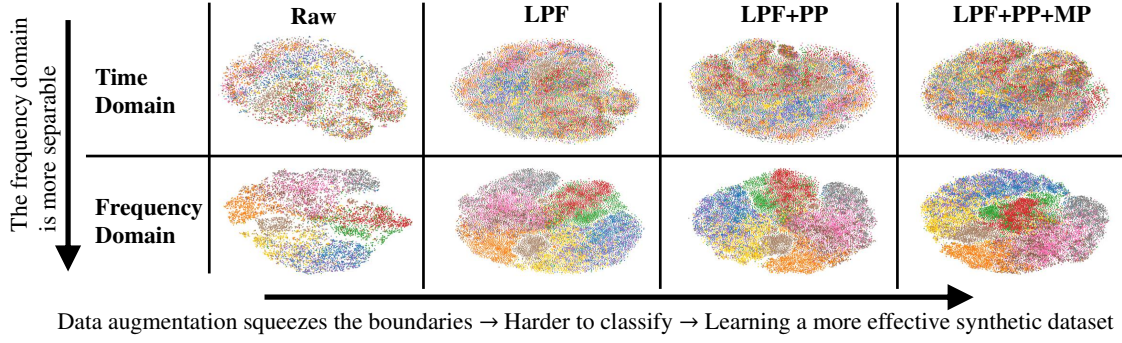
Figure 3: The TSNE visualization of the Insect dataset in both the time domain and the frequency domain. This demonstrates the intuition to do augmentation and utilize the dual-domain information for the time series data. The data in the frequency domain shows better decision boundaries and the data augmentations squeeze the boundaries between different classes.

4.2 Multi-view Data Augmentation

Goal: This module essentially projects the synthetic data $\mathcal{S}$ into embedding spaces of multiple views by conducting several data augmentations [55] sequentially. The visualization in Figure 3 demonstrates the increased separability of the frequency domain, suggesting that augmenting data in the frequency domain yields more effective results. Consequently, different from [13, 61] that uses augmentations for images such as jittering and cropping, we employ frequency-domain augmentations such as low pass filtering. The augmentation process, as depicted in Figure 3, generates more data and compresses the boundaries between classes in the frequency domain. As a result, the synthetic data $\mathcal{S}$ can more effectively adhere to the training dynamics such as gradient and embeddings exhibited by the real data $\mathcal{T}$ by virtue of matching the surrogate objectives across different views of the synthetic data $\mathcal{S}$.

Augmentations: As shown in Fig. 2(a), we will conduct three frequency-enhanced data augmentations sequentially to $\mathcal{S}$, including Low Pass Filter (LPF), Fourier Transform Phase Perturbation (FTPP), Fourier Transform Magnitude Perturbation (FTMP).

First, by denoting the Fourier Transform and Inverse Fourier Transform as $FT(\cdot)$ and $IFT(\cdot)$ respectively, the first data augmentation LPF could be formally written as follows:

$$\mathcal{S}_{raw} = \mathcal{S}, \tag{5}$$

$$\mathcal{S}_{LPF} = IFT(LPF(FT(\mathcal{S}_{raw}))). \tag{6}$$

Here $LPF(\cdot)$ indicates a low pass filter. We directly keep 50% of the low-frequency components and discard the remaining, thereby ensuring the robustness of the synthetic data while preserving the key information in the frequency domain (above 90%). Then, we perturb the phase of the low-passed synthetic data $\mathcal{S}_{LPF}$ as follows:

$$\mathcal{S}_{PP} = IFT(Noise(FT(\mathcal{S}_{LPF})_{phase})). \tag{7}$$

Here $FT(\cdot)_{phase}$ indicates the phase of the frequency domain. To introduce diversity and enhance the robustness of the synthetic data, Gaussian noise is added to the phase components using the $Noise(\cdot)$ function. This perturbation further expands the variation within the synthetic dataset. Furthermore, we perturb the magnitude of $\mathcal{S}_{PP}$ as follows:

$$\mathcal{S}_{MP} = IFT(Noise(FT(\mathcal{S}_{PP})_{magnitude})). \tag{8}$$

Here $FT(\cdot)_{magnitude}$ indicates the magnitude of the frequency domain. This perturbation introduces variations in the magnitude of the frequency domain, allowing the synthetic data to capture a wider range of amplitude and intensity characteristics.

In summary, the Multi-view Data Augmentation module in the CondTSC framework plays a crucial role in enriching the data samples and enhancing the matching of surrogate objectives by projecting the synthetic data $\mathcal{S}$ into multiple high-dimensional spaces through sequential data augmentations.

4.3 Dual Domain Training

Goal: As shown in Fig. 3, the frequency domain is more separable, which means fusing the frequency domain into the dataset condensation could enhance the performance and effectiveness. Consequently, this module aims to incorporate both the time and frequency domains and utilize them in the construction of surrogate objectives in the next module.

Training Module: The Training Module receives three types of input: the synthetic data in both the time domain $\mathcal{S}^t$ and the frequency domain $\mathcal{S}^f$, the full real data in both the time domain $\mathcal{T}^t$ and the frequency domain $\mathcal{T}^f$, and the sampled network parameters in both time domain θ_0^t and the frequency domain θ_0^f randomly sampled from $p(\theta)$. **For simplicity**, we here introduce the training process in the time domain, while the process in the frequency domain is identical.

The training process follows standard training procedures, where we utilize the synthetic data $\mathcal{S}^t$ and the real data $\mathcal{T}^t$ to train the network parameters θ_0^t. We employ standard training settings, such as stochastic gradient descent or Adam optimization, to update the parameters based on the loss between the network predictions and the ground truth labels. Furthermore, we train the network with $\mathcal{S}^t$ and $\mathcal{T}^t$ for N and M iterations respectively. We set $N \ll M$ to avoid overfitting to the small synthetic data $\mathcal{S}^t$.

The output of the Training Module is the trained parameters $\widehat{\theta}_N^t$ (trained by $\mathcal{S}$) and $\bar{\theta}_M^t$ (trained by $\mathcal{T}$). These trained parameters play a crucial role in the subsequent modules as they are utilized in the construction of surrogate objectives, which facilitate the matching of training dynamics between the synthetic and real datasets.

Algorithm 1: Pseudo code for CondTSC.

Input: Full real data $\mathcal{T}$, network parameter distribution
$p(\theta)$, samples per class spc, training epochs Epo.

Output: Condensed Synthetic data $\mathcal{S}$

1 Initialize $\mathcal{S}_i$ for $i = \{0, 1, \ldots, C - 1\}$ based on Eq. (4);

2 **for** $e \leftarrow range(0, Epo, 1)$ **do**

3    $\mathcal{L} \leftarrow 0.0, \quad aug\_list \leftarrow [raw, LPF, PP, MP]$;

4   **for** $aug \leftarrow aug\_list$ **do**

     // Multi-view Data Augmentation

5   Compute $\mathcal{S}_{aug}$ based on Eqs (5),(6),(7),(8);

     // Dual Domain Training

6      $\mathcal{S}_{aug}^t \leftarrow \mathcal{S}_{aug}, \quad \mathcal{S}_{aug}^f \leftarrow FT(\mathcal{S}_{aug}^t)$;

7      $\mathcal{T}^t \leftarrow batch(\mathcal{T}), \quad \mathcal{T}^f \leftarrow FT(\mathcal{T}^t)$;

8      $\theta_0^t, \theta_0^f \leftarrow Sample(p(\theta))$;

9      $\widehat{\theta}_N^t \leftarrow Train\_Niters(\mathcal{S}_{aug}^t, \theta_0^t)$;

10     $\widehat{\theta}_N^f \leftarrow Train\_Niters(\mathcal{S}_{aug}^f, \theta_0^f)$;

11     $\bar{\theta}_M^t \leftarrow Train\_Miters(\mathcal{T}^t, \theta_0^t)$;

12     $\bar{\theta}_M^f \leftarrow Train\_Miters(\mathcal{T}^f, \theta_0^f)$;

     // Dual Objectives Matching

     // Surrogate objective 1: multi-step
       gradient

13     $\mathcal{L}_t^{grad} = \frac{||\widehat{\theta}_N^t - \bar{\theta}_M^t||_2^2}{||\theta_0^t - \bar{\theta}_M^t||_2^2}, \quad \mathcal{L}_f^{grad} = \frac{||\widehat{\theta}_N^f - \bar{\theta}_M^f||_2^2}{||\theta_0^f - \bar{\theta}_M^f||_2^2}$;

     // Surrogate objective 2: embedding space

14   Compute $\mathcal{L}_t^{emb}, \mathcal{L}_f^{emb}$ based on Eqs 14,15;

15     $\mathcal{L} = \mathcal{L} + \mathcal{L}_t^{grad} + \mathcal{L}_f^{grad} + \lambda(\mathcal{L}_t^{emb} + \mathcal{L}_f^{emb})$

16   **end**

17   Back-propagate $\mathcal{L}$ and update $\mathcal{S}$

18 **end**

19 **return** $\mathcal{S}$

Model Parameters $p(\theta)$**:** As aforementioned, the aim of dataset condensation is to train a network using the synthetic dataset $\mathcal{S}$ that achieves performance comparable to a network trained using the real dataset $\mathcal{T}$. To achieve this, we will match the surrogate objectives with the given parameter distribution. However, in practice, computing the exact parameter distribution is infeasible. Therefore, we collect the parameters from the training trajectories when training a network with $\mathcal{T}$ and utilize them as an approximation for the parameter distribution $p(\theta)$. Moreover, in line with prior research [4, 54, 61, 63], we employ a single network structure in the training process, i.e., the used $p(\theta)$ is from a parameter space of single network structure f. Furthermore, we conduct experiments that train $\mathcal{S}$ based on one network structure distribution $p_1(\theta)$ and evaluate the trained $\mathcal{S}$ on another distribution $p_2(\theta)$ to further evaluate the effectiveness of our proposed framework.

4.4 Dual Domain Surrogate Objective Matching

Goal: This module aims to generate surrogate objectives from different views using both the synthetic dataset $\mathcal{S}$ and the real dataset $\mathcal{T}$,

and subsequently match these objectives. By computing the matching loss across the different views, we can update the condensed dataset $\mathcal{S}$. After the surrogate objectives are well matched, the synthetic dataset $\mathcal{S}$ could be considered as the condensed dataset of the original dataset $\mathcal{T}$ for learning the downstream task.

Gradient Matching: The aim of gradient matching is to make the gradient of $\mathcal{S}$ and $\mathcal{T}$ similar. By doing so, training a network with $\mathcal{S}$ could generate a similar trained model, and thus the performance could be comparable. Here we take the multi-step gradient of training a network parameterized by θ_0 as the surrogate objective similar to [4, 14, 58]. Intuitively, we directly minimize the mean square error between these gradients. Taking the parameter in the time domain as an example, the error could be formulated as follows:

$$\mathcal{L}_t^{grad} = \frac{||grad_{\mathcal{S}} - grad_{\mathcal{T}}||_2^2}{||grad_{\mathcal{T}}||_2^2}. \tag{9}$$

Here, we use the final network parameter $\widehat{\theta}_N^t$ to minus the initial network parameter θ_0^t to get the gradient $grad_{\mathcal{S}} = \widehat{\theta}_N^t - \theta_0^t$. Moreover, $grad_{\mathcal{T}}$ could be similarly calculated. Consequently, we get the gradient matching loss in the time domain as follows:

$$\mathcal{L}_t^{grad} = \frac{||(\widehat{\theta}_N^t - \theta_0^t) - (\bar{\theta}_M^t - \theta_0^t)||_2^2}{||\bar{\theta}_M^t - \theta_0^t||_2^2}$$
$$= \frac{||\widehat{\theta}_N^t - \bar{\theta}_M^t||_2^2}{||\theta_0^t - \bar{\theta}_M^t||_2^2}. \tag{10}$$

Furthermore, the gradient matching loss in the frequency domain could be similarly formulated as follows

$$\mathcal{L}_f^{grad} = \frac{||\widehat{\theta}_N^f - \bar{\theta}_M^f||_2^2}{||\theta_0^f - \bar{\theta}_M^f||_2^2}. \tag{11}$$

Parameter Space Matching Previous methods such as those described in [29, 53, 62] aimed to match the distribution of embedding between $\mathcal{S}$ and $\mathcal{T}$ using maximum mean discrepancy (MMD)[18, 50, 62]. However, these methods use random model parameters θ_{rand} and thus did not utilize the rich information of training dynamics contained in the space of trained parameters $\widehat{\theta}_N^t$ and $\bar{\theta}_M^t$. By matching these spaces, these parameters could generate similar embeddings, and thus the training direction of $\widehat{\theta}_N$ could be better guided. To match these spaces, we first compute the embedding of $\mathcal{S}$ in these spaces

$$\widehat{H}_N^t = f_{\widehat{\theta}_N^t}(\{s_i|(s_i, y_i) \in \mathcal{S}^t\}), \bar{H}_M^t = f_{\bar{\theta}_M^t}(\{s_i|(s_i, y_i) \in \mathcal{S}^t\}), \tag{12}$$

$$\widehat{H}_N^f = f_{\widehat{\theta}_N^f}(\{s_i|(s_i, y_i) \in \mathcal{S}^f\}), \bar{H}_M^f = f_{\bar{\theta}_M^f}(\{s_i|(s_i, y_i) \in \mathcal{S}^f\}). \tag{13}$$

Then, we match the position-wise mean of the embedding of the time domain parameters and the frequency domain parameters separately, i.e., the mean that converts tensor size from (B, D) to (D) where B is the batch size and D is the embedding size as follows:

$$\mathcal{L}_t^{emb} = ||Mean(\widehat{H}_N^t) - Mean(\bar{H}_M^t)||_2^2, \tag{14}$$

$$\mathcal{L}_f^{emb} = ||Mean(\widehat{H}_N^f) - Mean(\bar{H}_M^f)||_2^2. \tag{15}$$

Finally, the total loss could be summarized as follows:

$$\mathcal{L}_{aug} = \mathcal{L}_t^{grad} + \mathcal{L}_f^{grad} + \lambda(\mathcal{L}_t^{emb} + \mathcal{L}_f^{emb}). \tag{16}$$

Table 1: Details of time series datasets.

Dataset	# Train	# Test	Length	# Channel	# Class
HAR	7,352	2,947	128	9	6
Electric	8,926	7,711	96	1	7
Insect	25,000	25,000	600	1	10
FD	8,184	2,728	5,120	1	3
Sleep	25,612	8,910	3,000	1	5

Here $aug \in \{raw, LPF, PP, MP\}$ indicates different views of synthetic data as shown in Fig. 2 and λ is a scaling factor between the gradient loss and the embedding space matching loss. We then summarize the losses of all views as follows.

$$\mathcal{L} = \mathcal{L}_{raw} + \mathcal{L}_{LPF} + \mathcal{L}_{PP} + \mathcal{L}_{MP} \tag{17}$$

The loss could be summed and back-propagated to $\mathcal{S}$ to make it better capture the training dynamics of $\mathcal{T}$.

5 EXPERIMENT

5.1 Datasets

In line with previous research [53, 54, 63], which utilizes small benchmark datasets for image classification dataset condensation task, we include five public real-world benchmark datasets as shown in Table 1. (1) **HAR [2]:** The Human Activity Recognition (HAR) dataset comprises recordings of 30 individuals who volunteered for a health study and engaged in six different daily activities. (2) **Electric** [11]: The ElectricDevice dataset comprises measurements obtained from 251 households, with a sampling frequency of two minutes. (3) **Insect [5]:** The InsectSound dataset comprises 50,000 time series instances, with each instance generated by a single species of fly. (4) **Sleep [17]:** The Sleep Stage Classification dataset comprises recordings of 20 people throughout a whole night with a sampling rate of 100 Hz. (5) **FD [31]:** The Fault Diagnosis (FD) dataset contains sensor data from a bearing machine under four different conditions, and we chose the first one.

5.2 Experiment Setting

Evaluation protocol: We follow the evaluation protocol of previous studies [61–63]. Concretely, it contains three stages sequentially: (1) train a synthetic data $\mathcal{S}$. (2) train a randomly initialized network of the same structure on the trained synthetic data $\mathcal{S}$ using the same optimization setting (with 1e-3 learning rate and 300 training epochs). (3) Evaluate the trained network on the same test data with the same evaluation setting. In all experiments, this process would be repeated five times to reduce the randomness of the results. We report the mean and the standard deviation of the results. In line with prior research [13, 25–27, 36, 41, 46, 54, 61–63, 65] that utilize simple deep models to verify the effectiveness of the proposed method, we train the synthetic data based on a wide range of basic models. The details of the network architecture and the parameter searching are in the Appendix B. In summary, we believe the comparison between our proposed method and baseline methods is fair.

Hyper-parameters: There are only a small number of hyper-parameters associated with CondTSC. For simplicity, we use the same set of hyper-parameters for all datasets. For the training process in Train Module, we use a learnable learning rate of 1e-3

following previous settings [4, 14, 58]. Moreover, we set $N = 10$ and $M = 1000$ for the number of the training iterations for $\mathcal{S}$ and $\mathcal{T}$ respectively. For the loss in Dual Matching Module, we set $\lambda = 1.0$. For the update of the synthetic data $\mathcal{S}$, we use an SGD optimizer with a learning rate of $lr_{\mathcal{S}} = 1.0$ and update epochs $Epo = 2000$. The whole experiment is implemented by Pytorch 1.10.0 on RTX3090. The code and data are available in https://github.com/zhyliu00/TimeSeriesCond.

5.3 Overall Performance

Baselines: We select ten baselines to evaluate the performance of CondTSC on the time series data condensation task. These baselines comprise both coreset selection methods and previous SOTA data condensation methods.

- Coreset selection methods: We randomly select data samples (**Random**), choose the K-means clustering centroids (**K-means**), add samples of each class to the coreset greedily (**Herding**).
- Data condensation methods: Since there is no method designed for time series data condensation, we select several data condensation methods for image data. **DD [54]** maximizes the performance of the network trained by the synthetic data. **DC [63]** matches the one-step training gradient of the synthetic data and the real data. **DSA [61]** implements siamese augmentation based on DC. **DM [62]** matches the embedding distribution of the synthetic data and the real data. **MTT [4]** matches the multi-step training gradient. **IDC [27]** uses efficient parameterization to improve the space utilization. **HaBa [36]** decomposes the synthetic data into bases and hallucinators.

Overall Performance Table 2 shows the accuracy of baselines and CondTSC. Ratio (%) indicates the condensed ratio which is $size(\mathcal{S})/size(\mathcal{T})$ and spc indicates the number of samples per class of the synthetic data. Full indicates the performance of the network trained by full data. We report the mean and the standard deviation of the accuracy of five experiments with different random seeds.

From Table 2, the following observations could be drawn: (1) CondTSC achieves superior performance compared to the baselines. The accuracy gap between the condensed data and full data is comparable to the data condensation task in computer vision [61–63], and the condensed data is enough for downstream tasks such as Neural Architecture Search, which we will introduce later. (2) The result of CondTSC is significant in all settings, and CondTSC could approach the performance of full data by using only 1% of the size of the real data. (3) Coreset selection methods achieve stable but mediocre performance. This indicates that directly selecting some samples of the real data gives a base but not global information about the real data. (4) The performance of the data condensation methods is not as satisfactory as the performance of the image dataset condensation task. This highlights the significant disparity between the task of condensing time series data and image data.

Furthermore, we evaluate the performance of CondTSC and coreset selection methods with larger condensed data sizes as shown in Fig. 4. We can observe that increasing the data size makes the methods approach the upper bound and CondTSC has the highest performance. The users could make a trade-off between the accuracy and the training cost according to their downstream tasks.

Table 2: Overall accuracy(%) performance. The experiment is conducted with the CNN network with batch normalization. The best result is highlighted in bold and grey in each row, and the second-best result is underlined. Marker * indicates the mean of the results is statistically significant (* means t-test with p-value < 0.01).

	spc	ratio (%)	Random	K-means	Herding	DD	DC	DSA	DM	MTT	IDC	HaBa	CondTSC	Full (upper bound)
HAR	1	0.1	44.38±4.27	50.97±4.30	56.09±2.87	21.29±6.32	<u>58.30±3.32</u>	55.41±6.67	52.87±4.64	50.36±3.85	40.93±4.62	45.53±6.44	**61.38±3.71***	93.14±1.03
	5	0.5	62.98±2.21	72.19±1.83	64.37±1.85	18.71±5.27	65.83±3.97	65.16±3.67	70.82±4.63	<u>77.66±3.86</u>	54.41±5.02	56.22±5.56	**82.20±2.68***	
	10	1	68.77±1.82	76.31±2.24	69.37±2.49	19.64±6.78	67.50±5.15	70.99±3.46	78.25±3.43	<u>83.66±2.65</u>	59.99±3.37	60.37±6.43	**86.64±2.10***	
Electric	1	0.1	38.40±4.28	39.07±2.67	32.10±3.61	34.95±6.38	47.13±5.74	<u>47.63±6.05</u>	44.86±6.05	39.03±3.20	42.56±3.60	41.91±3.24	**48.84±2.18***	68.35±0.79
	5	0.5	45.55±3.91	46.87±2.62	35.46±4.65	35.09±8.16	50.52±3.58	<u>51.76±2.67</u>	51.22±2.85	49.09±3.99	49.52±5.82	44.32±2.53	**56.17±1.91***	
	10	1	47.83±1.88	48.76±2.10	47.92±2.48	35.69±8.23	50.37±2.66	52.06±4.15	<u>52.75±3.74</u>	51.29±2.97	50.34±5.08	46.71±2.23	**57.86±1.89***	
Insect	1	0.05	14.26±0.65	<u>16.70±0.95</u>	10.42±0.57	10.61±0.74	15.48±2.09	15.12±1.89	12.24±0.86	15.54±1.75	13.63±0.61	14.63±0.72	**45.15±2.99***	70.78±1.19
	10	0.5	23.34±1.16	25.55±1.07	19.40±0.59	10.15±0.41	21.56±2.12	21.63±2.13	20.60±1.02	<u>28.17±4.65</u>	21.29±1.32	22.83±1.50	**63.75±0.58***	
	20	1	30.29±1.28	30.10±1.22	23.88±0.78	10.18±0.30	24.63±4.11	24.12±5.50	25.11±1.24	<u>33.83±3.04</u>	25.12±1.96	26.62±2.48	**64.69±0.60***	
FD	1	0.036	46.24±2.89	36.48±9.52	43.79±6.60	33.93±16.0	35.50±17.8	47.19±5.73	–	44.41±6.81	<u>49.87±2.15</u>	45.91±1.67	**71.00±7.12***	98.51±0.35
	10	0.36	54.99±1.96	54.06±3.80	58.58±2.07	29.00±18.3	44.02±16.8	41.15±12.9	59.62±4.01	<u>68.29±4.00</u>	53.75±5.96	51.43±1.03	**83.25±1.76***	
	20	0.72	60.72±2.37	57.74±4.19	59.68±1.76	33.47±16.3	45.73±15.5	45.46±0.01	64.75±3.64	<u>74.63±3.95</u>	61.77±7.75	50.18±2.03	**90.78±1.36***	
Sleep	1	0.02	22.68±5.01	31.62±7.09	22.46±6.52	23.43±6.94	22.18±6.73	22.98±6.98	30.03±8.88	30.39±4.96	28.24±4.97	<u>31.72±3.98</u>	**49.71±7.53***	80.01±1.12
	10	0.2	40.35±3.90	36.88±3.80	27.09±3.71	22.53±5.97	23.90±3.97	23.82±3.93	36.08±5.92	<u>47.61±5.81</u>	30.03±5.50	40.13±3.34	**57.63±2.25***	
	50	1	57.10±4.84	45.88±2.93	30.69±3.75	22.82±5.65	24.31±6.42	25.57±4.53	40.09±4.51	<u>60.52±2.41</u>	34.59±5.60	55.12±5.67	**66.46±1.00***	

Table 3: The Neural Architecture Search (NAS) result. We implement this experiment on the HAR dataset with the search space of 324 ConvNets. The condensed dataset has a size of 60 (10 samples/class, 1% condensed ratio).

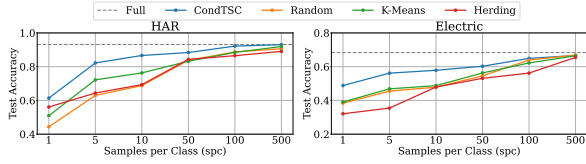|	Random	K-means	Herding	DC	DSA	DM	MTT	IDC	HaBa	CondTSC	Full
Accuracy	87.70±0.80	89.59±1.09	89.91±0.75	90.38±0.62	90.68±1.05	90.04±1.38	90.07±0.97	91.01±0.66	82.99±0.59	**92.71±0.81**	93.74±0.78
Correlation	0.587	0.547	0.482	0.560	0.307	0.528	0.582	0.364	-0.413	**0.665**	1.000
Time cost(min)	24.49	24.69	24.53	24.45	24.49	24.45	24.50	24.39	144.89	24.36	667.91
Training samples	60	60	60	60	60	60	60	60	60	60	7,352

Figure 4: The accuracy(%) with larger condensed data size.

5.4 Downstream Task: NAS

The condensed dataset could be used as a proxy dataset for Neural Architecture Search (NAS) [16]. Following the setting of previous research [61, 62], we implement NAS to search the best architecture of 324 Convolutional neural networks on the HAR dataset. The space is Depth $\in \{2, 3, 4\}$, Width $\in \{$ 32, 64, 128 $\}$, Normalization $\in \{$ None, BatchNorm, InstanceNorm, LayerNorm $\}$, Activation $\in \{$ Sigmoid, ReLU, LeakyReLU $\}$, Pooling $\in \{$ None, Max, Mean $\}$. We use 10 samples/class condensed datasets generated by each method.

The result is shown in Table 3. We report 1) the test performance of the model of the best-selected architecture. 2) Spearman's rank correlation coefficient between the validation accuracy obtained from the condensed dataset and the full dataset. We select the top 100 accuracies to filter the outliers following previous research [61, 62]. 3) The training time on an RTX3090 GPU and the number of training samples. We can observe that CondTSC finds the best architecture (92.71%) and performance correlation to the full dataset (0.665) while decreasing the time cost (from 667 to 24 minutes) and training samples (7352 samples to 60 samples). The condensing cost is one time off and negligible when training thousands of possible architectures in NAS. This result indicates that the condensed data

could efficiently and effectively accelerate the process of NAS and get good architecture. Besides, the performance and efficiency of HaBa are bad due to its factorization-based method.

5.5 Ablation Study

In this part, we evaluate the performance of each module of CondTSC to verify the effectiveness of the proposed module. We denote the Base framework as the simple multi-step gradient matching framework [4] that directly matches the multi-step training gradient of the synthetic data $\mathcal{S}$ and the real data $\mathcal{T}$. Then we add the Multi-view Data Augmentation module (A), Dual Domain Training module (T), and Dual Objectives Matching module(M) to the

Table 4: Ablation Study. In each row, the best accuracy(%) result is highlighted in bold and grey.

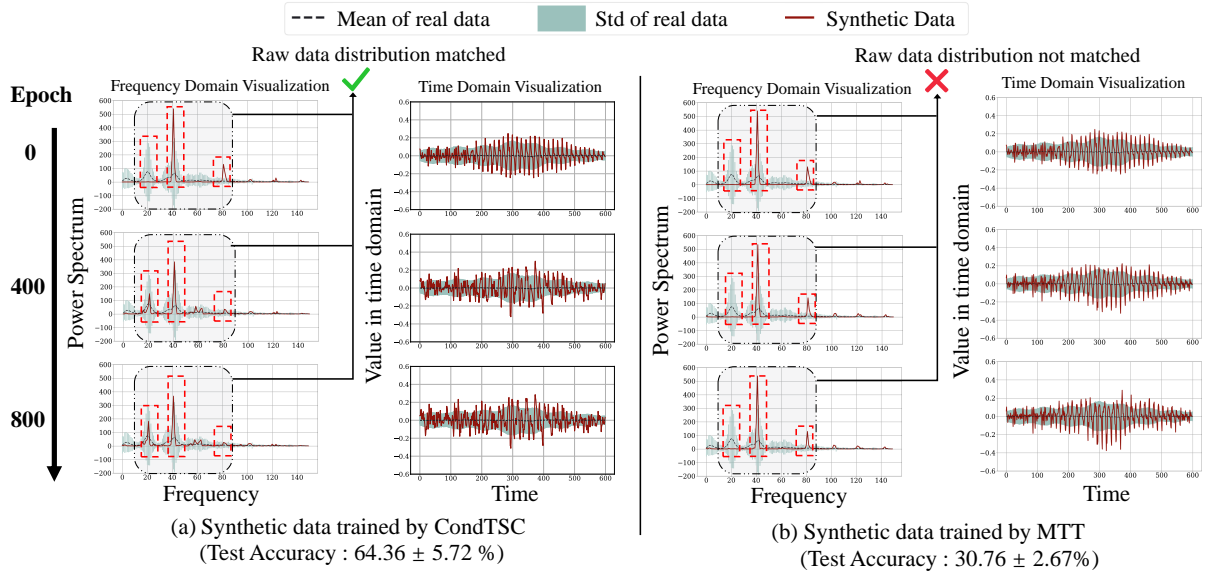|	ratio(%)	I/C	Base	Base+A	Base+A+T	Base+A+T+M
HAR	0.1	1	50.36±3.85	55.63±2.23	60.61±3.49	**61.38±3.71**
	0.5	5	77.66±3.86	80.38±1.85	80.93±3.17	**82.20±2.68**
	1	10	83.66±2.65	84.71±2.61	85.43±2.05	**86.64±2.11**
Electric	0.1	1	39.03±3.20	42.12±4.11	47.53±3.21	**48.84±2.18**
	0.5	5	49.09±3.99	50.14±2.30	55.67±1.29	**56.17±1.91**
	1	10	51.29±2.97	55.59±1.72	57.20±1.21	**57.86±1.89**
Insect	0.05	1	15.54±1.75	18.58±1.72	44.64±2.97	**45.15±2.99**
	0.5	10	28.17±4.65	39.85±2.19	62.83±0.58	**63.75±0.58**
	1	20	33.83±3.04	46.90±1.79	63.93±0.55	**64.69±0.60**
Sleep	0.02	1	30.39±4.96	33.31±3.12	46.09±7.64	**49.71±7.53**
	0.2	10	47.61±5.81	49.20±4.18	55.26±3.27	**57.63±2.25**
	1	50	60.52±2.41	63.43±1.11	65.86±1.22	**66.46±1.00**
FD	0.036	1	44.41±6.81	51.41±3.72	69.62±5.23	**71.00±7.12**
	0.36	10	68.29±4.00	72.48±4.13	81.11±2.66	**83.25±1.76**
	0.72	20	74.63±3.95	77.38±3.38	86.44±2.15	**90.78±1.36**

Figure 5: the frequency domain and time domain visualization on Insect dataset of the learning process of synthetic data trained by CondTSC and MTT separately. We could observe that the synthetic data trained by CondTSC conforms to the distribution of the real data and consequently achieves remarkable performance.

Table 5: Accuracy(%) of CondTSC on the HAR dataset with $spc = 5$ initialized by different methods. Initial means the performance of the initial dataset and Final means the performance of dataset optimized by CondTSC.

Methods	Random	K-means		Kernel K-means		Agglomerative Clustering	
Metric	-	Cosine	Euclidean	Linear	RBF	Ward	Complete
Initial	68.73±1.54	73.56±2.46	72.19±1.83	73.62±1.81	72.43±3.09	64.47±1.83	60.61±1.66
Final	79.89±2.94	82.43±1.86	82.47±3.39	81.90±1.95	81.74±2.77	80.22±2.74	77.13±2.84
Improvement	16.23%	12.05%	14.24%	11.24%	12.85%	24.42%	25.60%

Base framework sequentially and report their mean and standard deviation of accuracy on five experiments with random seeds.

The results are shown in Table 4. We can observe that by sequentially adding the modules, the performance is getting better. By adding the Multi-view Data Augmentation module (Base+A), the synthetic data is extended to several extra spaces to match with the real data, which leads to a considerable performance improvement. By adding the Dual Domain Training module (Base+A+T), the information contained in the frequency domain is introduced into the framework, which leads to a big performance improvement. The improvement is significant in the dataset that is easy to classify in the frequency domain such as Insect and FD. By adding the Dual Objectives Matching module (Base+A+T+M), the matching between synthetic data and real data is strengthened by the parameter space matching loss, which leads to a decent performance improvement. In summary, each module proposed in this paper is effective and contributes to the final performance of CondTSC.

5.6 Case study

In this section, we will investigate why CondTSC outperforms other baselines in the time series dataset condensation task. We randomly select some data samples of the Insect dataset as the

initial condensed synthetic dataset $\mathcal{S}$ and use CondTSC and MTT (strongest baseline) to train $\mathcal{S}$ separately. Then, we visualize the training process of a random sample in $\mathcal{S}$ in both the frequency domain and the time domain, which is shown in Fig. 5.

From Fig. 5, we could observe that: (1) In the frequency domain, the synthetic data trained by CondTSC exhibits a tendency to conform to the distribution of the original data (as shown in the dashed rectangular boxes in the figure). The synthetic data (red solid line) are to fit the mean of the real data (black dashed line) and conform to the standard deviation (green vertical segment) simultaneously. This conformity is subsequently manifested in the time domain. We can see significant changes in the data of the time domain since signals of various frequencies are modified. (2) However, the synthetic data trained by MTT does not vary much, especially in the frequency domain. Moreover, we could observe that the initial high-frequency signals, which deviate from the underlying distribution of the original data, are not modified and incorporated into the final synthetic data. (3) This significant difference in the learned dataset leads to different results, while CondTSC achieves remarkable performance.

5.7 Different Initialization

In this section, we would like to investigate the influence of different initializations of CondTSC. We try different initializations of different metrics including random selection, K-means, Kernel K-means, and Agglomerative Clustering. We show the performance of the initial condensed dataset and the final condensed dataset of 5 runs in Table 5, from which the following observations could be made. (1) The performance of different initialized condensed datasets is improved. This indicates that CondTSC is effective and could improve the performance of various initializations. (2) CondTSC utilizes the

K-means method, which gets relatively good performance. (3) Even with the random selection initialization, CondTSC still produces a comparable condensed dataset. This indicates that CondTSC is an initialization-independent framework.

6 CONCLUSION

In this work, we propose a novel framework that incorporates the time domain and the frequency domain for the time series dataset condensation task. The proposed framework CondTSC has three novel modules, and each module is designed with different purposes and contributes to the final performance. Extensive experiments verify the effectiveness of our proposed framework. In our future work, we will aim to further analyze condensed data and improve the performance of CondTSC.

ACKNOWLEDGMENT

REFERENCES

[1] KwangHoon An, Myung Jong Kim, Kristin Teplansky, Jordan R Green, Thomas F Campbell, Yana Yunusova, Daragh Heitzman, and Jun Wang. 2018. Automatic Early Detection of Amyotrophic Lateral Sclerosis from Intelligible Speech Using Convolutional Neural Networks.. In *Interspeech*. 1913–1917.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

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

[4] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4750–4759.

[5] Yanping Chen, Adena Why, Gustavo Batista, Agenor Mafra-Neto, and Eamonn Keogh. 2014. Flying insect classification with inexpensive sensors. *Journal of insect behavior* 27 (2014), 657–677.

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

[7] Giacomo Chiarot and Claudio Silvestri. 2023. Time series compression survey. *Comput. Surveys* 55, 10 (2023), 1–32.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555 [cs.NE]

[9] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. 2022. Scaling up dataset distillation to imagenet-1k with constant memory. *arXiv preprint arXiv:2211.10586* (2022).

[10] Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* (2016).

[11] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica* 6, 6 (2019), 1293–1305.

[12] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping Chen, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[13] Zhiwei Deng and Olga Russakovsky. 2022. Remember the past: Distilling datasets into addressable memories for neural networks. *Advances in Neural Information Processing Systems* 35 (2022), 34391–34404.

[14] Jiawei Du, Yidi Jiang, Vincent YF Tan, Joey Tianyi Zhou, and Haizhou Li. 2023. Minimizing the accumulated trajectory error to improve dataset distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3749–3758.

[15] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21. http://jmlr.org/papers/v20/18-598.html

[16] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research* 20, 1 (2019), 1997–2017.

[17] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation* 101, 23 (2000), e215–e220.

[18] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.

[19] Vibhor Gupta, Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2021. Continual learning for multivariate time series tasks with variable input dimensions. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 161–170.

[20] Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. 2019. Multitask learning and benchmarking with clinical time series data. *Scientific Data* 6, 1 (jun 2019). https://doi.org/10.1038/s41597-019-0103-9

[21] Nasir Hayat, Krzysztof J Geras, and Farah E Shamout. 2022. MedFuse: Multi-modal fusion with clinical time-series data and chest X-ray images. In *Machine Learning for Healthcare Conference*. PMLR, 479–503.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[24] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167 [cs.LG]

[25] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. 2022. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 720–730.

[26] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580* (2021).

[27] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. 2022. Dataset condensation via efficient synthetic-data parameterization. In *International Conference on Machine Learning*. PMLR, 11102–11118.

[28] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. 2022. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications* 197 (2022), 116659.

[29] Hae Beom Lee, Dong Bok Lee, and Sung Ju Hwang. 2022. Dataset condensation with latent space knowledge factorization and sharing. *arXiv preprint arXiv:2208.10494* (2022).

[30] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. 2022. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*. PMLR, 12352–12364.

[31] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. 2016. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In *PHM Society European Conference*, Vol. 3.

[32] Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. 2022. Dataset distillation using parameter pruning. *arXiv preprint arXiv:2209.14609* (2022).

[33] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[34] Chumeng Liang, Zherui Huang, Yicheng Liu, Zhanyu Liu, Guanjie Zheng, Hanyuan Shi, Kan Wu, Yuhao Du, Fuliang Li, and Zhenhui Li. 2023. CBLab: Supporting the Training of Large-scale Traffic Control Policies with Scalable Traffic Simulation. arXiv:2210.00896 [physics.soc-ph]

[35] Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. 2022. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697* (2022).

[36] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. 2022. Dataset distillation via factorization. *Advances in Neural Information Processing Systems* 35 (2022), 1100–1113.

[37] Zhanyu Liu, Jianrong Ding, and Guanjie Zheng. 2024. Frequency Enhanced Pre-training for Cross-city Few-shot Traffic Forecasting. arXiv:2406.02614 [cs.LG]
[38] Zhanyu Liu, Chumeng Liang, Guanjie Zheng, and Hua Wei. 2023. FDTI: Fine-grained Deep Traffic Inference with Roadnet-enriched Graph. arXiv preprint arXiv:2306.10945 (2023).
[39] Zhanyu Liu, Guanjie Zheng, and Yanwei Yu. 2023. Cross-city Few-Shot Traffic Forecasting via Traffic Pattern Bank. arXiv preprint arXiv:2308.09727 (2023).
[40] Zhanyu Liu, Guanjie Zheng, and Yanwei Yu. 2024. Multi-scale Traffic Pattern Bank for Cross-city Few-shot Traffic Forecasting. arXiv preprint arXiv:2402.00397 (2024).
[41] Noel Loo, Ramin Hasani, Mathias Lechner, and Daniela Rus. 2023. Dataset Distillation with Convexified Implicit Gradients. arXiv preprint arXiv:2302.06755 (2023).
[42] Alice Marascu, Pascal Pompey, Eric Bouillet, Michael Wurst, Olivier Verscheure, Martin Grund, and Philippe Cudre-Mauroux. 2014. TRISTAN: Real-time analytics on massive time series using sparse dictionary compression. In 2014 IEEE International Conference on Big Data (Big Data). IEEE, 291–300.
[43] Hussein Sh Mogahed and Alexey G Yakunin. 2018. Development of a lossless data compression algorithm for multichannel environmental monitoring systems. In 2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE). IEEE, 483–486.
[44] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10). 807–814.
[45] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. 2020. Dataset meta-learning from kernel ridge-regression. arXiv preprint arXiv:2011.00050 (2020).
[46] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. 2021. Dataset distillation with infinitely wide convolutional networks. Advances in Neural Information Processing Systems 34 (2021), 5186–5198.
[47] Hojjat Rakhshani, Hassan Ismail Fawaz, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévilliers, and Pierre-Alain Muller. 2020. Neural architecture search for time series classification. In 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 1–8.
[48] Noveen Sachdeva, Mehak Preet Dhaliwal, Carole-Jean Wu, and Julian McAuley. 2022. Infinite Recommendation Networks: A Data-Centric Approach. arXiv:2206.02626 [cs.IR]
[49] Ahmad Sajedi, Samir Khaki, Ehsan Amjadian, Lucy Z Liu, Yuri A Lawryshyn, and Konstantinos N Plataniotis. 2023. Datadam: Efficient dataset distillation with attention matching. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 17097–17107.
[50] Ilya O Tolstikhin, Bharath K Sriperumbudur, and Bernhard Schölkopf. 2016. Minimax estimation of maximum mean discrepancy with radial kernels. Advances in Neural Information Processing Systems 29 (2016).
[51] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. Instance Normalization: The Missing Ingredient for Fast Stylization. arXiv:1607.08022 [cs.CV]
[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).
[53] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. 2022. Cafe: Learning to condense dataset by aligning features. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 12196–12205.
[54] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. 2020. Dataset Distillation. arXiv:1811.10959 [cs.LG]
[55] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. 2020. Time series data augmentation for deep learning: A survey. arXiv preprint arXiv:2002.12478 (2020).
[56] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. arXiv preprint arXiv:2202.01575 (2022).
[57] Ling Yang and Shenda Hong. 2022. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In International Conference on Machine Learning. PMLR, 25038–25054.
[58] Lei Zhang, Jie Zhang, Bowen Lei, Subhabrata Mukherjee, Xiang Pan, Bo Zhao, Caiwen Ding, Yao Li, and Dongkuan Xu. 2023. Accelerating dataset distillation via model augmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 11950–11959.
[59] Wenrui Zhang, Ling Yang, Shijia Geng, and Shenda Hong. 2022. Cross reconstruction transformer for self-supervised time series representation learning. arXiv preprint arXiv:2205.09928 (2022).
[60] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. 2022. Self-supervised contrastive pre-training for time series via time-frequency consistency. Advances in Neural Information Processing Systems 35 (2022), 3988–4003.
[61] Bo Zhao and Hakan Bilen. 2021. Dataset condensation with differentiable siamese augmentation. In International Conference on Machine Learning. PMLR, 12674–12685.
[62] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 6514–6523.
[63] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. Dataset condensation with gradient matching. arXiv preprint arXiv:2006.05929 (2020).
[64] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In International Conference on Machine Learning. PMLR, 27268–27286.
[65] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. 2022. Dataset distillation using neural feature regression. Advances in Neural Information Processing Systems 35 (2022), 9813–9827.

A SURROGATE OBJECTIVE MATCHING

In this section, we introduce the details of objective matching. To learn the condensed synthetic dataset $\mathcal{S}$, we would match some surrogate objectives of $\mathcal{S}$ and $\mathcal{T}$. The aim is to ensure that the synthetic dataset $\mathcal{S}$ accurately captures the dynamics present in the real dataset $\mathcal{T}$. Consequently, a network trained using the synthetic dataset can exhibit comparable performance to a network trained using the complete real dataset. Here, we introduce two types of surrogate objectives used in this paper.

A.1 Gradient Matching

The training gradient for a given initial model trained by $\mathcal{S}$ and $\mathcal{T}$ is a good surrogate objective. By matching the training gradients of $\mathcal{S}$ and $\mathcal{T}$, we can effectively align the training dynamics of the synthetic dataset with that of the real dataset [14, 61, 63]. The single-step gradient matching objective is shown as follows.

$$S^* = \arg \min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim p(\theta)} [\mathbf{D}(\nabla_{\theta_0} \ell(f_{\theta_0}(\mathcal{S})), \nabla_{\theta_0} \ell(f_{\theta_0}(\mathcal{T})))] \quad (18)$$

Here $\mathbf{D}(\cdot, \cdot)$ is the distance function such as cosine distance and $p(\theta)$ is the distribution of the network parameter. $\nabla_{\theta_0} \ell(f_{\theta_0}(\mathcal{S}))$ is the gradient of the training loss $\ell(f_{\theta_0}(\mathcal{S}))$ of network f parameterized by θ_0 on dataset $\mathcal{S}$. This equation aims to find the synthetic data $\mathcal{S}$ that has the most similar gradient to real data $\mathcal{T}$ with a given initial network f_{θ_0}. In practice, the synthetic data $\mathcal{S}$ would be updated by many networks f_{θ_0} to ensure the ability of generalization.

A.2 Distribution Matching

Another surrogate objective is the data distribution generated by a given model based on $\mathcal{S}$ and $\mathcal{T}$. Its essential aim is that the network trained by $\mathcal{S}$ could generate similar embeddings to those from the network trained by $\mathcal{T}$. Consequently, the data distributions of $\mathcal{S}$ and $\mathcal{T}$ are similar, which makes $\mathcal{S}$ a condensed dataset from $\mathcal{T}$. The core idea of distribution matching is reducing the empirical estimate of the Maximum Mean Discrepancy [62], which is shown as follows.

$$S^* = \arg \min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim p(\theta)} [||\frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} f_{\theta_0}(s_i) - \frac{1}{|\mathcal{T}|} \sum_{j=1}^{|\mathcal{T}|} f_{\theta_0}(x_j)||^2] \quad (19)$$

Here $p(\theta)$ is the distribution of the network parameter and $f_{\theta_0}(\cdot)$ is the network f parameterized by θ_0 which outputs the embedding of the input. In practice, the calculation of Eq. 19 is usually between a batch of $\mathcal{S}$ and a batch of $\mathcal{T}$. This equation is essentially finding the synthetic data $\mathcal{S}^*$ that has the most similar distribution to the full data $\mathcal{T}$ in the network embedding space f_θ.

By matching the surrogate objectives, the models trained by the condensed synthetic data could be similar to the models trained

Table 6: Cross-Architecture accuracy(%) performance. We train the synthetic data in one network structure and evaluate the synthetic data in another network structure. The experiment is conducted in the HAR dataset with $spc = 5$. DD is not selected as one of the baselines due to its bad performance. The best result is highlighted in bold and grey in each column.

Train Model	Methods	MLP	CNNBN	CNNIN	ResNet18	TCN	LSTM	Transformer	GRU
MLP	DC	**60.64±4.46**	67.98±3.66	55.08±4.65	31.35±8.44	59.85±7.64	47.50±7.92	72.12±4.26	55.85±7.05
	DSA	57.60±5.98	67.89±3.35	53.02±2.56	30.89±5.24	67.23±4.03	45.59±12.5	73.55±3.62	**57.01±7.42**
	DM	48.79±0.97	63.75±3.01	59.16±1.01	51.68±2.70	61.39±2.89	**48.60±0.95**	72.04±2.71	51.43±2.81
	MTT	51.17±0.79	63.36±1.29	58.51±1.47	59.66±2.43	50.97±3.76	43.50±5.42	69.67±2.52	49.10±1.44
	IDC	47.67±1.75	54.61±7.40	49.76±8.28	50.93±7.32	49.67±3.32	42.44±2.12	65.88±2.68	50.40±1.09
	HaBa	40.40±6.31	41.35±9.39	40.25±5.13	34.06±9.57	39.52±7.12	35.42±3.11	49.41±5.62	25.74±4.53
	CondTSC	52.53±3.71	**73.95±1.96**	**65.40±3.43**	**75.55±1.75**	**67.49±1.58**	45.90±2.04	**73.97±1.10**	51.63±1.26
CNNBN	DC	50.60±1.92	65.83±3.97	42.07±6.85	22.74±7.06	70.89±5.25	45.39±9.78	73.39±4.24	54.18±3.85
	DSA	48.81±1.27	65.16±3.67	52.03±5.96	29.40±7.94	71.76±4.54	50.43±8.53	75.84±3.93	59.97±3.45
	DM	49.21±1.02	70.82±4.63	65.57±3.09	64.92±2.17	65.68±3.34	45.91±6.34	74.57±3.03	56.13±7.49
	MTT	50.38±1.16	77.66±3.86	58.95±1.86	68.73±4.42	67.86±2.84	49.71±7.36	77.55±1.58	56.35±3.92
	IDC	47.58±1.93	54.41±5.02	45.02±5.67	51.03±4.42	45.36±5.80	40.60±3.50	56.83±7.86	40.60±3.50
	HaBa	44.90±1.50	56.22±5.56	50.25±2.27	54.21±7.04	53.84±2.56	39.98±5.58	61.56±6.56	47.60±3.63
	CondTSC	**51.92±3.36**	**82.20±2.68**	**72.33±2.93**	**76.02±3.48**	**76.95±2.77**	**53.16±4.32**	**79.81±1.93**	**60.37±2.61**
CNNIN	DC	45.41±4.45	57.64±6.92	52.84±2.71	25.93±7.38	66.85±3.36	36.40±5.84	46.53±5.03	35.50±1.79
	DSA	51.27±2.19	61.01±4.43	53.08±2.56	27.43±8.02	66.94±3.29	48.78±10.1	74.58±2.86	**55.33±5.00**
	DM	47.37±2.30	57.87±2.33	57.13±4.08	58.24±2.67	56.77±3.19	43.72±8.05	57.72±2.70	39.81±4.24
	MTT	51.22±1.45	64.61±1.94	58.20±2.10	52.13±2.56	64.30±1.57	46.94±9.39	73.59±1.84	48.84±4.02
	IDC	43.20±0.93	52.57±6.55	48.00±4.09	52.13±5.33	44.56±6.41	45.47±4.93	55.67±3.62	41.75±2.84
	HaBa	49.54±1.30	27.35±6.46	59.44±2.74	29.41±7.66	46.89±6.47	37.12±2.19	53.16±7.89	41.47±4.09
	CondTSC	**54.52±3.33**	**74.37±2.56**	**71.60±2.58**	**70.42±3.94**	**67.09±1.38**	**49.12±4.29**	**75.64±2.10**	49.23±1.04
ResNet18	DC	17.22±2.21	59.26±4.03	52.78±3.64	30.07±6.06	64.32±1.94	21.68±7.34	69.90±6.81	18.25±3.10
	DSA	46.72±4.31	59.91±4.27	51.93±2.99	32.36±8.29	66.25±5.69	**47.65±10.5**	70.95±5.27	38.33±6.82
	DM	35.80±1.94	49.01±2.86	49.95±2.11	50.16±2.61	43.28±4.27	39.32±9.09	34.01±7.36	43.64±4.00
	MTT	49.75±0.60	67.42±1.55	57.87±1.90	65.43±2.50	50.82±3.62	43.20±2.02	73.50±2.21	44.28±3.33
	IDC	45.11±1.16	44.27±5.39	28.35±6.03	42.35±6.28	38.59±4.08	42.79±7.10	49.46±4.44	42.45±6.20
	HaBa	31.04±4.93	19.92±5.23	33.59±6.04	18.09±1.45	30.25±6.32	35.00±9.14	29.77±8.47	30.66±5.86
	CondTSC	**53.68±2.02**	**76.53±1.46**	**70.08±3.74**	**77.67±2.09**	**69.85±0.95**	44.56±3.79	**77.24±2.38**	**45.02±1.69**
TCN	DC	47.75±1.93	59.42±3.01	53.97±3.38	31.07±7.48	64.02±4.58	48.85±11.7	71.76±3.12	54.04±5.62
	DSA	47.71±1.63	60.92±3.42	55.30±3.17	30.14±6.29	64.31±4.00	45.03±13.2	70.15±5.12	**54.11±3.40**
	DM	48.21±1.52	59.66±6.08	57.17±1.95	50.98±2.61	63.67±2.66	48.59±10.3	70.08±5.21	51.48±3.44
	MTT	50.59±0.68	65.70±1.50	59.25±1.50	61.65±2.25	58.33±2.82	47.29±5.12	70.48±2.01	50.46±2.88
	IDC	48.29±1.77	58.92±3.92	51.82±1.91	57.12±7.36	49.91±2.47	47.59±1.40	67.11±4.49	48.42±1.48
	HaBa	36.76±2.41	34.68±5.41	35.61±5.12	35.91±7.07	25.53±5.54	29.85±3.03	39.54±7.14	27.20±4.95
	CondTSC	**52.64±2.95**	**71.01±3.54**	**66.45±2.49**	**67.96±4.04**	**65.67±1.74**	**48.88±2.63**	**71.93±3.47**	51.29±1.34

by the full real data. Consequently, training a model with the condensed synthetic data results in minimal performance degradation and is much more efficient simultaneously.

B EXPERIMENT DETAILS AND EFFICIENCY

Network Structure: In line with prior research [13, 25–27, 36, 41, 46, 54, 61–63, 65] that utilize simple deep neural networks to verify the effectiveness of the proposed method, we train the synthetic data based on a wide range of basic network architectures, including multi-layer perception (MLP), convolution neural network (CNN), ResNet [22], temporal convolutional network (TCN) [3], LSTM [23], Transformer [52], gate recurrent unit (GRU) [8]. For MLP, we use a 3-layer MLP with a hidden size of 128 and a ReLU [44] activation function. For CNN, we use a 3-layer CNN with hidden channels of 128, kernel size of 3x1, maxpooling function, and ReLU activation function. Specifically, following the previous studies [65], we denote CNN with batch normalization [24] and instance normalization [51] as CNNBN and CNNIN respectively. For ResNet, we use the default network structure of Resnet18. For TCN, we use a 1-layer TCN with a kernel size of 64x1. For LSTM, we use a 1-layer LSTM with a hidden size of 100. For Transformer, we use a 1-layer Transformer with a hidden size of 64. For GRU, we use a 1-layer GRU with a hidden size of 100. These networks except ResNet have a similar size of model parameters, which is approximately 30,000.

Hyper-parameter Searching of Baselines: We use the following hyper-parameter searching strategy to fairly evaluate baseline methods. It is hard to explore the same grid search space due to the time limit (one baseline might take dozens of days to search). Consequently, for each method, we first search for a good range for each parameter in the same log space while fixing other parameters. Then we use the searched range to do a grid search as follows. we choose the epoch with the highest validation accuracy for testing. Consequently, we believe the comparisons are fair.

- **DD:** Training epochs is 100 and evaluate every 2 epochs. The learning rates of the synthetic data are searched in $[10^{-3}, 10^{-2},$
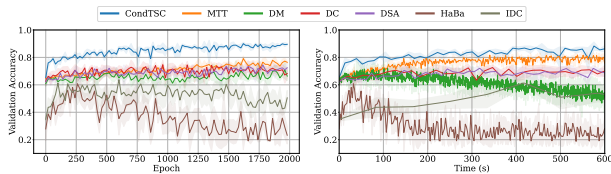
Figure 6: The training curve on the HAR dataset with $spc = 5$. We show the training epoch and training time as the x-axis respectively. The mean and standard deviation of validation accuracy of 5 runs are shown.

10^{-1}, 1, 10]. The distilled steps are searched in [1, 5, 10, 20, 30, 40], and the distilled epochs are searched in [1, 2, 3, 4, 5].

- **DC, DSA, DM:** Training epochs is 2000 and evaluate every 100 epochs. The feature learning rate are searched in [10^{-3}, 10^{-2}, 10^{-1}, 1, 10], and the net learning rate is searched in [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000].
- **MTT:** Training epochs is 2000 and evaluate every 100 epochs. The learning rate are searched in [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000]. The α learning rate is searched in [10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}].
- **IDC:** Training epochs is 100 and evaluate every 2 epochs. The matching surrogate objective is set to feature matching. The multi-formation factor is searched in [1, 2, 3], and the learning rate is searched in [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000].
- **HaBa:** Training epochs is 2000 and evaluate every 100 epochs. The learning rates of features and styles are searched in [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000]. The learning rate of α is searched in [10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}].

All of the hyper-parameters that are not mentioned above are kept the same as in the original paper. Moreover, we show the training curve of different methods with respect to training epochs and time in Fig. 6. The experiment is conducted on the HAR dataset with $spc = 5$ and the mean and standard deviation of validation accuracy of 5 runs are shown. We can observe that it only costs 10 minutes to train CondTSC to converge. Moreover, We can observe that CondTSC achieves the best training accuracy no matter with the same training epoch or the same training time. Furthermore, the training curve indicates all of the baseline methods are converged and we choose the epoch with the highest validation accuracy for testing. Consequently, we believe the comparisons are fair.

C CROSS ARCHITECTURE PERFORMANCE

In the dataset condensation task, the ability of cross-architecture generalization is one of the important indicators for evaluating methods [54, 63]. This is because when training the synthetic data $\mathcal{S}$, we usually utilize only one network architecture [4, 27, 29, 30, 61], i.e., $p(\theta)$ is from a parameter space of single network structure f. Consequently, the learned synthetic data performs well when

training the network f but might perform satisfactorily when training the network with another structure g. To verify the ability to cross-architecture generalization of the learned synthetic data of CondTSC, we conduct extensive experiments on the HAR dataset with $spc = 5$. We train the synthetic data $\mathcal{S}$ with one network structure and evaluate $\mathcal{S}$ with another network structure. The result is shown in Table 6.

From Table 6, we could observe that: (1) Overall, CondTSC performs best in the cross-architecture experiment. It achieves stable and outstanding performance, especially in the CNN-based and Transformer-based models. This indicates that CondTSC learns a robust and easy-to-generalize synthetic dataset and is resistant to overfitting one network architecture. (2) In some simple network architectures such as MLP and RNN, the single-step gradient matching methods and distribution matching methods, i.e., DC, DSA, and DM perform relatively well due to their simple design. However, their performance is unstable, which indicates that they could not learn a robust synthetic dataset and lack the ability to generalize to other network architectures.

D PARAMETER SENSITIVITY

In this section, we would like to show the performance of CondTSC on different hyper-parameters, which throws light on the analysis of CondTSC. We evaluate the performance of different values of hyper-parameters on all five datasets, and the results are shown in Fig. 7. (1) We evaluate the performance of different loss scaler λ, which balances the gradient matching loss $\mathcal{L}^{grad}$ and the embedding matching loss $\mathcal{L}^{emb}$. We test a range of log-scale values and observe setting $\lambda = 1$ is best for all datasets. (2) We evaluate the performance of different learning rates for the synthetic data $lr_{\mathcal{S}}$. We also test a range of log-scale values for $lr_{\mathcal{S}}$. In previous studies on image dataset condensation [4, 36, 61, 62], the value of $lr_{\mathcal{S}}$ are usually set to a large value, i.e., $lr_{\mathcal{S}} \geq 1e3$. However, we could observe that setting $lr_{\mathcal{S}} = 1$ is best for our time series dataset condensation task. This demonstrates that the time series dataset condensation task has essential differences from other dataset condensation tasks, and designing special techniques for time series dataset condensation is necessary. (3) We also evaluate the performance of different values of N and M, which is the number of training iterations in the Dual Domain Training module for the synthetic data $\mathcal{S}$ and the real data $\mathcal{T}$ respectively. From our analysis in the Method section, we should set $N \ll M$ to avoid overfitting to the real data. Here, the experiment results show that the performance is relatively good when $N \approx 10$ and $M \approx 1000$ and the performance could be unsatisfactory when $M \approx N$. This further validates our analysis.

Overall, we conducted extensive experiments on evaluating the performance of different values of hyper-parameters. We find that setting the loss scaler $\lambda = 1$, the learning rate of synthetic data $lr_{\mathcal{S}} = 1$, and training iterations $N \ll M$ is a good hyper-parameter setting.
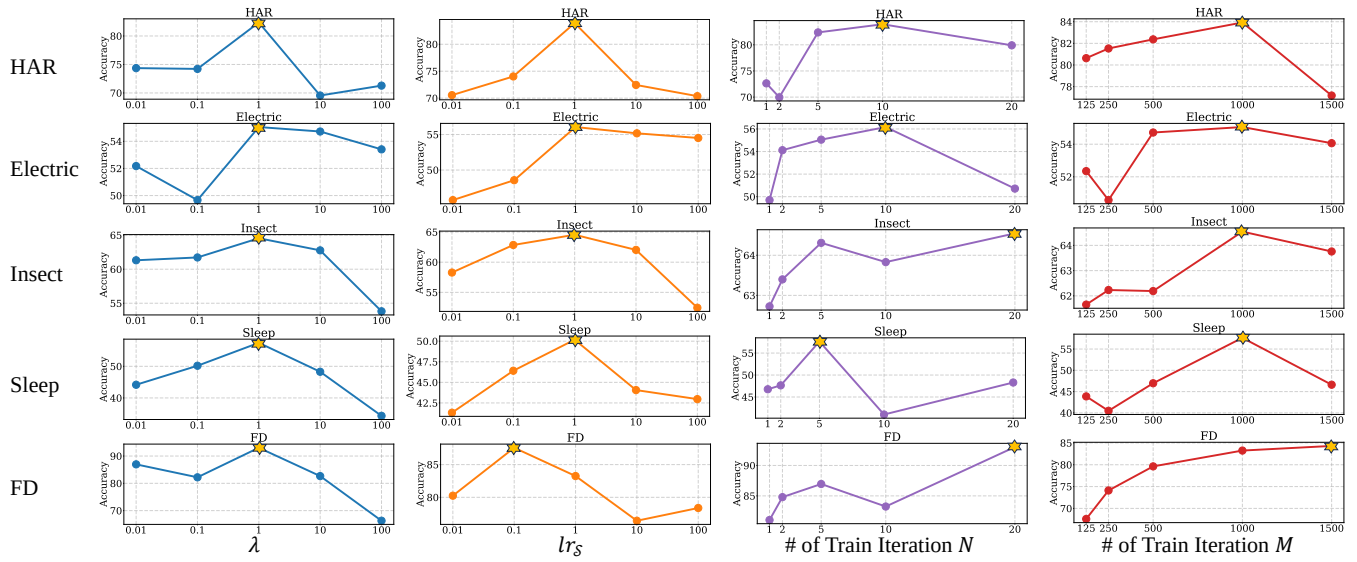
Figure 7: The parameter sensitivity analysis. Different values of hyper-parameters are evaluated on all five datasets. We evaluate four hyper-parameters, including loss scaler λ, the learning rate of the synthetic data $lr_{\mathcal{S}}$, the number of train iterations for the synthetic data N, the number of train iterations for the real data M. The yellow star indicates the hyper-parameter value with the highest test accuracy.