A APPENDIX

Table 8: Summary of Notation

Symbol	Definition
w	Sliding window size
TRE	Trend
SEA	Seasonality
T	Time series
t_i	Observation in T at the i-th timestamp
$\mathcal{T}$	Time series dataset
M	Length of $\mathcal{T}$
$\mathcal{S}$	Condensed time series dataset
N	Length of $\mathcal{S}$
θ	Parameters learned on $\mathcal{T}$
$\theta^{\mathcal{S}}$	Parameters learned on $\mathcal{S}$
T^c_{input}	Channel-independent time series
$\mathcal{B}$	Expert buffer

A.1 Preliminaries

A.1.1 Notation. Table 8 lists notation used throughout the paper.

A.2 Methodology

Algorithm 3: The TimeDC Framework

Input: A buffer $\mathcal{B}$ with a set of pre-trained trajectories on the original time series dataset $\mathcal{T}$ parameterized by $\{\Theta^k_{\mathcal{T}}\}_{k=1}^K$; numbers of two-fold matching steps: T_0; f_{θ^S} training steps: T_1; initialized condensed time series dataset: $\mathcal{S}$.

Output: Optimized condensed time series dataset: $\mathcal{S}$.

1 **while** $\eta < T_0$ **do**
2 Sample a pre-trained trajectory in $\mathcal{B}$ based on the curriculum trajectory query (see Algorithm 2);
3 calculate the L_{tmm} according to Equation 12);
4 **for** $\gamma < T_1$ **do**
5 Train f_{θ^S} via gradient matching and frequency matching (see Equation 14);
6 $\widetilde{\theta}_{\gamma+1} \leftarrow \widetilde{\theta}_\gamma - \alpha\nabla(\mathcal{L}(f_{\theta^S}, \mathcal{S}) + L_{Fre})$, where α is the learning rate;
7 Update the condensed time series dataset $\mathcal{S}$ according to Equation 19;
8 **return** $\mathcal{S}$

A.2.1 Algorithm. The whole process of TimeDC is shown in Algorithm 3, where lines 2–3 cover the curriculum trajectory query and matching, lines 4–6 cover the training process of f_{θ^S} and frequency matching, and line 7 covers the optimization of $\mathcal{S}$.

A.3 Experiment

A.3.1 Dataset Statistics. The forecasting and classification dataset statistics are provided in Tables 9 and 10, respectively.

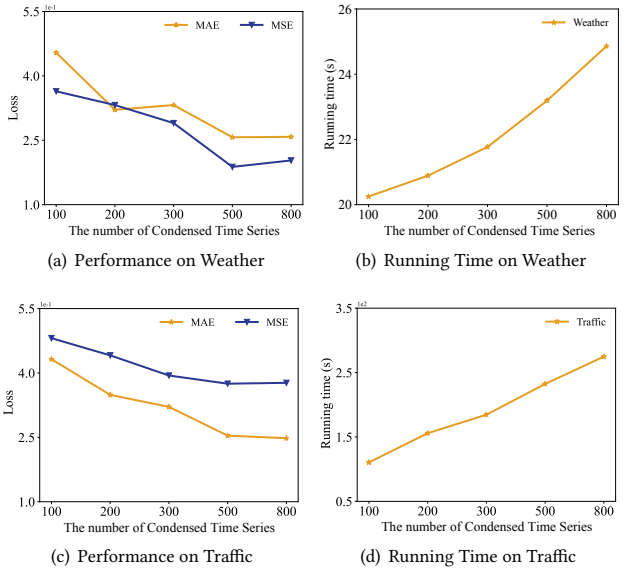
A.3.2 Trade-off between Performance and Efficiency. To study the effect of the number of condensed time series on performance and running time, we conduct experiments with 100, 200, 300, 500, and 800 condensed time series on the Weather and Traffic datasets. The results, shown in Figure 10, indicate that the performance initially drops and then either stabilizes (Figure 10(a)) or increase slightly (Figure 10(c)). In addition, as the number of condensed time series increases, the running time decreases considerably. Generally, the results show that model performance improves with an increase in condensed time series data, but at the cost of an increased training time. When the number of condensed time series is set to 500, TimeDC achieves outstanding performance with an acceptable training time, making 500 an appropriate setting for balancing performance and efficiency.

Table 9: Statistics of Forecasting Datasets

Dataset	Feature	Time step	Granularity
Weather	21	52696	10 minutes
Traffic	862	17544	1 hour
Electricity	321	26304	1 hour
ETTh1 & ETTh2	7	17420	1 hour
ETTm1 & ETTm2	7	69680	15 minutes

Table 10: Statistics of Classification Datasets

Dataset	Class	Length	Type
ECG200	2	96	Electrocardiography
ElectricDevices	7	96	Device
FordB	2	500	Sensor



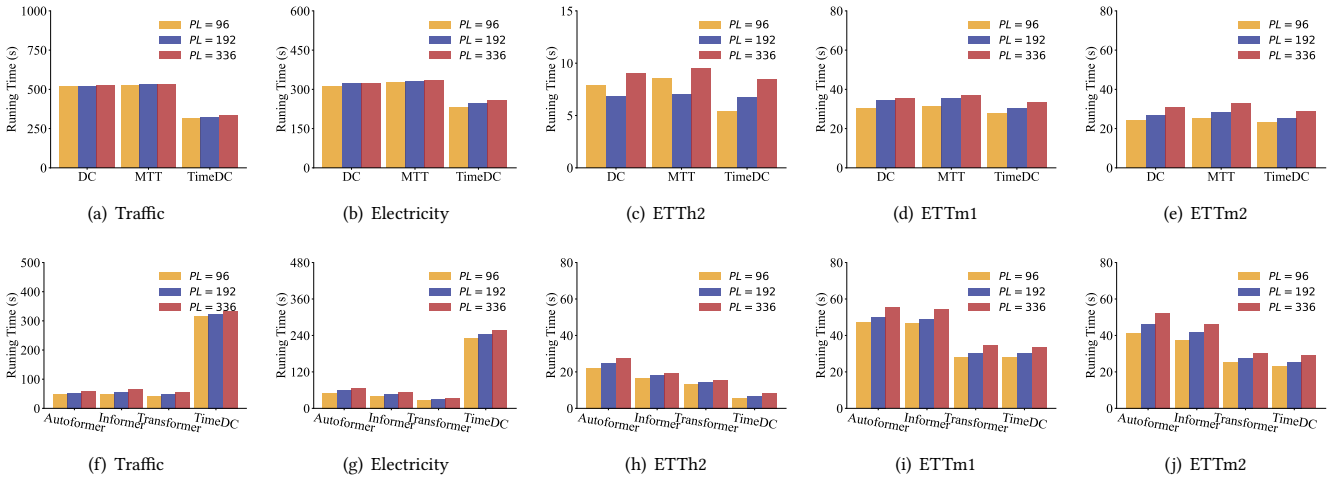
(a) Performance on Weather (b) Running Time on Weather

(c) Performance on Traffic (d) Running Time on Traffic

Figure 10: Trade-off between Performance and Efficiency on Two Datasets

Table 11: Comparison between Autoformer$_r$ and TimeDC on Weather ($PL = 96$)

Metric	MAE		RMSE	
Method	Autoformer$_r$	TimeDC	Autoformer$_r$	TimeDC
$\mathcal{B}_0$	0.603	0.456	0.625	0.477
$\mathcal{B}_1$	0.835	0.512	1.055	0.533
$\mathcal{I}$	0.638	0.656	0.687	0.512



(a) Traffic (b) Electricity (c) ETTh2 (d) ETTm1 (e) ETTm2

(f) Traffic (g) Electricity (h) ETTh2 (i) ETTm1 (j) ETTm2

Figure 11: Running Time Comparison

Table 12: Training Time of TimeDC and Training Time on Original Datasets (s/epoch)

Dataset	Condensed Dataset	Original Dataset
Weather	22.39	35.26
Traffic	232.34	346.76
Electricity	314.56	522.85
ETTh1	14.38	20.43
ETTh2	5.43	8.95
ETTm1	27.83	35.75
ETTm2	23.15	30.46

A.3.3 Performance Comparison on Streaming Data. To assess more comprehensively the effectiveness of TimeDC, we compare it with the replay-based method Autoformer$_r$ on Weather, which adopts an explicit buffer to store a random subset of the base set $\mathcal{B}$. When the incremental set $\mathcal{I}$ arrives, we randomly select 500 samples from the buffer and then fuse them with the new data to update the model parameters. The results are shown in Table 11, where TimeDC outperforms Autoformer$_r$, which shows a simple replay-based method is insufficient for time series streaming learning due to the concept drift problems. The results show that TimeDC achieves relatively stable performance as well as notably better performance on the base and incremental sets in five out of six cases.

A.3.4 Training Time on Dataset Condensation and Original Datasets. We compare the training time of TimeDC and training time on original datasets based on the stacked *TSOperators* in terms of an epoch, as shown in Table 12. The results indicate that the training time of TimeDC is significantly lower than training on the original dataset. For example, the training time of TimeDC for dataset condensation is reduced by 39.84% compared to that of training a model on the original dataset on Electricity. Additionally, Table 1 shows that the model trained on the condensed dataset performs comparably to the model trained on the original dataset. Thus, TimeDC not only reduces the training time to achieve convergence for dataset condensation but also maintains good performance, showing the efficiency and practicality of time series dataset condensation, especially for resource-intensive tasks.

A.3.5 Running Time Efficiecy. As efficiency is important in dataset condensation to enable scalability, especially on resource-constrained edge computing devices, we study the training time (of an epoch) for the condensation methods and different architectures. We conduct experiments on five datasets, i.e., Traffic, Electricity, ETTh2, ETTm1, and ETTm2, as shown in Figure 11. Overall, TimeDC consumes the least training time in most cases.

We study the training time (of an epoch) for the condensation methods. Figures 11(a)–11(e) report the training time on Traffic, Electricity, ETTh2, ETTm1, and ETTm2. We see that the training time of TimeDC is below those of DC and MTT, which is largely because of the expert buffer in the CT2M module that stores precomputed trajectories. This indicates the feasibility of TimeDC for model deployment in large time series dataset reduction scenarios.

We also study training time across different network architectures on Traffic, Electricity, ETTh2, ETTm1, and ETTm2—see Figures 11(f)–11(j). Here, TimeDC consumes the least training time in most cases. TimeDC is faster than the other methods due to its patching mechanism, which reduces the complexity of the self-attention mechanism through input data simplification. However, TimeDC consumes more training time on Traffic and Electricity. This is because Traffic and Electricity have many more features than the other datasets—862 versus 321 features. The proposed *TSOperators* require more training time to learn these features simultaneously because of the channel-independent mechanism. Nonetheless, TimeDC achieves much better performance on Traffic and Electricity, indicating its ability to learn correlations across different channels.

A.3.6 Training Time of TimeDC and Its Variants. We report the training time of TimeDC and its variants in Table 13. The training time of *w/o_patch* is much lower than those of the other variants, primarily due to the patch construction and storage in the patching mechanism. However, the patching mechanism enables the

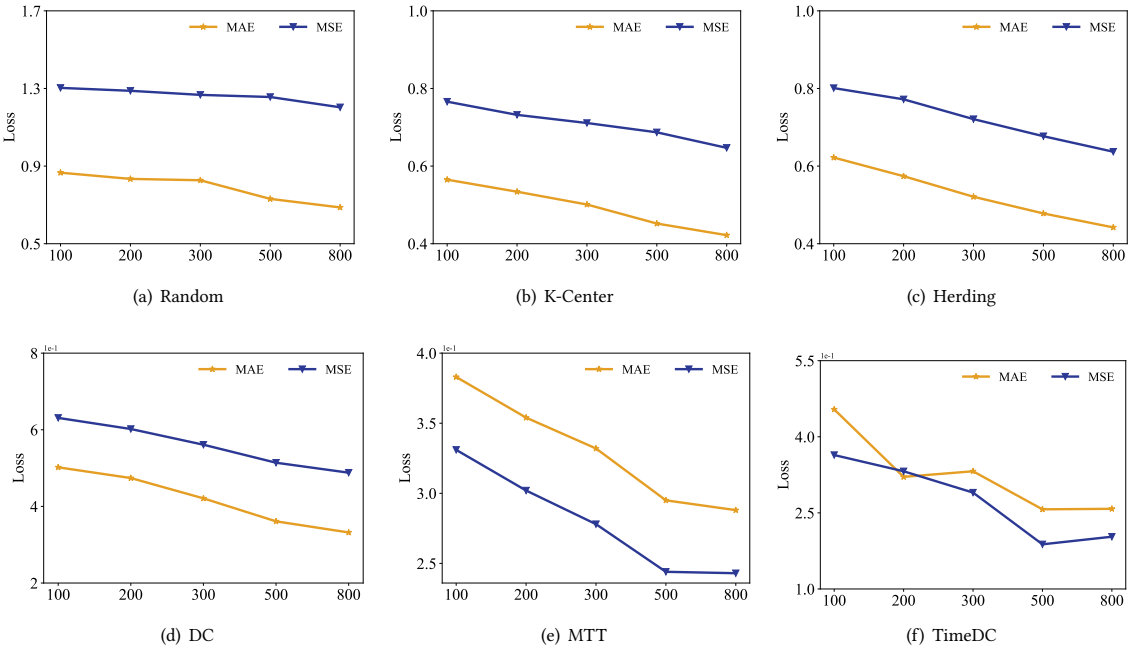


Figure 12: Effect of the Size of Condensed TS Datasets across Different Methods on Weather

modeling of local semantics, which enables the model to process longer historical sequences to improve the feature extraction, thus enabling better model performance.

Table 13: Training Time of TimeDC and Its Variants (s/epoch)

Dataset	*w/o_Patch*	*w/o_DDFM*	*w/o_CT^{2}M*	TimeDC
Weather	20.34	22.44	33.4	22.84
Traffic	50.45	278.74	510.4	314.56
Electricity	60.43	201.45	331.32	232.34
ETTh1	15.68	16.45	25.64	18.75

A.3.7 Effect of the Size of Condensed TS Datasets across Different Methods. We study the effect of the size of the condensed time series (TS) dataset across different methods in Figure 12. We observe that the performance curves drop significantly in most cases. This shows that the model performance improves with a larger condensed time series dataset. This increase in performance is attributed to the availability of more training data containing valuable knowledge. It is noteworthy that TimeDC shows a slight decrease in performance when using 800 condensed time series compared to 500 condensed time series on the Weather dataset. This may be because the patterns in these time series are relatively straightforward. Additional condensed time series data might introduce recurring patterns, thereby making the model overfit to these patterns and degrading performance on other data. For detailed results for other datasets, please refer to the repository at https://github.com/uestc-liuzq/STdistillation.

A.3.8 Dynamic Tensor Memory Cost. We conduct experiments to compare the memory used by the dynamic (online) tensor across DC, MTT, and TimeDC on all datasets in Table 14. TimeDC is able to significantly alleviate heavy online memory and computation costs thanks to the training trajectories precomputed offline.

Table 14: Dynamic Tensor Memory Cost on All Datasets

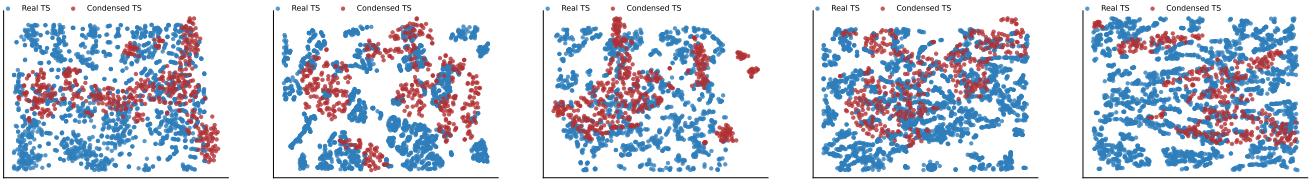
Dataset	DC	MTT	TimeDC
Weather	10.0 GB	8.9 GB	3.3 GB
Traffic	17.8 GB	13.7 GB	10.9 GB
Electricity	8.5 GB	932.5 MB	516.0 MB
ETTh1	1.9 GB	845.5 MB	280.9 MB
ETTh2	1.8 GB	932.5 MB	516.1 MB
ETTm1	1.8 GB	932.2 MB	515.9 MB
ETTm2	1.7 GB	932.5 MB	516.1 MB

A.3.9 The Size of Condensed Datasets on Seven Datasets. We report the condensed dataset size for each baseline and TimeDC in Table 15. We observe that the dataset sizes are similar across the different baselines because we fix the number of condensed time series at 500 for each method, resulting in minimal variation in the dataset size.

A.3.10 Case Study on Dataset Condensation. To further assess the effectiveness of TimeDC on synthesizing condensed time series datasets that cover the original time series distribution well, we show t-SNE graphs of the original time series dataset and the condensed time series dataset for Traffic, Electricity, ETTh2, ETTm1, and ETTm2. Figure 13 compares the dataset distributions, where blue and red dots represent the original (i.e., real) and condensed dataset, respectively. We randomly sample 500 time series from the original time series dataset for the visualization of the original time

Table 15: Condensed Dataset Size for Seven Datasets

Dataset	Random	K-Center	Herding	DC	MTT	TimeDC
Weather	40.3 MB	40.3 MB	40.3 MB	39.8 MB	38.6 MB	38.5 MB
Traffic	1.7 GB	1.7 GB	1.7 GB	1.7 MB	1.7 GB	1.7 GB
Electricity	316.3 MB	308.2 MB	316.3 MB	316.3 MB	310.8 MB	308.2 MB
ETTh1	13.4 MB	13.4 MB	13.4 MB	13.3 MB	13.4 MB	12.8 MB
ETTh2	13.3 MB	13.4 MB	13.4 MB	13.4 MB	13.4 MB	13.4 MB
ETTm1	13.4 MB	13.4 MB	13.4 MB	13.4 MB	13.4 MB	13.4 MB
ETTm2	13.4 MB	13.4 MB	13.4 MB	13.4 MB	13.4 MB	13.4 MB



(a) Traffic (b) Electricity (c) ETTh2 (d) ETTm1 (e) ETTm2

Figure 13: Dataset Distribution Comparison on Five Datasets

Table 16: Effect of Prediction Length on Weather

Prediction Length	Dynamic Tensor	Training Time
96	3.34 GB	20.82 s
192	3.41 GB	21.73 s
336	3.48 GB	22.39 s

Table 17: Effect of Computing Nodes

Computing Nodes	Dynamic Tensor	Training Time
1	8.5 GB	314.56 s
2	5.3 GB	224.76 s
3	4.3 GB	179.99 s
4	3.7 GB	147.78 s

series. We observe that the red dots are well integrated with the blue dots, indicating similarity in distribution between the original and condensed datasets. This indicates that the condensed dataset is of high quality and that the condensation method is effective.

A.3.11 Scalability. To assess the scalability of TimeDC, we conduct experiments to study the effect of different prediction lengths on Weather. The results are shown in Table 16. With an increase in the prediction length, TimeDC achieves similar dynamic tensor costs and training times, demonstrating its efficiency and scalability in terms of time series prediction length.

In addition, TimeDC scales with the input dimensionality (i.e., number of features). In particular, the increase in the dynamic tensor and storage costs of TimeDC is significantly smaller than the difference in the number of input features of various datasets in most cases (see Tables 3 and 6). For example, the number of features in Traffic (i.e., 862) is approximately 123 times that of ETTh1 (i.e., 7), but the dynamic tensor and storage costs of TimeDC on Traffic are 12 and 64 times those of ETTh1, respectively.

Moreover, we observe that the training costs of TimeDC decrease with the increase in the number of computing nodes (i.e., GPUs). For example, as the number of computing nodes increases from 1 to 4, the training time of TimeDC decreases on Electricity from 314.56 s/epoch to 147.78 s/epoch (see Table 17). Meanwhile, the utilization of the dynamic tensor on each GPU reduces from 8.5 GB to 3.7 GB, indicating the feasibility of training TimeDC across multiple smaller devices.

Further, TimeDC scales with the number of condensed time series. According to Figures 10(b) and 10(d), the running time of TimeDC increases approximately linearly with the number of time series.