

Supplemental Materials for “Revisiting PCA for Time Series Reduction in Temporal Dimension”

A DATA DESCRIPTION

The experimental data comprises 13 widely-used datasets from various domains, each distinguished by unique attributes:

- ETT (Zhou et al., 2021): The ETT dataset includes two hourly-level datasets (ETTh1 and ETTh2) and two 15-minute-level datasets (ETTM1 and ETTM2). Each dataset comprises seven oil and load features of electricity transformers spanning from July 2016 to July 2018. There are 7 variables for each dataset, with 17,420 time steps for ETTh and 69,680 time steps for ETTM. The series in these datasets exhibit strong periodicity. For univariate forecasting, only the “oil temperature” variable is used for training and testing.
- EthanolConcentration (Bagnall et al., 2018): The dataset comprises 544 time series formed by the raw spectra of water and ethanol solutions in authentic whisky bottles, with each series having a length of 1,751. Ethanol concentrations range from 35%, 38%, 40%, to 45%. The primary objective of this dataset is to ascertain the ethanol concentration (category) within each sample. As a multivariate dataset, each variable corresponds to measurements at different wavelengths, spanning Ultraviolet (UV) light, Visible (VIS) light, and Near Infrared (NIR). For our experiments, the NIR variable is selected.
- Handwriting (Bagnall et al., 2018): This dataset comprises 1,000 time series samples of subjects wearing a smartwatch while writing the 26 English letters. Each series has a length of 152, with three dimensions corresponding to three accelerometer values. In our experiments, we select the last dimension.
- SelfRegulationSCP (Bagnall et al., 2018): SelfRegulationSCP encompasses two datasets, SelfRegulationSCP1 and SelfRegulationSCP2, involving self-regulation of slow cortical potentials. In SelfRegulationSCP1, data from a healthy subject include cursor movement on a computer screen, with visual feedback regulating slow cortical potentials (Cz-Mastoids). SelfRegulationSCP1 consists of 561 series samples, each with a length of 896. In SelfRegulationSCP2, data from an artificially respiration ALS patient similarly involve cursor movement, with auditory and visual feedback regulating slow cortical potentials. SelfRegulationSCP2 comprises 380 series samples, each with a length of 1,152. The classification objective is to categorize based on recorded slow cortical potentials, where positive and negative potentials correspond to different classes. The analysis in both datasets focuses on the last dimension of the data in experiments.
- UWaveGestureLibrary (Bagnall et al., 2018): The UWaveGestureLibrary dataset comprises eight simple gestures generated from accelerometers, totaling 4,479 series samples. Each sample includes the x, y, z coordinates of a gesture, with each series having a length of 315. In the experiments, the analysis is focused on the z-coordinate series.
- FloodModeling (Tan et al., 2021): FloodModeling comprises three hourly datasets (FloodModeling1, FloodModeling2, and FloodModeling3). These datasets aim to predict the maximum water depth for flood modeling. The three datasets contain 673, 559, and 613 hourly rainfall events time series, respectively. Each time series in the datasets has a length of 266 time steps. These time series are utilized to predict the maximum water depth of a domain represented by a Digital Elevation Model (DEM). Both the rainfall events and DEM are synthetically generated by researchers at Monash University.
- Covid3Month (Tan et al., 2021): The Covid3Month dataset comprises 201 time series, where each time series represents the daily confirmed cases for a country. The length of each time series is 84. The objective of this dataset is to predict the COVID-19 death rate on April 1, 2020, for each country using the daily confirmed cases over the preceding three months.

B DETAILS ON TIME SERIES MODELS

The descriptions and implementations of the evaluated time series models are provided below:

Linear (Zeng et al., 2023): The Linear model represents a groundbreaking method utilizing a linear model, outperforming a substantial portion of Transformer-based models for TSF. The corresponding code is accessible at: <https://github.com/cure-lab/LTSF-Linear>.

Informer (Zhou et al., 2021): Informer is an efficient Transformer architecture specifically designed for TSF. The code for this model can be found at <https://github.com/zhouhaoyi/Informer2020>.

FEDformer (Zhou et al., 2022): FEDformer is an efficient Transformer architecture that reduces computational complexity through frequency-domain self-attention, utilizing Fourier or wavelet transforms and random selection of frequency bases. The code for this model can be accessed at: <https://github.com/MAZiqing/FEDformer>.

TimesNet (Wu et al., 2023): TimesNet transforms 1D time series into a set of 2D tensors based on multiple periods and utilizes a CNN-based model to extract features. The code for TimesNet can be found at <https://github.com/thuml/Time-Series-Library>.

PatchTST (Nie et al., 2022): PatchTST employs a segmentation approach for time series by dividing it into multiple time patches, treating each as a token. The model uses an attention module to learn the relationships between these tokens. The publicly available source code for PatchTST can be found at <https://github.com/yuqinie98/patchtst>.

C TSF RESULTS OF PATCHTST WITH PCA PREPROCESSING

PCA preprocessing is separately applied to each patch series in the patch-based time series model PatchTST. Additionally, to enhance prediction stability, PatchTST employs instance normalization technology (Kim et al., 2022). However, integrating this technology with PCA series poses challenges: the fluctuation of PCA series is considerable, and adding instance normalization further destabilizes the predictions. Consequently, after applying PCA processing, we exclude the instance normalization module from PatchTST. For comparative analysis, we also assess the performance of PatchTST without the instance normalization module on the original series.

Table 7 presents the forecasting results of PatchTST. It is observed that the original PatchTST achieves optimal performance. However, a surprising discovery is the pivotal role played by the instance normalization process in PatchTST. Omitting the instance normalization module results in a significant deterioration in PatchTST performance, exhibiting much worse results compared to training PatchTST (also without the instance normalization module) after PCA preprocessing. These findings suggest that PCA is effective for patch-based time series models, yet further exploration is required to identify alternative methods to instance normalization.

D TSF RESULTS OF RNN-BASED MODELS WITH PCA PREPROCESSING

Due to issues with gradient vanishing or exploding (Hanin, 2018), RNN-based models exhibit unstable performance in TSA with long historical series windows and have consequently been increasingly supplanted by Transformer, linear, and CNN-based models. Nonetheless, to more comprehensively evaluate the impact of PCA preprocessing, we assess its effect on RNN-based models for TSF tasks. Specifically, two typical RNN-based models, GRU (Chung et al., 2014) and LSTM (Hochreiter, 1997), are tested. Original historical series or PCA series are fed into the GRU or LSTM cells to extract features, and their hidden state h , containing the feature information, are projected and transformed to obtain the final predictions. Table 8 shows that for GRU, PCA preprocessing leads to superior performance in 18 out of 32 settings, and for LSTM, PCA preprocessing achieves better results in half of the settings. These results indicate that PCA preprocessing does not degrade the performance of RNN-based models. Additionally, since RNN models process time series sequentially, their computational cost is more sensitive to the length of the model input. Table 9 demonstrates that PCA preprocessing has a significant acceleration effect on RNN-based models, reducing training time to one-fourth and inference time to one-third of the original times. Although RNN-based models are not as commonly used as other models, PCA remains an effective tool for time series reduction in scenarios where they are appropriate.

Table 7: TSF experiments of PatchTST. The - symbol after the model signifies the removal of instance normalization processing, and the * symbol after the model indicates the application of PCA. The best result is indicated in bold font, while the second-best result is underlined.

Models		PatchTST		PatchTST-		PatchTST*	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.055	0.179	0.141	0.300	<u>0.073</u>	<u>0.214</u>
	192	0.071	0.205	0.196	0.368	<u>0.082</u>	<u>0.234</u>
	336	0.081	0.225	0.186	0.360	<u>0.087</u>	<u>0.237</u>
	720	0.087	0.232	0.372	0.527	<u>0.131</u>	<u>0.289</u>
ETTh2	96	0.129	0.282	0.232	0.381	<u>0.166</u>	<u>0.324</u>
	192	0.168	0.328	0.221	0.368	<u>0.214</u>	0.376
	336	0.185	0.351	0.537	<u>0.542</u>	<u>0.224</u>	<u>0.390</u>
	720	0.224	0.383	0.485	0.561	<u>0.298</u>	<u>0.447</u>
ETTm1	96	0.026	0.121	0.122	0.296	<u>0.031</u>	<u>0.134</u>
	192	0.039	0.150	0.127	0.299	<u>0.041</u>	<u>0.157</u>
	336	0.053	0.173	0.252	0.450	<u>0.058</u>	<u>0.184</u>
	720	0.074	0.207	0.276	0.454	<u>0.084</u>	<u>0.220</u>
ETTm2	96	0.065	0.186	0.130	0.281	<u>0.070</u>	<u>0.200</u>
	192	0.094	0.231	0.132	0.283	<u>0.098</u>	<u>0.238</u>
	336	0.120	0.265	0.165	0.322	<u>0.124</u>	<u>0.269</u>
	720	0.171	0.322	0.286	0.424	<u>0.177</u>	<u>0.328</u>

Table 8: TSF Results of RNN-based models. The * symbols after models indicate the application of PCA before inputting the series into the models. Bold font represents the superior result.

Models		GRU		GRU*		LSTM		LSTM*	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.182	0.349	0.167	0.141	0.323	0.498	0.209	0.382
	192	0.326	0.487	0.148	0.316	0.354	0.515	0.292	0.469
	336	0.233	0.408	0.144	0.310	0.387	0.553	0.261	0.441
	720	0.266	0.441	0.183	0.352	0.370	0.539	1.565	1.215
ETTh2	96	0.307	0.405	0.257	0.403	0.153	0.313	0.419	0.516
	192	0.227	0.382	0.279	0.417	0.207	0.364	0.298	0.440
	336	0.320	0.462	0.273	0.419	0.333	0.461	0.249	0.497
	720	0.392	0.502	0.285	0.435	0.421	0.534	0.349	0.378
ETTm1	96	0.070	0.198	0.164	0.335	0.091	0.249	0.130	0.292
	192	0.141	0.295	0.188	0.360	0.175	0.349	0.131	0.280
	336	0.227	0.393	0.275	0.455	0.217	0.381	0.282	0.461
	720	0.400	0.547	0.268	0.446	0.368	0.525	0.289	0.467
ETTm2	96	0.074	0.200	0.141	0.298	0.086	0.218	0.151	0.310
	192	0.119	0.267	0.187	0.352	0.119	0.270	0.211	0.368
	336	0.193	0.360	0.161	0.310	0.218	0.378	0.158	0.314
	720	0.224	0.368	0.258	0.408	0.240	0.385	0.298	0.446
Better Count		14		18		16		16	

Table 9: Average training/inference time (s) of RNN-based models on TSF tasks. The * symbols after the time series models indicate the application of PCA. Bold font represents the superior result.

	GRU	GRU*	LSTM	LSTM*
Training time	167.65	41.50	177.12	46.59
PCA time	-	0.88	-	0.88
Inference time	3.02	0.97	3.06	0.99
PCA time	-	0.01	-	0.01

E DETAILED TRAINING/INFERENCE TIME

Table 10 presents the average training and inference time (including PCA processing time) for various time series models, evaluated across different TSA tasks. With the assistance of PCA preprocessing, the training and inference of the models are accelerated to varying degrees.

Table 10: Average training/inference time (s) of different time series models across different TSA tasks. The * symbols after the time series models indicate the application of PCA before inputting the series into the models.

	Linear	Linear*	Informer	Informer*	FEDformer	FEDformer*	TimesNet	TimesNet*	PatchTST	PatchTST*
Training time	25.47	14.82	336.74	232.16	1560.67	1450.31	488.65	372.66	118.04	67.67
PCA time	-	0.88	-	0.88	-	0.88	-	0.88	-	0.88
Inference time	0.67	0.63	4.94	2.97	12.15	11.31	5.75	4.03	1.41	1.24
PCA time	-	0.01	-	0.01	-	0.01	-	0.01	-	0.01

F IMPACT OF THE NUMBER OF PRINCIPAL COMPONENTS

The number of principal components is a crucial hyperparameter in PCA. If too many principal components are selected, the reduction in dimensionality may be insufficient, failing to achieve the desired acceleration in training/inference. Conversely, too few principal components can result in the loss of important features, leading to a decline in model performance.

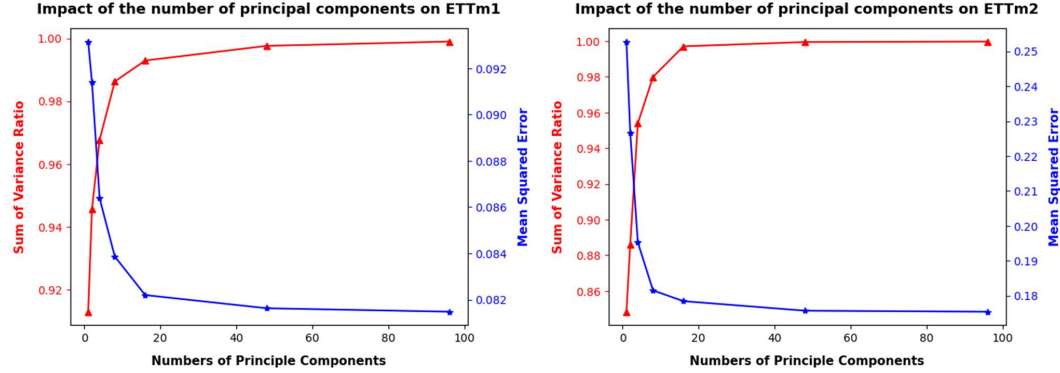


Figure 6: Impact of the number of principal components on model's performance.

Fig. 6 illustrates the impact of the number of principal components on the performance of Linear for the ETTm1 and ETTm2 datasets. The red line depicts the variation of the sum of variance ratio with the number of principal components, representing the importance of the features after PCA dimensionality reduction. As the number of principal components increases, the importance of the selected features also increases, but the rate of increase diminishes. Notably, even with only one principal component, the importance of the features is already approximately 90%, and after the number of principal components reaching to 48 (the number chosen in our experiment), further increasing the number of principal components results in minimal change in feature importance. The blue line represents the MSE of the model on the test set as a function of the number of principal components. As the number of principal components increases, the MSE decreases, but the rate of decrease also diminishes. These results suggest that selecting 48 principal components strikes a judicious balance between computational efficiency and predictive performance for TSF.

G PCA VISUALIZATIONS

Fig. 7 depicts the shapes of series after PCA preprocessing and the series obtained by inverse transforming PCA series. It is evident that PCA series include the primary information of the original series with a small subset of initial values (principal components), while the remaining values exhibit minimal fluctuations. The similarity of the original series can also be reflected in the PCA series. Furthermore, series inverse transformed from PCA series appear significantly smoother compared to the original series, effectively achieving denoising of the series.

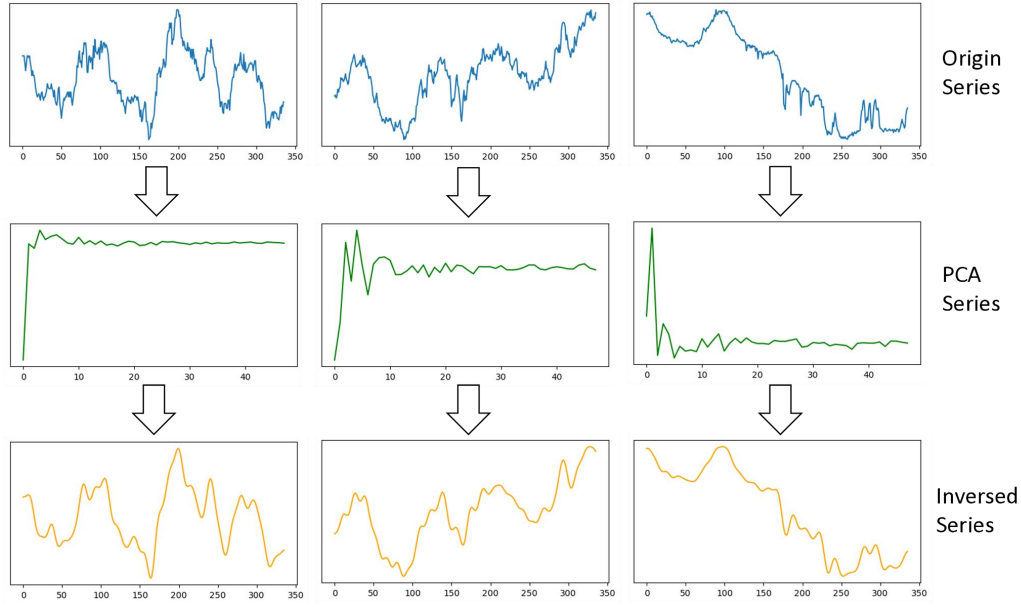


Figure 7: Visualizations of original series, PCA series and PCA-inversed series.

H PREDICTION SHOWCASES

Fig. 8 presents some prediction showcases of the Linear model with and without PCA preprocessing. It is observed that the predictions of the Linear model on the original series and the PCA series are highly consistent.

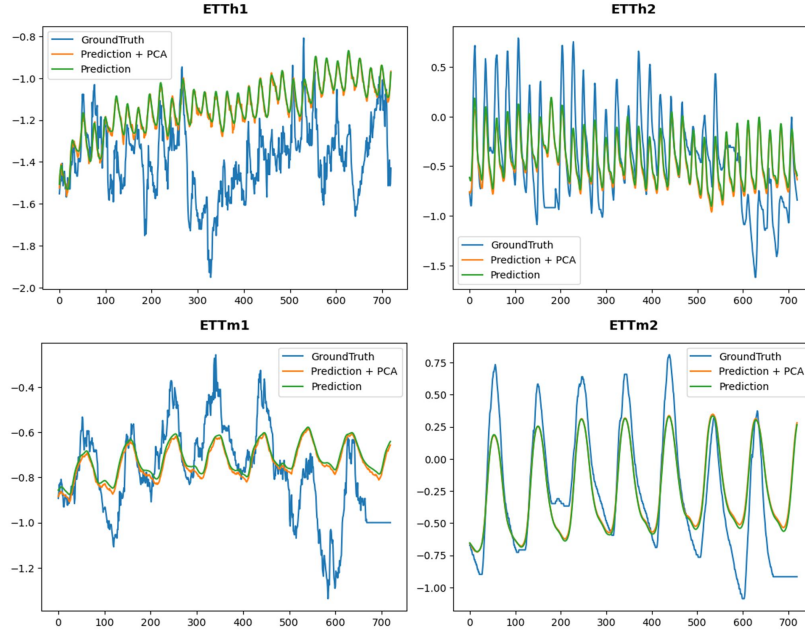


Figure 8: Prediction showcases on ETT datasets.