ZERO-SHOT TIME-SERIES RECONSTRUCTION FROM STATISTICAL SUMMARY

Anonymous authors
Paper under double-blind review

ABSTRACT

Retaining raw time-series data is often impractical under resource constraints, leaving only coarse statistical summaries available. However, when raw observations are required for downstream analysis, reconstructing fine-grained time series from summaries alone remains inherently limited. Meanwhile, deep generative and predictive models can synthesize realistic temporal patterns, but suffer from severe distribution shift when applied to unseen time series, as they are only capable of interpolating from learned data. This study proposes zero-shot time-series reconstruction from statistical summary with generative and predictive models. Statistical summary can arise from many situations where the full retention of time series is impractical, but a quick summary is feasible. We augment existing generative and predictive models with the statistical summary to enable zero-shot time-series generation. Experimental results show that the proposed method effectively reconstructs original data from statistical summary with both unseen time-range and unseen time-series generations. This allows to train a generator/predictor only once with a generic time series and use the same generator/predictor for the reconstruction of all other datasets from their statistical summaries alone.

1 INTRODUCTION

An enormous amount of time-series data is being produced daily through various sources such as financial applications, medical applications, environmental monitoring, and traffic monitoring (Chung et al., 2014; Hochreiter & Schmidhuber, 1997; Zhang et al., 2019; Fawaz et al., 2019; Fu, 2011; Reinsel, 2003; Jiang et al., 2025; Jeon et al., 2021). This rise in data generation prompted many possibilities to draw information from the time-series data and predict the future or learn important trends in the data, which typically can be realized by predictive or generative models.

However, generative and predictive models for time-series data are bound by their training data and can only interpolate based on their learned training data (Balestriero et al., 2021; Zhan et al., 2022). As a result, their accuracy declines when they attempt to advance further into unobserved time ranges or when faced with entirely new time series. In other words, they lack the ability to adapt to unseen data.

Often in certain circumstances, people tend to derive digestible information from the available data to make decisions, since it takes immense time and effort to go through the entire data to do so. One way to achieve this is through the summary of data. In particular, for time-series data, we can summarize them in statistical manner, as statistical information could be sufficient for interpreting the time-series data with a significant reduction in the volume. For instance, we may consider scenarios where the whole data is absent, but its statistical summary is available (Kumar & Reinartz, 2018; Malhotra, 2020; Chiarot & Silvestri, 2023; Sayood, 2017; InfluxData, 2026; Guo et al., 2010; Johnson et al., 2021; Guo et al., 2020). First, it is easy to gather statistical summary that represents entire population from time-series applications ranging from marketing research and product testing to experimental and observational data generated from scientific applications, since it is a costly and time-consuming process to gather all the details of the entire population. Second, limitations on computational resources such as storage space, data transfer bandwidth, and I/O rate may call for time-series data reduction, where the statistical summary can be stored in place of the entire data.

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

Third, due to difficulty in directly understanding raw time series, some data mining applications use statistical summary for better interpretability and better prediction.

While statistical summary can be adequate for many applications, every so often we may need entire time-series data for further downstream applications that benefit from the whole picture of time series. For instance, users may need visualization of the time series or prediction and classification algorithms may require the entire data. Unfortunately, in these scenarios, we do not have access to the original data anymore.

While limited on their own, the statistical summary and generators/predictors together bring a new opportunity in time-series reconstruction. We propose zero-shot time-series reconstruction from statistical summary that can predict time series that are near-identical to unseen time series. The proposed methodology paves the way to utilize existing generative and predictive models for the generation of unseen time series without costly retraining. To enable this, we consider two techniques for statistical summary: second-order summary statistics that are in widespread use (Gupta & Kapoor, 2020) and statistical similarity based data reduction (Lee et al., 2016; 2019).

We show that the proposed methodology can very closely generate unseen time series when a generator/predictor has been provided with i) the initial part of time series, to demonstrate its extrapolation capability, and ii) a generic time series, to demonstrate its zero-shot learning capability. Experimental results also demonstrate that our methodology is future-proof: better generative/predictive models in the future, along with our methodology, will only enhance the prediction performance. Our code is available at `https://anonymous.4open.science/r/Zero-shot-Time-series-Reconstruction-265B`.

2 Related work

Generative models synthesize data based on training data they are trained on so that synthetic, generated data has similar characteristics to the training data. On the other hand, predictive models learn the patterns in the training data to forecast future time series data conditional on a given historical sequence.

Generative adversarial networks (GAN) (Goodfellow et al., 2020) with generator and discriminator neural networks, which are trained with noise and original data to synthesize fake data similar to original data, has also found the time-series applications. Time-series generative adversarial networks (TimeGAN) (Yoon et al., 2019) combines autoencoders along with the generator and the discriminator to capture both point-wise temporal distributions as well as temporal changes across time. TimeGAN generates data using both open loop and closed loop to learn temporal dynamics in the original data. DoppelGANger (Lin et al., 2020) is also based on GAN and designed to eliminate mode collapse issues by implementing individual learnable normalization, which generates synthetic data by a recurrent neural network (RNN) as a batch.

Meanwhile, time-series predictive models utilize an attention layer to retain long-range dependency for forecasting successive time steps. The time-series transformer model (Wu et al., 2020) takes up data in parallel contrary to the sequential input in the case of RNNs, which has attention layers in the encoder and decoder to predict only one time step ahead. Informer (Zhou et al., 2021) has lower computational burden compared to the vanilla transformer (Vaswani et al., 2017) thanks to *ProbSparse* self-attention and the distillation of the attention product. It also employs generative decoder that predicts long-sequence time series from ground truths at one pass.

Recently, denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020) were introduced as another generative model that learns to reverse a gradual noising process using a Markov chain. And the application of diffusion models for time series data has also been studied. Among them, Diffusion-TS (Yuan & Qiao, 2024) uses an encoder-decoder transformer that learns disentangled temporal representations of time series. Diffusion-TS can be used in time-series generation as well as prediction.

Large language models (LLMs) (Brown et al., 2020) are transformer-based large models that are pre-trained on a corpus of data. Since the LLMs are pre-trained, they can be utilized for either few-shot or zero-shot forecasting tasks on their own. Prompts to the LLMs play a huge role in directing them to perform various tasks. LLMTime (Gruver et al., 2023) employs LLMs for time-series zero-

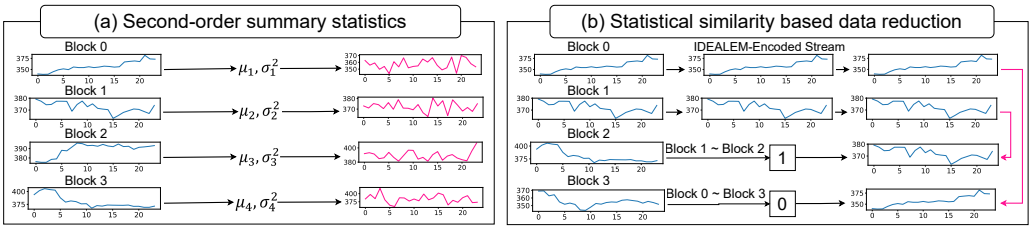


Figure 1: Two types of statistical summary considered in our research. Each block is from the original dataset. (a) Second-order summary statistics for each data block. Under the Gaussian distribution assumption, random sampling is carried out to construct statistical information. (b) Statistical similarity based data reduction, called IDEALEM. Similar blocks after the first block are replaced with indices in the encoded stream to create statistical summary. Later, they are replaced back to yield statistical information.

shot forecasting and suggests tokenization methodology for time series data. The forecasted data of LLMTime is conditioned on the time series provided in the prompt.

Like LLMTime, developing pre-trained time-series forecasting models has been a recent research trend (Ansari et al., 2024; 2025; Woo et al., 2024; Liu et al., 2025). As these models can demand high computational resources characterized by a large number of parameters, which may not always meet cost constraints in real-world scenarios, smaller models that can bring similar or better performance have been researched as well. In this regard, Tiny time mixers (TTM) (Ekambaram et al., 2024) is a small pre-trained model trained on varied multiple public time-series datasets for both zero-shot and few-shot time-series predictions. The TTM model is mainly based on TSMixer architecture (Ekambaram et al., 2023), a light-weight time-series forecasting model that consists of multi-layer perceptron (MLP) blocks interleaved with gated attention. TTM follows a multi-level architecture where it is pre-trained in channel-independent manner and then fine-tuned with channel mixing to incorporate correlations across targets and exogenous channels in the target domain.

Given generative and predictive model outcomes are thoroughly conditioned on the training data designated to the models, their extrapolation capability is limited. While LLMs and TTM have been already pre-trained on vast amounts of data and show zero-shot learning capability, their performance alone is limited as shown in Section 4. Meanwhile, statistical summary alone is limited in providing sufficient information for the reconstruction of original time series, as they are not originally meant for data compression, which typically calls for a carefully designed encoder and decoder pair (Sayood, 2017). Our proposed methodology enables the reconstruction of original time series from the statistical summary by unlocking the new potential of existing generative and predictive models.

3 Methodology

We utilize statistical summary with any data-driven generative and predictive models, regardless of their varying architectures, to implement statistically assisted models without any retraining. After training with either the initial part of time series or a generic time series, a generator/predictor is capable of synthesizing time series values. During this inference phase, statistical information serves as an additional condition (Section 3.3). The synthesized time series is then merged with the statistical information to yield a final output (Section 3.4).

3.1 Statistical summary

Several techniques have been identified to derive statistical summary from a given dataset enabling quick interpretation. Among them, we consider two plausible summary techniques.

Summary statistics is popular, which comprises variable frequency, mean, variance, deviation, coefficients of dispersion, and so on (Gupta & Kapoor, 2020). In particular, second-order statistics using mean μ and variance σ^2 is the most widely used one, which can be derived block-wise from time-series data, as depicted in Figure 1(a). Here we assume the distribution of each data block follows a Gaussian distribution, which is fully defined by μ and σ^2. Random sampling is carried out for each distribution to construct statistical information, though its values could be in random

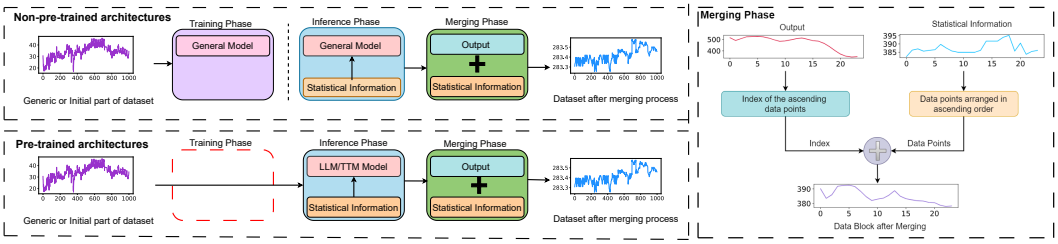


Figure 2: Our proposed methodology that enables zero-shot time-series reconstruction. Merging refines the output from the inference phase by leveraging the actual distribution of time series in question.

order. This process is repeated as needed by varying generators and predictors. Note that we do not perform random sampling for LLMTime, as the summary statistics can be directly provided in the form of a prompt.

Another technique to derive statistical summary is using statistical similarity based data reduction proposed in Lee et al. (2016; 2019). Specifically, they used Kolmogorov–Smirnov test (Knuth, 2014) to find similar and thus exchangeable data blocks in time series. Since exchangeable blocks are similar to each other, one can only keep the first instance of such blocks and replace other blocks with the same index. This technique is based on the assumption that any time series can be explained by a few random number generators and we can view the encoded stream of data blocks and indices as statistical summary. Unlike the second-order statistics that are parametric by their nature, this technique admits to non-parametric distributions.

The encoded stream from IDEALEM encoder (Lee et al., 2016; 2019) typically consists of a small number of original data blocks and lots of indices each of which refers to one of the original data blocks, as depicted in Figure 1(b). To construct statistical information from the encoded stream, we replace each index with a corresponding original data block. We note that the original IDEALEM decoder (Lee et al., 2016; 2019) employs a random permutation when it reconstructs an exchangeable data block with an index. However, in our work, it is unnecessary to perform such an additional step, as the block is merely treated as a distribution, where the order of values is ignored.

3.2 Training Phase

Conventional approaches in training generative/predictive models need a set of relevant training data, as the output quality of inference phase will rely on the training data. Although pre-trained models such as LLMTime and TTM, whose training data consist of large corpus of text data to gain a general understanding of language or large collection of diverse time-series datasets for time-series specific understanding, do not require another training, they are also bound to the original training data during the inference phase. The details of those approaches are described in Appendix A.

On the other hand, as shown in Figure 2, during the training phase of our methodology, a model is trained with any dataset, which could be the initial part of a time series in question where we want to advance deeper into unseen time range or any generic time series that is unrelated to the time series in question. In other words, the proposed methodology allows generative/predictive models not to be conditioned by the dataset injected during the training phase. As for LLMTime and TTM, the initial part of or generic time series is fed during the inference phase, as shown in Figure 2.

3.3 Inference Phase

In the inference phase, conventional approaches feed noise and/or the original data used in training to a model to synthesize time series (i.e., generator) or predict next steps (i.e., predictor). On the other hand, as shown in Figure 2, during the inference phase of our methodology, the model will receive the statistical information of time series in question to help generate the target time series, which is shown in Figure 3. The statistical information divided into blocks can be constructed from the statistical summary. In the case of TimeGAN and DoppelGANger, statistical information blocks, instead of the noise, are introduced into their generative phases to create synthesized blocks. Since Informer and Diffusion-TS are predictive models, statistical information blocks are fed into the lookback window for their predictive phases.

As for LLMTime, prompt engineering is employed to specify the tasks that help LLMTime to understand what is expected from the model. We provide a time-series sequence as a history, i.e, the initial part in the extrapolation or generic time series in the zero-shot learning, for synthesizing a future time series and also describe the statistical summary to be used in prediction. (Actual prompts can be found in Appendix B.) TTM model is first fine-tuned on either the initial part or generic time series. Statistical information blocks are subsequently fed into the fine-tuned predictive phase to create forecasting blocks.

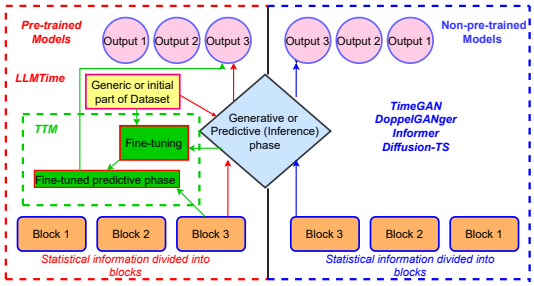


Figure 3: Inference phase for generative/predictive models.

3.4 MERGING

Although the statistical information is used during the inference phase, the output of generative/predictive models needs further refinement to reconstruct time series that are close to the target time series. To this end, the last step of our methodology is merging. Here, we consider the best of both worlds: realistic pattern of time series from the generator/predictor and actual distribution of time series. For instance, in Figure 1(a), random sampling from second-order summary statistics can only yield a sequence in random order, which tends to fluctuate more than the original sequence. In addition, duplicating the first instance for exchangeable indices in Figure 1(b) creates non-realistic patterns. These discrepancies can be effectively handled by utilizing patterns generated from the generator/predictor.

In the right side of Figure 2, we show the merging process in more detail. Realistic patterns from generative and predictive models can be captured by the change of index when the output sequence is sorted in ascending order.[1] We then enforce this index change to a statistical-information block sorted in ascending order to create high-fidelity time series. Here sorting is simply necessary to remember the original order of generator/predictor output. And the permutation map remembers such an order and its inverse mapping can eventually restore the original pattern. Algorithm 1 formally presents the merging process.

Algorithm 1: Merging process

Input: generator/predictor output $\mathbf{x} \in \mathbb{R}^N$, statistical information $\mathbf{y} \in \mathbb{R}^N$
Output: merged output $\mathbf{z} \in \mathbb{R}^N$

$\mathbf{x}^* \leftarrow \mathbf{x}$
Rearrange $\mathbf{x}^*$ in the nondecreasing order such that $x_1^* \leq x_2^* \leq \cdots \leq x_N^*$
Find the permutation map π such that $x_i^* = x_{\pi(i)}$ for all $i \in [N]$
$\mathbf{y}^* \leftarrow \mathbf{y}$
Rearrange $\mathbf{y}^*$ in the nondecreasing order such that $y_1^* \leq y_2^* \leq \cdots \leq y_N^*$
$\mathbf{z} \leftarrow [y^*_{\pi^{-1}(1)}, y^*_{\pi^{-1}(2)}, \ldots, y^*_{\pi^{-1}(N)}]^\intercal$
return $\mathbf{z}$

The entire merging process mainly involves sorting operation whose computational complexity is $\mathcal{O}(N \log N)$. Nevertheless, since we adopt block-based processing, which makes $\log N$ merely a constant, the entire process yields a linear complexity. The merged data would be less noisy and more similar to the original data, which will be demonstrated in Section 4. Detailed analysis of merging with an example and theoretical treatment can be found in Appendix C.

4 EXPERIMENTS

To show that the proposed methodology can very closely reconstruct unseen time series from statistical summary, we consider two scenarios in experimental results: extrapolation and zero-shot learning.

[1] Alternatively, it can be in descending order as far as it is consistent between the output and the reconstructed block.

4.1 EXPERIMENTAL SETTING

We used seven datasets in our experiments: L1MAG, L2MAG, L3MAG (Stewart et al., 2016); Gold prices (Möbius, 2020); LIC share prices (Sahoo, 2022); Bike rental (Fanaee-T & Gama, 2014); Electricity Transformer Temperature (ETTh1) (Zhou et al., 2021).

The experimental results are compared qualitatively using plots and quantitatively using well-known error measures: mean squared error (MSE), mean absolute error (MAE), and symmetric mean absolute percentage error (SMAPE). We provide the average results of five runs along with standard deviations. Further details of experimental setting can be found in Appendix D.

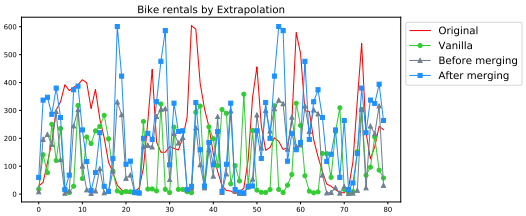
To the best of our knowledge, there is no other methodology similar to ours where generative and predictive models are employed to reconstruct time series from statistical summary. As such, our baselines are the original (vanilla) versions of generative/predictive models introduced in this paper: TimeGAN, DoppelGANger, Informer, Diffusion-TS, LLMTime, and TTM. These vanilla versions do not incorporate statistical summary during the inference phase and the merging is also excluded. Another baseline is reconstruction from statistical summary only. In the tables, boldfaced numbers represent the best average results for each error category. Due to space constraints, TimeGAN results and standard deviations for the best results are reported in Appendix E.

4.2 EXTRAPOLATION

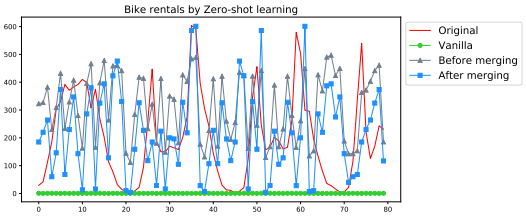
In the extrapolation scenario, experiments are conducted on seven datasets. We divide each dataset into two equal parts: the initial part and the later part. Since the experiments are performed to reconstruct the later part of the dataset, models except LLMTime and TTM are trained on the initial part of the dataset and the testing or inference phase receives the statistical information of the later part derived through either the summary statistics or the statistical similarity based data reduction. In the case of LLMTime and TTM, the training phase is absent. Thus the initial part of the dataset along with the statistical information is introduced into its inference phase. After the inference phase, the merging process follows. As for the baselines, only the initial part of the dataset is introduced into both the training and inference phases, or into only the inference phase for LLMTime and TTM. We compare their reconstruction results with the later part of the dataset to measure the errors.

The following series of tables shows the expected results where the baselines have very high values of errors compared to the proposed models. Table 1 shows the baseline results from five vanilla models. If we compare their results with those shown in Tables 2 and 3, where we show the results with the summary statistics and the statistical similarity based data reduction respectively, we observe that there are dramatic differences across all datasets except Bike Rental.

This is mainly due to the high dynamic range (between 0 and 600) of the dataset and their recurring patterns of peaks and valleys unlike other datasets as shown in Figure 4. This results in baselines having difficulties in accurately generating/predicting original time series patterns, thereby compounding errors during the inference and merging. For example, in Figure 4a, the vanilla result from Informer is nowhere near the recurring patterns of peaks and valleys. In addition, the bike rental dataset contains 14 features that explain environmental and seasonal conditions to enable forecasting techniques to take into account of all the features to predict the number of rentals per day. Therefore in a univariate scenario like ours, it is difficult for any generators/predictors to exactly match the locations of peaks and valleys, and even small deviation results in very high errors.



(a) Bike rental data and its extrapolation results.



(b) Bike rental data and its zero-shot results.

Figure 4: Bike rental dataset and its reconstruction effort by Informer.

Table 1: Extrapolation errors from vanilla generative/predictive models

Model	DoppelGANger			Informer			Diffusion-TS			LLMTime			TTM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0454	0.1748	0.0308	0.0272	0.1391	0.0245	0.0272	0.1391	0.0245	**0.0133**	**0.0933**	**0.0164**	0.0250	0.1343	0.0236
L2MAG	0.0287	0.1402	0.0247	0.0222	0.1261	0.0223	0.0210	0.1227	0.0217	0.0200	**0.1177**	**0.0208**	**0.0196**	0.1189	0.0210
L3MAG	0.0318	0.1470	0.0259	0.0245	0.1306	0.0230	0.0245	0.1303	0.0229	**0.0146**	**0.1002**	**0.0176**	0.0232	0.1272	0.0224
Gold Price	66516	207.12	21.496	**19064**	**108.52**	**13.189**	20849	113.76	13.722	27682	135.87	21.559	20378	112.19	13.548
LIC	8319.9	74.177	4.9947	6031.7	61.876	4.1997	6087.0	58.937	3.9476	37326	192.66	12.240	**5929.6**	**58.225**	**3.9040**
Bike Rental	51375	178.73	**51.055**	40939	161.03	54.446	48388	175.11	60.150	**38071**	**146.75**	57.975	41525	162.59	57.900
ETTh1	371.82	15.765	41.895	**290.20**	**14.168**	**40.560**	293.42	14.479	41.219	341.30	17.664	47.420	293.76	14.446	41.033

Table 2: Extrapolation errors from generative/predictive models with summary statistics

Model	DoppelGANger			Informer			Diffusion-TS			LLMTime			TTM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0009	0.0215	0.0038	**0.0007**	**0.0202**	**0.0036**	0.0009	0.0223	0.0039	0.0009	0.0216	0.0038	0.0008	0.0206	0.0036
L2MAG	0.0009	0.0222	0.0039	0.0009	0.0223	0.0039	0.0010	0.0230	0.0041	0.0010	0.0226	0.0040	**0.0008**	**0.0201**	**0.0036**
L3MAG	0.0010	0.0233	0.0041	0.0009	0.0223	0.0039	0.0011	0.0245	0.0043	0.0010	0.0230	0.0040	**0.0008**	**0.0203**	**0.0036**
Gold Price	111.33	**6.2570**	0.8035	**108.88**	6.4842	0.8292	110.02	6.5457	0.8270	119.84	6.6236	0.8535	129.81	6.3764	**0.8009**
LIC	3.3893	1.1671	0.0845	**3.2795**	1.1657	0.0844	3.4836	1.2195	0.0883	3.6442	**1.1058**	**0.0798**	3.3640	1.1931	0.0864
Bike Rental	39127	156.57	50.090	36084	149.52	48.917	38478	156.54	50.256	**35389**	149.34	49.415	36029	**148.62**	**48.587**
ETTh1	**5.3836**	**1.7212**	**9.5931**	6.3459	1.8622	10.055	6.7432	1.9652	10.661	6.1695	1.8532	10.260	6.3483	1.8681	9.9853

Table 3: Extrapolation errors from generative/predictive models with statistical similarity based data reduction

Model	DoppelGANger			Informer			Diffusion-TS			LLMTime			TTM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0009	0.0227	0.0040	0.0008	0.0222	0.0039	0.0010	0.0235	0.0041	0.0008	0.0216	0.0038	**0.0008**	**0.0215**	**0.0038**
L2MAG	0.0011	0.0259	0.0046	0.0012	0.0267	0.0047	0.0012	0.0271	0.0048	**0.0011**	**0.0245**	**0.0043**	0.0011	0.0256	0.0045
L3MAG	0.0012	0.0271	0.0048	0.0013	0.0274	0.0048	0.0013	0.0279	0.0049	**0.0012**	**0.0259**	**0.0046**	0.0012	0.0262	0.0046
Gold Price	204.70	8.3414	1.0320	207.84	8.3727	1.0280	208.35	8.2531	1.0127	**158.16**	**7.2229**	**0.9280**	196.41	8.0544	0.9971
LIC	7.3237	1.5311	0.1111	7.4027	1.5416	0.1118	7.5355	1.5939	0.1155	**5.4913**	**1.1703**	**0.0847**	7.7233	1.5748	0.1142
Bike Rental	39824	151.89	45.643	37780	146.32	44.746	36765	141.59	**42.863**	36602	143.85	44.790	**33393**	**137.02**	43.229
ETTh1	8.0449	2.0911	11.559	8.0856	2.0636	11.125	**7.6721**	**1.9967**	11.048	8.6380	2.0766	11.184	7.7020	2.0010	**10.830**

We also observe that the extrapolation with summary statistics shows better results than that with statistical similarity based data reduction, except for the case of Bike Rental. This shows that statistical information in the form of simple second-order summary statistics and under the assumption of Gaussian distribution is sufficient to produce the original time series for our proposed method.

Lastly, we observe that best error results come from varying generative/predictive models depending on the dataset, the statistical information, and error metrics. There is not a single dominant model that outperforms others in all the cases.

4.3 ZERO-SHOT LEARNING

In this scenario, ETTh1 is used as a generic dataset since it exhibits a reasonable dynamic range and thus is similar to many real-world datasets. (Appendix D.1 has its details.) The ETTh1 dataset is used in the training phase (or in the inference phase in LLMTime and TTM). The resulting models generate/predict datasets that are very different from each other as well as from ETTh1. For instance, datasets such as L1MAG, L2MAG, and L3MAG have values greater than 250 but data points vary in the range of 10^{-1}. On the other hand, for datasets such as LIC and gold prices, data points vary in the range greater than 10^2. And the bike rental dataset exhibits a higher dynamic range.

Through a series of experiments, we demonstrate that ETTh1, which is commonly available data, can be used even for the generation of exclusive datasets when their statistical information is readily available. We thus exclude ETTh1 from the experimental results and conduct experiments on six other datasets. As in the case of the extrapolation, inference outputs from the models is post-processed by applying the block-wise merging. As for the baselines, the ETTh1 dataset is induced into the inference phase of vanilla models. We compare the reconstruction results with the dataset of interest.

Tables 4, 5, and 6 show the results where the baselines have very high values of errors compared to the proposed models. Specifically, Table 4 shows the baseline results from five vanilla models. Except for the case of Bike Rental, we see dramatic differences across all datasets in errors between Table 4 and Tables 5 and 6. Similarly to the extrapolation, with the bike rental dataset, it is difficult

Table 4: Zero-shot learning errors from vanilla generative/predictive models

Model	DoppelGANger			Informer			Diffusion-TS			LLMTime			TTM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	68615	261.57	85.748	68605	261.74	85.655	68225	261.04	85.204	**65634**	**256.15**	**82.181**	68205	261.01	85.178
L2MAG	67594	259.67	84.883	68335	261.23	85.682	67953	260.53	85.229	**65346**	**255.59**	**82.174**	67602	259.87	84.810
L3MAG	67883	260.21	84.782	68602	261.73	85.553	68267	261.12	85.155	**65798**	**256.47**	**82.276**	68225	261.05	85.104
Gold Price	135176	352.62	87.628	136447	353.61	87.764	135931	352.95	87.433	**126175**	**342.02**	**85.505**	135894	352.89	87.404
LIC	445815	667.40	93.423	448930	669.92	94.106	447914	669.15	93.890	**439396**	**662.67**	**92.124**	447826	669.09	93.873
Bike Rental	48576	166.94	69.120	48418	166.54	68.269	48283	166.09	67.761	**45181**	**159.20**	**64.179**	48186	165.99	67.715

Table 5: Zero-shot learning errors from generative/predictive models with summary statistics

Model	DoppelGANger			Informer			Diffusion-TS			LLMTime			TTM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0010	0.0231	0.0041	0.0009	0.0219	0.0038	0.0009	0.0223	0.0039	0.0009	0.0213	0.0038	**0.0008**	**0.0208**	**0.0037**
L2MAG	0.0012	0.0258	0.0046	0.0010	0.0231	0.0041	0.0010	0.0231	0.0041	0.0009	**0.0215**	**0.0038**	**0.0009**	0.0220	0.0039
L3MAG	0.0012	0.0254	0.0045	0.0009	0.0227	0.0040	0.0011	0.0245	0.0043	0.0011	0.0235	0.0041	**0.0009**	**0.0220**	**0.0039**
Gold Price	114.99	6.5114	0.8313	**108.88**	**6.4842**	0.8292	110.21	6.5425	0.8265	112.68	6.5387	0.8485	120.01	6.5312	**0.8175**
LIC	3.3171	1.1929	0.0864	3.4077	1.1869	0.0859	3.4313	1.2187	0.0883	3.6580	1.1056	0.0798	**2.7500**	**1.0986**	**0.0795**
Bike Rental	39765	154.75	48.790	**35727**	**148.01**	**48.533**	38539	156.34	50.155	36011	150.54	49.479	38955	155.85	49.816

Table 6: Zero-shot learning errors from generative/predictive models with statistical similarity based data reduction

Model	DoppelGANger			Informer			Diffusion-TS			LLMTime			TTM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0010	0.0250	0.0044	0.0009	0.0228	0.0040	0.0010	0.0234	0.0041	**0.0008**	**0.0219**	**0.0039**	0.0010	0.0242	0.0043
L2MAG	0.0013	0.0288	0.0051	0.0011	0.0261	0.0046	0.0012	0.0271	0.0048	**0.0011**	**0.0244**	**0.0043**	0.0012	0.0267	0.0047
L3MAG	0.0015	0.0293	0.0052	**0.0012**	0.0264	0.0046	0.0013	0.0279	0.0049	0.0012	**0.0261**	**0.0046**	0.0013	0.0278	0.0049
Gold Price	201.88	8.2728	1.0162	212.33	8.3937	1.0266	207.47	8.2407	1.0110	**158.57**	**7.4355**	**0.9480**	198.49	8.1967	1.0157
LIC	7.0384	1.5206	0.1102	7.5041	1.5322	0.1111	7.5664	1.5971	0.1157	**5.2271**	**1.1803**	**0.0854**	7.4341	1.5070	0.1093
Bike Rental	52585	179.37	50.966	35475	144.91	46.248	36822	141.73	**42.866**	36008	141.85	44.512	**33465**	**140.79**	45.087

for baselines to reduce errors, since generating/predicting cannot exactly match the locations of peaks and valleys as shown in Figure 4b.

We also observe that the zero-shot errors with summary statistics show better results than those with statistical similarity based data reduction, except for the case of Bike Rental, similarly to the extrapolation. And best error results come from varying generative/predictive models depending on the dataset, the statistical information, and error metrics.

4.4 STATISTICAL SUMMARY ONLY

Table 7: Statistical summary only errors

Type	Statistics			IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0009	0.0225	0.0040	0.0009	0.0227	0.0040
L2MAG	0.0010	0.0236	0.0042	0.0012	0.0264	0.0047
L3MAG	0.0010	0.0240	0.0042	0.0013	0.0279	0.0049
Gold Price	112.85	6.6332	0.8461	205.63	8.3249	1.0299
LIC	3.1152	1.1359	0.0823	6.0349	1.3076	0.0948
Bike Rental	41562	161.35	50.017	30457	116.75	35.275
ETTh1	7.2914	2.0431	11.445	8.5982	1.9515	10.934

In this baseline scenario, statistical summary is used solely to reconstruct seven datasets. Table 7 shows errors resulting from the statistical information constructed from the summary, as explained in Section 3.1, where we present two different statistical summary types: summary statistics (Statistics) and statistical similarity based data reduction (IDEALEM). We observe that summary statistics show better results than those with statistical similarity based data reduction, except for the case of Bike Rental, similarly to cases involving generators/predictors.

5 DISCUSSION

As observed in Tables 1 and 4, it is virtually impossible for vanilla models alone to reconstruct unseen time range or unseen time series. In addition, there are orders of differences between errors in Tables 1 and 4 due to more difficulty in zero-shot reconstruction. It is interesting to note that for these vanilla scenarios, LLMTime performs the best, which we believe is attributable to its pre-trained nature with large model size that has generalization capacity (cf. small model size for TTM).

Table 7 shows reconstruction results when statistical summary is used alone to reconstruct the original time series. Since the statistical summary contains key information (i.e., actual distribution) to

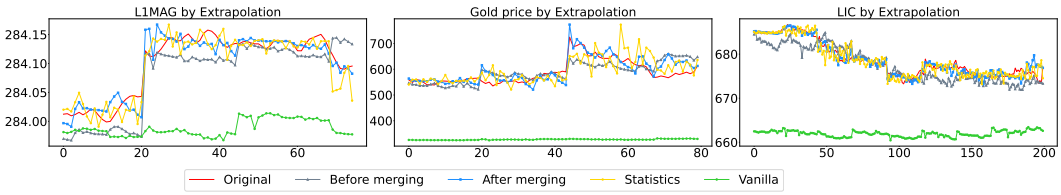


Figure 5: Example plots for reconstruction by Informer extrapolation and summary statistics.

reconstruct time series, these baseline results are superior to those of vanilla generative/predictive models. However, the statistical summary only reconstruction is agnostic to the realistic pattern of time series: random sampling for the second-order summary statistics and duplicating exchangeable data blocks for the statistical similarity based data reduction.

For both the extrapolation and the zero-shot learning, the summary statistics showed better results than the statistical similarity based data reduction did. This might look counterintuitive at first glance. This result comes from two reasons. First, while the Gaussian distribution assumption can be considered primitive, it models real-world datasets well in practice and works especially well with existing generative/predictive models when used as statistical information in the inference phase and merging. Second, it is better to have second-order statistics of every block rather than representing a statistically similar block, in terms of the Kolmogorov–Smirnov test, with the first such block (i.e., non-parametric distribution) in the dataset, when using the proposed methodology.

On the contrary, for the bike rental dataset shown in Figure 4, the statistical similarity based data reduction (IDEALEM) works better than the summary statistics. This is also the case where baseline models struggle to generate/predict actual patterns. As a result, the statistical summary only reconstruction (Table 7) shows better results than both the extrapolation and the zero-shot learning results (Tables 3 and 6). In Figure 4, the patterns do not follow the Gaussian distribution that centers around the mean, but more closely follow a uniform distribution (with some overlap and folding depending on block boundary). As such, this distribution is more correctly captured by the IDEALEM-style statistical summary, which is non-parametric and can capture any distribution.

We also note that both the extrapolation and the zero-shot learning show similar performance. This looks counterintuitive as well, since the errors for vanilla zero-shot learning are far greater than those for vanilla extrapolation. In fact, we can sometimes observe better results in the case of zero-shot learning (e.g., LIC dataset with summary statistics). This shows that the proposed methodology works regardless of any dataset generative/predictive models were originally trained with (or provided with in case of LLMTime and TTM). And this also serves as a testament to the proposed methodology's zero-shot learning capability.

Figure 5 shows example results of reconstruction for the extrapolation. It is clear that vanilla versions introduced only with the initial part of data struggle to accurately generate original time series. However, the results can be significantly improved by the inference phase with the statistical information (Before merging) and further refined by merging (After merging). When the statistical summary is solely employed for reconstruction, we note that distribution shift is still captured, but the result is noisy as it is agnostic to the realistic pattern of time series. Further ablation study can be found in Appendix F.

6 Conclusion

We considered scenarios where it is not desirable to retain raw time-series data, leaving only statistical summaries available. We proposed a methodology to reconstruct time series from statistical summary with existing generative and predictive models, where these models alone are limited in handling unseen time range and unseen time series.

With available statistical information of the dataset, our proposed methodology helps the models to expand their horizons by producing data of entirely unseen time series, enabling zero-shot reconstruction of time series.

AI USE STATEMENT

In this work, we used a generative AI tool for experiments with LLMTime due to their inherent design. We have not used generative AI tools for all other required disclosure tasks. Additionally, we used generative AI tools for identifying some of relevant literature. We have reviewed all AI-assisted work to verify their correctness. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 3, Appendices A, B, and D have all the details of reproducing the proposed work. Additionally, the link to our code can be found at the end of Section 1.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Randall Balestriero, Jerome Pesenti, and Yann LeCun. Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pp. 1877–1901, 2020.

Giacomo Chiarot and Claudio Silvestri. Time series compression survey. *ACM Computing Surveys*, 55(10):1–32, 2023.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. TSMixer: Lightweight MLP-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 459–469, 2023.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, pp. 74147–74181, 2024.

Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127, 2014.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*, pp. 19622–19635, 2023.

Chonghui Guo, Hailin Li, and Donghua Pan. An improved piecewise aggregate approximation based on statistical features for time series mining. In *Proceedings of the 4th International Conference on Knowledge Science, Engineering and Management*, pp. 234–244, 2010.

Chonghui Guo, Menglin Lu, and Jingfeng Chen. An evaluation of time series summary statistics as features for clinical prediction tasks. *BMC Medical Informatics and Decision Making*, 20(1):48, 2020.

S. C. Gupta and V. K. Kapoor. *Fundamentals of Mathematical Statistics*. Sultan Chand & Sons, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pp. 6840–6851, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

InfluxData. Downsample and retain data — InfluxDB OSS v1 Documentation. `https://docs.influxdata.com/influxdb/v1/guides/downsample_and_retain/`, 2026. Accessed: 2026-09-25.

Jinsung Jeon, Soyoung Kang, Minju Jo, Seunghyeon Cho, Noseong Park, Seonghoon Kim, and Chiyoung Song. Lightmove: A lightweight next-POI recommendation for taxicab rooftop advertising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3857–3866, 2021.

Yushan Jiang, Kanghui Ning, Zijie Pan, Xuyang Shen, Jingchao Ni, Wenchao Yu, Anderson Schneider, Haifeng Chen, Yuriy Nevmyvaka, and Dongjin Song. Multi-modal time series analysis: A tutorial and survey. *arXiv preprint arXiv:2503.13709*, 2025.

Nari Johnson, Sonali Parbhoo, Andrew S Ross, and Finale Doshi-Velez. Learning predictive and interpretable timeseries summaries from ICU data. In *AMIA Annual Symposium Proceedings*, pp. 581–590, 2021.

Donald E Knuth. *The Art of Computer Programming: Seminumerical Algorithms, Volume 2*. Addison-Wesley Professional, 2014.

Vineet Kumar and Werner Reinartz. *Customer Relationship Management*. Springer, 2018.

Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. Novel data reduction based on statistical similarity. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, pp. 1–12, 2016.

Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. IDEALEM: Statistical similarity based data reduction. *arXiv preprint arXiv:1911.06980*, 2019.

Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using GANs for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the 20th ACM Internet Measurement Conference*, pp. 464–483, 2020.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering time series foundation models with sparse mixture of experts. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Naresh K. Malhotra. *Marketing Research: An Applied Orientation*. pearson, 2020.

Möbius. Learn time series forecasting from gold price. https://www.kaggle.com/datasets/arashnic/learn-time-series-forecasting-from-gold-price, 2020. Accessed: 2026-09-25.

Gregory C Reinsel. *Elements of Multivariate Time Series Analysis*. Springer Science & Business Media, 2003.

Debashis Sahoo. LIC stock price data. https://www.kaggle.com/datasets/debashis74017/lic-stock-price-data, 2022. Accessed: 2026-09-25.

Khalid Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 2017.

Emma Stewart, Anna Liao, and Ciaran Roberts. Open μPMU: A Real World Reference Distribution Micro-Phasor Measurement Unit Data Set for Research and Application Development. LBNL Technical Report 1006408, Lawrence Berkeley National Labarotory, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, 2017.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable diffusion for general time series generation. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Jingtao Zhan, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Evaluating interpolation and extrapolation performance of neural retrieval models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2486–2496, 2022.

Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 1409–1416, 2019.

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

A Training phase of generative/predictive models

Conventional approaches in training generative/predictive models are shown in Figures 6 and 7, where we need a set of relevant training data, as the output quality of inference phase will rely on the training data. We note that Diffusion-TS is used in the prediction mode, i.e., conditional generation (Yuan & Qiao, 2024).

Although pre-trained models such as LLMTime and TTM shown in Figure 8, whose training data consists of large corpus of text data to gain a general understanding of language or multiple public time-series datasets of approximately 1 billion samples (Ekambaram et al., 2024) to gain a general understanding of time series, do not require another training, they are also bound to the original training data during the inference phase.

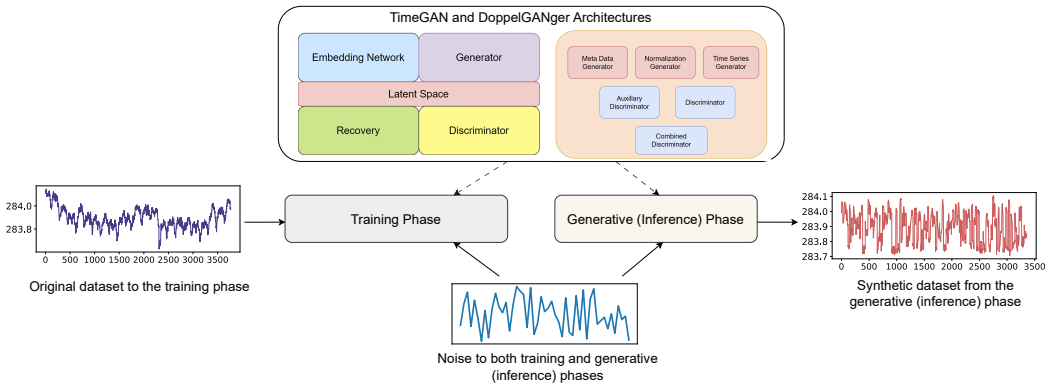


Figure 6: Generative models (e.g., TimeGAN and DoppelGANger) synthesize time series based on the original training data.

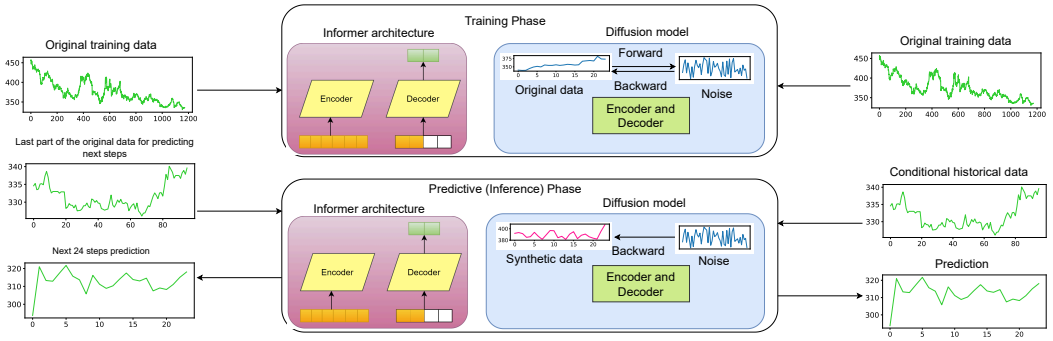


Figure 7: Predictive models (e.g., Informer and Diffusion-TS) predict time series based on the original training data.

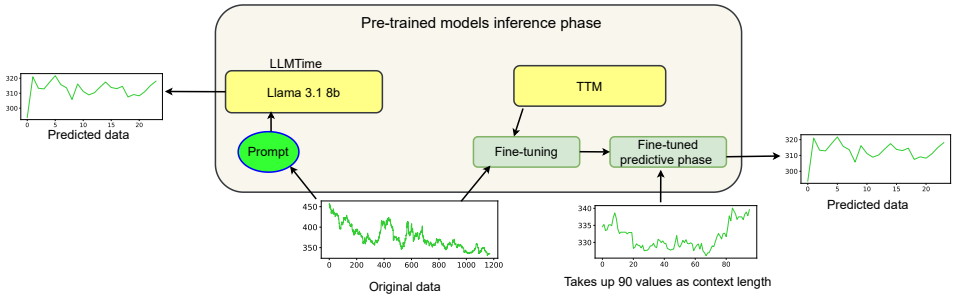


Figure 8: Pre-trained models (e.g., LLMTime and TTM) predict time series based on the original training data.

B PROMPTS FOR LLMTIME

For LLMTime, a time-series sequence has to be provided in the prompt of LLM so that it acts as the history for forecasting the future time series. In this regard, prompt engineering helps to specify the tasks that help LLMTime to understand what is expected from the model, where we explain the prominence of statistical summary for the prediction.

Since two types of statistical information are available, the prompt changes for these types. For the summary statistics, we include instructions in the prompt to predict a data block with reference to the distribution with the mean μ and the variance σ^2. For the statistical similarity based data reduction, the prompt describes the probability distribution of the next block to be the reconstruction from the IDEALEM-encoded stream.

Listing 1: Prompt for vanilla model

```
llama_sys_message= "The user provides you with input sequence, you
    have to predict its future sequence. The predicted sequence is of
    only 24 words separated by only commas with no spaces and no other
     characters in string. Please do not produce any additional text."
few_shot_prompt_input1= "The input sequence is: "
input1= "Please kindly produce the predicted sequence which is the
    continuation of the input sequence: "

chat_completion = chat(
    model="llama3.1:8b",
    options={
        "temperature": 0.3,
        #top_p= 1,
        "seed":int(time.time())
    },
    messages=[
        {
            "role": "system",
            "content":  llama_sys_message
        },
        {
            "role": "user",
            "content": few_shot_prompt_input1+
                few_shot_prompt_generic_data
        },
        {
            "role": "assistant",
            "content": few_shot_prompt_TTM_output
        },
        {
            "role": "user",
            "content": input1+generic_data
        },
    ]
)
```

For the vanilla model, we use the prompt shown in Listing 1. For the summary statistics, we use the prompt shown in Listing 2. And for the statistical similarity based data reduction, we use the prompt shown in Listing 3. The lookback window size is 144.

In the prompts, the system role refers to setting up the AI persona: asking it to work as a predictor. User role refers to the current query specific request. The assistant role in the prompt is an example output provided as a reference to the LLM, expecting the future output of the LLM follows the methodology or the concept used in few shot prompting. Few shot prompting is adopted here to provide the LLM with a few examples so that it understands the type of output the user expects. Specifically, statistical information from L1MAG (only the first block) and reference output from TTM (only the first block) are used for that purpose. We chose TTM output, since LLMTime and TTM belong to the same category of pre-trained models.

C Detailed analysis of merging

We first start with a concrete example for the merging process described in Algorithm 1. Let $\mathbf{x} = [1, 3, 5, 7, 6, 4, 2, 0]^\intercal$, with a distinctive hat pattern. Sorting $\mathbf{x}$ in the nondecreasing order will yield $\mathbf{x}^* = [0, 1, 2, 3, 4, 5, 6, 7]^\intercal$, where we can learn the permutation π, represented by $\{(1, 8), (2, 1), (3, 7), (4, 2), (5, 6), (6, 3), (7, 5), (8, 4)\}$.

Now let $\mathbf{y} = [22, 18, 30, 26, 14, 10, 38, 34]^\intercal$. Note that the order of values in the statistical information $\mathbf{y}$ is unimportant and we could have also represented it with a set

Listing 2: Prompt for summary statistics

```
llama_sys_message= "The user provides an input time series sequence
    along with a target mean and standard deviation. You must predict
    only 24 values that conform to the given mean and standard
    deviation. Output ONLY 24 comma-separated numerical values with no
     spaces and no additional text."
few_shot_prompt_input1= "Please kindly produce the normally
    distributed sequence, where its mean is : {mean}".format(mean=
    few_shot_prompt_stat_info_mean)
few_shot_prompt_input2= "Please kindly produce the normally
    distributed sequence, where its standard deviation is : {std}".
    format(std=few_shot_prompt_stat_info_std)
few_shot_prompt_input3= "The input sequence is: "
input1= "As shown in the example produce the normally distributed
    sequence based on mean and standard deviation given by user.
    Please kindly produce the normally distributed sequence, where its
     mean is : {mean}".format(mean=stat_info_mean)
input2= "Please kindly produce the normally distributed sequence,
    where its standard deviation is : {std}".format(std=stat_info_std)
input3= "The input sequence is: "

chat_completion = chat(
    model="llama3.1:8b",
    options={
        "temperature": 1,
        #top_p= 1,
        "seed":int(time.time()),
        "max_tokens":190
    },
    messages=[
        {
            "role": "system",
            "content":  llama_sys_message
        },
        {
            "role": "user",
            "content": few_shot_prompt_input1+few_shot_prompt_input2+
                few_shot_prompt_input3+few_shot_prompt_generic_data
        },
        {
            "role": "assistant",
            "content": few_shot_prompt_TTM_output
        },
        {
            "role": "user",
            "content": input1+input2+input3+generic_data
        },
    ]
)
```

$\{10, 14, 18, 22, 26, 30, 34, 38\}$.[2] In any case, the nondecreasing rearrangement of $\mathbf{y}$ is represented by $\mathbf{y}^* = [10, 14, 18, 22, 26, 30, 34, 38]^\intercal$.

From the permutation map π, we can derive its inverse mapping π^{-1}, which is $\{(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3),(8,1)\}$. Then we can finally get $\mathbf{z} = [14, 22, 30, 38, 34, 26, 18, 10]^\intercal$. Note that the distinctive hat pattern from $\mathbf{x}$ is preserved here using the new set of values from the statistical information.

We note that reconstruction errors from the extrapolation or the zero-shot learning arise from two sources: the generator/predictor output $\mathbf{x}$ and the statistical information $\mathbf{y}$. The following two lemmas state error-free conditions of these two sources.

[2] Without loss of generality, we assume that there are no duplicate entries in $\mathbf{y}$.

Listing 3: Prompt for statistical similarity based data reduction

```
llama_sys_message= "The user provides you with two sequences. One is
    the input sequence and another is the statistical distribution
    sequence. You have to predict the values of the input sequence,
    the predicticted sequence must be similar to the statistical
    distribution sequence. The predicted sequence is of only 24 words
    separated by only commas with no spaces and no other characters in
     string. Please do not produce any additional text."
few_shot_prompt_input1= "Please kindly produce the predicted sequence
    where its sequence depends on the statistical distribution
    sequence, which is: "
few_shot_prompt_input2= "The input sequence is: "
input1= "Please kindly produce the predicted sequence where its
    sequence depends on the statistical distribution sequence, which
    is: "
input2= "The input sequence is: "

chat_completion = chat(
    model="llama3.1:8b",
    options={
        "temperature": 0.3,
        #top_p= 1,
        "seed":int(time.time()),
    },
    messages=[
        {
            "role": "system",
            "content":  llama_sys_message
        },
        {
            "role": "user",
            "content": few_shot_prompt_input1+
                few_shot_prompt_stat_info+few_shot_prompt_input2+
                few_shot_prompt_generic_data
        },
        {
            "role": "assistant",
            "content": few_shot_prompt_TTM_output
        },
        {
            "role": "user",
            "content": input1+stat_info+input2+generic_data
        },
    ]
)
```

Lemma C.1. *The generator/predictor output* $\mathbf{x}$ *is free of errors if and only if its permutation map* π *is identical to that of the vector of ground-truth values* Y*.*

Proof. The quality of $\mathbf{x}$ is solely specified by how close its pattern is to the pattern of Y. More specifically, the relative position of each entry in $\mathbf{x}$, which can be revealed by sorting, should be close to that of Y. Sorting, whether it is rearranging values in the nondecreasing order or nonincreasing order, can naturally produce the permutation map π. If π is identical to that of Y, then the relative positions of values in $\mathbf{x}$ and Y are the same. The converse is also true by the definition of π. □

To see the error-free condition of Lemma C.1 with examples, let $Y = \mathbf{x} + C \cdot \mathbf{1}^\intercal$, where $C \in \mathbb{R}$ is an arbitrary constant. In this example, the difference between Y and $\mathbf{x}$ is explained by an offset C. In this case, the permutation maps for Y and $\mathbf{x}$ are identical to each other and therefore satisfy the error-free condition. As another example, let $Y = C \cdot \mathbf{x}$, where $C \in \mathbb{R}_{>0}$. Here, the difference between Y and $\mathbf{x}$ is explained by a multiplicative factor C, which also satisfies the error-free condition. Moreover, individual perturbations of entries do not break the error-free condition as far as the

permutation map remains intact. For instance, from our initial example of $\mathbf{x} = [1, 3, 5, 7, 6, 4, 2, 0]^\intercal$, we can add 0.5 to the first half of entries such that $\mathbf{x}' = [1.5, 3.5, 5.5, 7.5, 6, 4, 2, 0]^\intercal$. Here, the permutation maps of $\mathbf{x}$ and $\mathbf{x}'$ are the same.

Lemma C.2. *The statistical information* $\mathbf{y}$ *is free of errors if and only if its nondecreasing/nonincreasing rearrangement is identical to that of* Y.

Proof. The quality of $\mathbf{y}$ is solely specified by how close its entries are to the entries of Y, regardless of order. This is because the statistical information, whether it is from the summary statistics or the statistical similarity based data reduction, is the collection of empirical measures of samples. Thus, rearranging values in the nondecreasing/nonincreasing order for both $\mathbf{y}$ and Y can disclose whether they are from the same reference distribution. Conversely, if $\mathbf{y}$ and Y exhibit the same empirical measures of samples, then their rearrangements should be identical to each other. □

By Lemma C.2, we note that any permutations of $\mathbf{y}$ and Y do not break the error-free condition. With these two lemmas, we can state the following theorem.

Theorem C.1. *Merging by Algorithm 1 for the extrapolation or the zero-shot learning is free of errors if and only if both the generator/predictor output* $\mathbf{x}$ *and the statistical information* $\mathbf{y}$ *are free of errors.*

Proof. Algorithm 1 only uses error-free conditions from Lemmas C.1 and C.2. And Y can be completely decomposed into the relative positions and values of its entries. Since there is no other source of errors, satisfying both error-free conditions yields zero reconstruction errors by Algorithm 1. The converse is also true, since the same vectors yield the same permutation maps and the same rearrangement results. □

Corollary C.1. *If* $\mathbf{x}$ *is error-free, then the sum of squared error is measured by* $\|\mathbf{y}^* - Y^*\|^2$*, where* Y^* *is the nondecreasing arrangement of* Y.

While it is straightforward to measure the reconstruction errors when $\mathbf{x}$ is error-free, it is not the case when $\mathbf{x}$ is not error-free. And the reconstruction errors are in practice compound errors from two sources of errors.

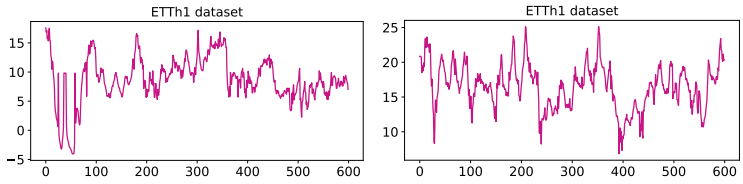
D Details of experiments

Overall, we used seven datasets in our experiments as follows.

1. L1MAG, L2MAG, L3MAG (Stewart et al., 2016): The three datasets were collected by sensors (μPMUs) for monitoring the power grid installed in Lawrence Berkeley National Laboratory. They contain three-phase magnitude measurements of line voltage.
2. Gold prices (Möbius, 2020): The variations of the gold prices.
3. LIC share prices (Sahoo, 2022): The open share price fluctuations of Life Insurance Corporation of India.
4. Bike sharing (rental) demand (Fanaee-T & Gama, 2014): The number of rentals per day in the Capital Bikeshare program in Washington, D.C.
5. Electricity Transformer Temperature (ETTh1) (Zhou et al., 2021): An hourly collection of oil temperature of the electricity transformer temperature dataset.

The experimental results are compared qualitatively using plots and quantitatively using well-known error measures. The errors employed here are mean squared error (MSE), $1/n \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$; mean absolute error (MAE), $1/n \sum_{i=1}^{n} |Y_i - \hat{Y}_i|$; and symmetric mean absolute percentage error (SMAPE), $100/n \sum_{i=1}^{n} |Y_i - \hat{Y}_i| / ((|Y_i| + |\hat{Y}_i|)/2)$. Here, n is the number of samples, Y is the vector of ground-truth values, and $\hat{Y}$ is the vector of estimated values.

We measure the errors between ground-truth data and the reconstructed time series from our proposed methodology. The lower values of errors are interpreted as the reconstructed data being closer to the ground truth. For a fair comparison, in the statistical similarity based data reduction, the



(a) Pattern 1. (b) Pattern 2.

Figure 9: Example patterns in ETTh1 dataset.

original data blocks (i.e., first instances of exchangeable blocks) carried over to the statistical information in Figure 1(b) are excluded from the evaluation since they are the ground truth themselves. Since neural networks yield different results every time they are executed and random sampling is involved with the summary statistics, to have a stable understanding, we provide the average results of five runs.

For both extrapolation and zero-shot learning, we divided each dataset into two equal parts, i.e., an initial/generic part and a later part. Specifically, the length of each part is 3768, yielding the total length of 7536.

Six generative/predictive models employed in this paper used the following hyperparameters. All other unspecified hyperparameters follow their default settings.

1. TimeGAN: `sequence length 24, iterations 15000`
2. DoppelGANger: `maximum length of each training sample 24, sample length 1, epochs 600`
3. Informer: `univariate prediction, sequence length 96, label length 72, prediction length 24`
4. Diffusion-TS: `window (sequence length) 72, sample size 72, batch size 96, dataloader test dataset proportion 0.0001`
5. LLMTime: prompts specified in Appendix B
6. TTM: `context length 90, prediction length 24, fewshot percent 100`

D.1 Electricity Transformer Temperature (ETTh1) dataset

ETTh1 exhibits neither a very high dynamic range nor recurring patterns. It is similar to many real-world datasets with their values ranging from -5 to 25. Sample patterns of ETTh1 are shown in Figure 9.

D.2 Impact of varying block sizes

We utilize block-wise processing in this work to incorporate statistical information, where the block length is kept the same (24) across all the models. Here the block size plays the role of navigating trade-offs involved in reducing the volume of statistical summary at the expense of introducing more errors in reconstructing data blocks, which is connected to the rate distortion theory (Sayood, 2017). We choose the block size of 24 as larger blocks will incur higher errors; but yet creating statistical summary from smaller blocks would defeat the purpose of summarization.

Figures 10, 11, and 12 show how varying block sizes impact the extrapolation errors and the zero-shot learning errors. Here we selected Informer and three datasets from the previous tables for simplicity: L1MAG, Gold Price, and LIC. In the figures, each error is followed by the statistical summary type: summary statistics (Statistics) and statistical similarity based data reduction (IDEALEM). Overall, we can observe that errors increase as a block size increases. We also observe that as discussed in Section 5, both the extrapolation and the zero-shot learning show similar performance, and in some cases, the zero-shot learning outperforms the extrapolation results.

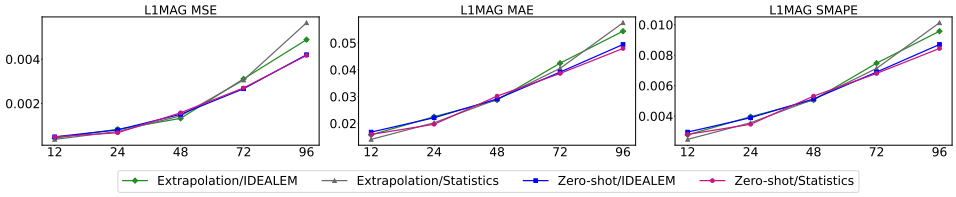


Figure 10: L1MAG errors vs. varying block sizes with Informer.

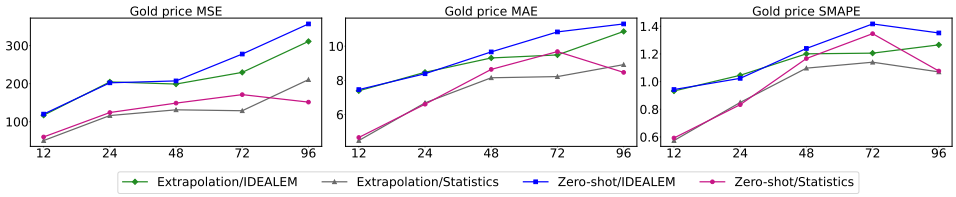


Figure 11: Gold price errors vs. varying block sizes with Informer.

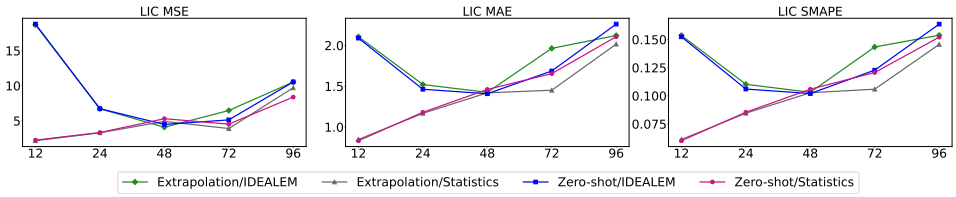


Figure 12: LIC errors vs. varying block sizes with Informer.

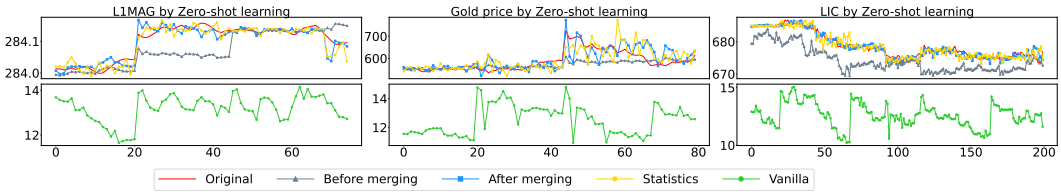


Figure 13: Example plots for reconstruction by Informer zero-shot learning and summary statistics.

E ADDITIONAL RESULTS

We report the extrapolation and the zero-shot learning results of TimeGAN results in Tables 8 and 9, where we present three different types by the statistical summary type: original (Vanilla), summary statistics (Statistics), and statistical similarity based data reduction (IDEALEM).

We also curate the best average results from Tables 1, 2, 3, 4, 5, and 6 and report them along with their standard deviations in Tables 10 and 11. We also present results from Table 7 here again with standard deviations for comparison in Table 12.[3]

F ABLATION STUDY

Figure 13 shows example results of reconstruction for the zero-shot learning. Since vanilla generative/predictive models can only produce outputs similar to ETTh1, resulting vanilla outputs are far from other datasets in question. Similar to Figure 5 though, the results can be significantly improved by the inference phase with the statistical information (Before merging) and further refined by merging (After merging).

[3]We note that statistical similarity based data reduction (IDEALEM) results are deterministic.

Table 8: Extrapolation errors from TimeGAN

Type	Vanilla			Statistics			IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0266	0.1364	0.0240	0.0009	0.0220	0.0143	0.0009	0.0230	0.0140
L2MAG	0.0173	0.1076	0.0190	0.0011	0.0239	0.0151	0.0012	0.0270	0.0150
L3MAG	0.0225	0.1231	0.0217	0.0011	0.0241	0.0150	0.0014	0.0282	0.0151
Gold Price	31220	157.91	18.216	106.96	6.2480	12.932	202.69	8.1560	12.985
LIC	8972.4	76.198	5.1252	3.5139	1.1926	1.1878	7.2480	1.5214	1.1998
Bike Rental	40427	156.95	57.498	39541	158.76	50.876	44331	167.14	50.170
ETTh1	269.29	13.689	38.130	6.6699	1.9031	22.664	9.6482	2.2376	23.197

Table 9: Zero-shot learning errors from TimeGAN

Type	Vanilla			Statistics			IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	67907	260.43	84.837	0.0009	0.0217	0.0143	0.0009	0.0230	0.0140
L2MAG	67603	259.85	84.824	0.0010	0.0230	0.0151	0.0012	0.0265	0.0150
L3MAG	68025	260.66	84.875	0.0011	0.0231	0.0150	0.0013	0.0278	0.0151
Gold Price	135286	353.01	87.714	119.36	6.4572	12.932	207.91	8.3076	12.985
LIC	446543	668.02	93.555	3.1974	1.1451	1.1878	7.3727	1.5391	1.1998
Bike Rental	48346	166.15	67.601	43372	169.97	50.876	46381	171.29	50.170

Tables 13 and 14 show the reconstruction errors of the datasets for two cases: 1) before applying the merging process (Before Merging); 2) when the merging process is applied directly to the baselines (Vanilla Merging). Here we selected three datasets from the previous tables for simplicity: L1MAG, Gold Price, and LIC. And we also selected best-performing models corresponding to the best results for each dataset from the previous tables. In the following tables, each case is followed by the statistical summary type: summary statistics (Statistics) and statistical similarity based data reduction (IDEALEM).

Specifically, we can compare Table 13 with Tables 2 and 3; Table 14 with Tables 5 and 6. The results show that for both cases (i.e., Before Merging and Vanilla Merging), errors are in between baselines and the proposed methodology. We also note that the merging process plays a significant role in reducing errors, since Vanilla Merging consistently shows much better performance than Before Merging does.

Table 10: Best overall extrapolation errors with standard deviations

Type	Vanilla			Statistics			IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	**0.0133±0.0002**	**0.0933±0.0011**	**0.0164±0.0002**	**0.0007±0.0001**	**0.0202±0.0009**	**0.0036±0.0002**	**0.0008±0.0000**	**0.0215±0.0002**	**0.0038±0.0000**
L2MAG	**0.0196±0.0005**	**0.1177±0.0005**	**0.0208±0.0001**	**0.0008±0.0000**	**0.0201±0.0005**	**0.0036±0.0001**	**0.0011±0.0000**	**0.0245±0.0007**	**0.0043±0.0001**
L3MAG	**0.0146±0.0002**	**0.1002±0.0007**	**0.0176±0.0001**	**0.0008±0.0000**	**0.0203±0.0003**	**0.0036±0.0001**	**0.0012±0.0000**	**0.0259±0.0004**	**0.0046±0.0001**
Gold Price	**19064±292**	**108.52±0.99**	**13.189±0.107**	**108.88±2.42**	**6.2570±0.2876**	**0.8009±0.0022**	**158.16±14.31**	**7.2229±0.1789**	**0.9280±0.0198**
LIC	**5929.6±4.5**	**58.225±0.015**	**3.9040±0.0009**	**3.2795±0.0640**	**1.1058±0.0365**	**0.0798±0.0026**	**5.4913±0.3244**	**1.1703±0.0396**	**0.0847±0.0028**
Bike Rental	**38071±207**	**146.75±0.89**	**51.055±3.205**	**35389±944**	**148.62±0.78**	**48.587±0.199**	**33393±118**	**137.02±0.36**	**42.863±0.006**
ETTh1	**290.20±21.07**	**14.168±0.448**	**40.560±0.573**	**5.3836±0.8185**	**1.7212±0.1743**	**9.5931±0.7832**	**7.6721±0.0048**	**1.9967±0.0011**	**10.830±0.013**

Table 11: Best overall zero-shot learning errors with standard deviations

Type	Vanilla			Statistics			IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	**65634±29**	**256.15±0.06**	**82.181±0.033**	**0.0008±0.0000**	**0.0208±0.0001**	**0.0037±0.0000**	**0.0008±0.0000**	**0.0219±0.0004**	**0.0039±0.0001**
L2MAG	**65346±27**	**255.59±0.05**	**82.174±0.031**	**0.0009±0.0000**	**0.0215±0.0006**	**0.0038±0.0001**	**0.0011±0.0000**	**0.0244±0.0004**	**0.0043±0.0001**
L3MAG	**65798±31**	**256.47±0.06**	**82.276±0.035**	**0.0009±0.0000**	**0.0220±0.0001**	**0.0039±0.0000**	**0.0012±0.0001**	**0.0261±0.0008**	**0.0046±0.0001**
Gold Price	**126175±78**	**342.02±0.10**	**85.505±0.045**	**108.88±2.42**	**6.4842±0.0812**	**0.8175±0.0015**	**158.57±25.80**	**7.4355±0.4187**	**0.9480±0.0427**
LIC	**439396±141**	**662.67±0.11**	**92.124±0.029**	**2.7500±0.0102**	**1.0986±0.0013**	**0.0795±0.0001**	**5.2271±0.2094**	**1.1803±0.0339**	**0.0854±0.0025**
Bike Rental	**45181±26**	**159.20±0.04**	**64.179±0.012**	**35727±1781**	**148.01±4.87**	**48.533±1.018**	**33465±47**	**140.79±0.10**	**42.866±0.016**

Table 12: Statistical summary only errors with standard deviations

Type	Statistics			IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0009±0.0000	0.0225±0.0003	0.0040±0.0001	0.0009	0.0227	0.0040
L2MAG	0.0010±0.0000	0.0236±0.0002	0.0042±0.0000	0.0012	0.0264	0.0047
L3MAG	0.0010±0.0000	0.0240±0.0002	0.0042±0.0000	0.0013	0.0279	0.0049
Gold Price	112.85±6.48	6.6332±0.0955	0.8461±0.0109	205.63	8.3249	1.0299
LIC	3.1152±0.0674	1.1359±0.0153	0.0823±0.0011	6.0349	1.3076	0.0948
Bike Rental	41562±623	161.35±0.90	50.017±0.202	30457	116.75	35.275
ETTh1	7.2914±0.1366	2.0431±0.0184	11.445±0.177	8.5982	1.9515	10.934

Table 13: Ablation study for extrapolation errors before merging and with vanilla merging

Case/Type	Before Merging/Statistics			Vanilla Merging/Statistics			Before Merging/IDEALEM			Vanilla Merging/IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0028	0.0414	0.0073	0.0009	0.0228	0.0040	0.0035	0.0464	0.0082	0.0009	0.0226	0.0040
Gold Price	452.33	184.79	2.1455	121.51	6.5889	0.8020	247.16	8.3157	1.0332	181.76	7.7362	0.9751
LIC	14.705	4.2301	0.3099	3.4269	1.0697	0.0773	9.2522	1.4636	0.1060	5.9419	1.2812	0.0924

Table 14: Ablation study for zero-shot learning errors before merging and with vanilla merging

Model/Statistical	Before Merging/Statistics			Vanilla Merging/Statistics			Before Merging/IDEALEM			Vanilla Merging/IDEALEM		
Data	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
L1MAG	0.0105	0.0833	0.0147	0.0011	0.0241	0.0042	0.0009	0.0229	0.0040	0.0009	0.0224	0.0040
Gold Price	452.33	14.741	13.314	107.98	6.5976	0.7958	273.96	8.4518	1.0388	200.62	8.1574	1.0194
LIC	431.54	17.857	1.2854	2.8058	1.1280	0.0816	7.1293	1.3650	0.0989	5.6248	1.2632	0.0914