Evaluating the Impact of Error-Bounded Lossy Compression on Time Series Forecasting

Carlos Enrique Muniz-Cuza
Aalborg University
cemc@cs.aau.dk

Søren Kejser Jensen
Aalborg University
skj@cs.aau.dk

Jonas Brusokas
Aalborg University
jonasb@cs.aau.dk

Nguyen Ho
Loyola University Maryland
tnho@loyola.edu

Torben Bach Pedersen
Aalborg University
tbp@cs.aau.dk

ABSTRACT

Time series data is widely used for decision-making and advanced analytics such as forecasting. However, the vast data volumes make storage challenging. Using lossy compression can save more space compared to lossless methods, but it can affect the forecasting accuracy. Understanding the impact of lossy compression on forecasting accuracy is a multifaceted challenge, necessitating experimental evaluation across various forecasting models, compression methods, and time series. This paper conducts such experimental evaluation by combining seven forecasting models, three lossy compression algorithms, and six datasets. By simulating a real-life scenario where forecasting models use lossy compressed data for prediction, we address three main research questions related to compression error and its effects on the time series characteristics and the forecasting models.

The results show that the Poor Man's Compression and Swing Filter lossy compression algorithms add less error than the Squeeze method as the error bound increases. Poor Man's Compression provides the best balance between compression ratio and forecasting accuracy. Specifically, we obtained an average compression ratio of 13.65, 5.56, and 14.97 for PMC, SWING, and SZ with an average impact on forecasting accuracy of 5.56%, 3.3%, and 8.5%, respectively. An analysis of several time series characteristics shows that the maximum Kullback-Leibler divergence between consecutive windows in the time series is the best indicator of the impact of lossy compression on forecasting accuracy. Finally, our results indicate that simple models like Arima, are more resilient to lossy compression than complex deep learning models. The source code and data are available at *https://github.com/cmcuza/EvalImpLSTS*.

1 INTRODUCTION

High-frequency time series generated by reliable sensors are rapidly transforming various industries such as manufacturing and renewable energy generation. This data is particularly important for optimizing the operation of industrial systems including wind turbines. However, the huge amount of data points generated by each sensor makes its transfer and storage infeasible. Lossless compression cannot solve the problem, e.g., for smart meters, it only provides compression ratios (CRs) between 2x and 4x [45]. Lossy compression provides significantly higher CRs at the expense of a small amount of error [11, 17, 33, 40]. However, the broad application of lossy compression in industry is still limited by a lack of understanding of how the decompression error affects data analytics such as time series forecasting.

As an example, consider a wind turbine transferring data to the cloud for analysis through a communication network. To reduce the bandwidth required, the time series are lossy compressed on the wind turbine [29, 53, 58]. Once the data is transferred to the cloud, the operators use it to forecast different variables to optimize energy production and make informed decisions about predictive maintenance. However, applying lossy compression can affect the accuracy of the forecasting models and thus cause incorrect decisions. Thus, it is necessary to select the combination of lossy compression method and forecasting model that provides the highest forecasting accuracy. However, this is a challenging task given the large number of possible combinations. Moreover, lossy compression algorithms generally require a user-defined error bound. This is hard to set apriori since its impact on forecasting accuracy is not well understood. Other factors like data characteristics and error distribution also impact forecasting accuracy. To determine, understand, and predict this impact, an experimental evaluation is necessary.

This paper conducts such an evaluation using seven well-known time series forecasting models, three lossy compression algorithms, and six time series datasets. We focus on pointwise error-bounded lossy time series compression methods as they possess the advantage of preserving the data's structure and outliers thus providing users with fine-grained control over the loss of information. Specifically, we evaluate the forecasting models: Arima [5], Gradient Boosting [7, 13], NBeats [42], Informer [65], DLinear [62], Transformer [18], and GRU [47]; the lossy compression algorithms Poor Man's Compression (PMC) [33], Swing Filter [11], and Squeeze (SZ) [35]; the datasets ETTm1, ETTm2 [65], Solar [61], Weather [66], ElecDem [15] and Wind (released as part of this paper); and different error bounds as parameters of the lossy compression methods. In addition, we analyze 42 time series characteristics that deal with shifts in the series' distribution, curvature, and stability, and conduct multiple correlation and regression analyses, to answer the three research questions:

- **RQ1:** *How does lossy compression affect time series?*
- **RQ2:** *How does lossy compression affect the accuracy of time series forecasting?*
- **RQ3:** *How does lossy compression affect individual forecasting models?*

Experimental results show that it is possible to obtain high CRs without significantly affecting forecasting accuracy. Specifically, the lossy compression methods obtain an average CR of 13.65 for PMC, 5.56 for SWING, and 14.97 for SZ with an average impact on forecasting accuracy of 5.5%, 3.5%, and 8.5%, respectively.

The maximum Kullback-Leibler divergence between consecutive windows emerged as the most important characteristic for predicting the impact of lossy compression on forecasting accuracy. Finally, our results suggest that models that heavily rely on capturing short-term fluctuations, like Transformer, experience a more significant drop in forecasting accuracy, while, models that prioritize broader trends, like Arima, remain more resilient. Throughout this paper, we make suggestions about which compression algorithm to use based on their CR, decompression error, and impact on forecasting accuracy.

In summary, we make the following contributions:

(1) We provide extensive experimental results of the impact that three well-known lossy compression methods have on seven forecasting models using six different datasets.
(2) We systematically analyze the relation between decompression error, compression ratio, and forecasting accuracy for multiple pointwise error bounds.
(3) We evaluate which time series characteristics are the best predictors of the impact of lossy compression on forecasting accuracy. We also provide guidelines on how to monitor the most important characteristics.
(4) We analyze the resilience of individual forecasting models to lossy compression on each dataset and provide empirical evidence regarding the time series characteristics associated with that resilience.

The paper's structure is the following. Section 2 introduces the definitions used throughout the paper. Section 3 describes the setup of our experimental evaluation. Section 4 analyses the experimental results and describes our major findings. Section 5 provides future research directions. Section 6 describes related work. Finally, Section 7 provides the conclusion and future work.

2 PRELIMINARIES

Definition 1. *A **time series** (ts) is a list of data points indexed in time order, e.g., $X = \langle (t_1, v_1), ..., (t_n, v_n) \rangle$. Each data point (t_i, v_i), is a tuple consisting of a timestamp t_i and the corresponding value v_i. The timestamp represents when the value was recorded.*

Definition 2. *A ts X is a **regular time series** if the time elapsed between consecutive data points is constant, i.e., $t_{i+1} - t_i = t_{j+1} - t_j, \forall\ 1 <= i, j < n$.*

Definition 3. *Given a regular time series X, a **segment** $x_s(i, j)$ of X is the list of data points from timestamp t_i to t_j, where $i <= j$.*

Definition 4. *Let x_s be a segment of a regular time series. The **pointwise error bound**, also known as the L_∞ norm, denoted as ϵ, is the maximum allowed difference between values in x_s and their corresponding decompressed values.*

We say that ϵ is relative if it represents the maximum relative allowed error between the values in x_s and the decompressed values, i.e., $\hat{v}_i \in [v_i - v_i\epsilon,\ v_i + v_i\epsilon], \forall\ \hat{v}_i \in x_s$, where $\hat{v}_i$ and v_i are the decompressed and original data point value at timestamp i, respectively. From now on, we will refer to these lossy compressors as *pointwise error-bounded lossy compression* (PEBLC).

Definition 5. *We define the **transformation** T of X, as a mapping between the raw time series X and the decompressed time series $\hat{X}$, i.e., $\hat{X} = T(X)$, where T maps every value of the time series $x_i = (t_i, v_i) \in X$ to its decompressed value $\hat{x}_i = (t_i, \hat{v}_i) \in \hat{X}$.*

Definition 6. *We define **transformation error** (TE) as the difference between the values of X and $\hat{X}$. This difference can be computed by any distance metric D, e.g., Root Mean Square Error.*

The TE metric quantifies the error introduced by lossy compression as measured by a distance metric (details in Section 3.5), thus it only yields positive values.

Definition 7. *Given a list of k values of X, $\langle x_{t-k}, ..., x_t \rangle$, the task of **time series forecasting** aims to predict the values of the next h values denoted by $\langle \hat{y}_{t+1},\ ...,\ \hat{y}_{t+h} \rangle$. These values are inferred by the forecasting model F such that:*

$$\hat{y}_{t+1},\ ,\ ...,\ \hat{y}_{t+h} = F(x_{t-k},\ ...,\ x_t) \tag{1}$$

Definition 8. *We define **forecasting error** (FE) as the difference between $\langle y_{t+1},\ ...,\ y_{t+h} \rangle$ and $\langle \hat{y}_{t+1},\ ...,\ \hat{y}_{t+h} \rangle$. This difference can be computed by any time series distance metric D.*

Definition 9. *We define **transformation forecasting error** (TFE) as the relative difference between the forecasting error obtained with the original and transformed time series as input.*

$$TFE = \frac{D(F(\hat{X}), y) - D(F(X), y)}{D(F(X), y)} \tag{2}$$

TFE can reflect whether transforming the data improves or worsens forecasting accuracy. Specifically, a negative TFE means the forecasting accuracy improved after compressing and decompressing, and a positive value that it degraded.

3 EXPERIMENTAL SETUP

3.1 Datasets

To select the datasets, we consider two criteria: the sampling interval and their use in highly cited time series forecasting papers, including the papers proposing the used forecasting models. For the first criterion, we only consider datasets with a short sampling interval, i.e., high-frequency data (HFD). Using HFD means the lossy compression algorithms can achieve high CRs. This is because in HFD consecutive data points tend to be very similar and a large increase or decrease within a short time is not common [25]. HFD is defined differently in different domains, ranging from a daily to a second sampling interval. [21, 56]. In this paper, we focus on a sampling interval of less than or equal to 30 minutes considering the forecasting literature and the availability of the data [56, 65, 66].

The second criterion, i.e., the use of well-known datasets, enables us to validate our baseline results and reuse the hyperparameter configuration of the forecasting models. Moreover, it facilitates the reproducibility of the results, thus reducing the chance that the forecasting models are incorrectly tuned for the dataset. For these reasons, we use the following datasets:

- **ETT** [65]: It has two subsets ETTm1 and ETTm2. Each contains 2 years of data from an electrical transformer with a sampling interval of 15 minutes. The target variable is the oil temperature of the transformer.
- **Solar** [61]: It contains the power output collected from 137 photovoltaic (PV) plants sampled every 10 minutes. The target variable is the power output of each PV.
- **Weather** [66]: It contains 21 meteorological indicators for a range of 1 year sampled every 10 minutes. The target variable is the CO2 concentration of ambient air.
- **ElecDem** [15]: It contains a time series representing the half-hourly electricity demand of Victoria state in Australia. The target variable is the electricity demand.
- **Wind:** contains the active power output of a wind turbine sampled every 2 seconds for 10 days. The data contains 9 other variables including the rotor and wind speeds. We

Table 1: Details and statistics of datasets.

Dataset	LEN	FREQ	MEAN	MIN	MAX	Q1	Q3	rIQD
ETTm1	69,680	15min	13.32	-4	46	7	18	82%
ETTm2	69,680	15min	26.60	-3	58	16	36	75%
Solar	52,560	10min	6.35	0.0	34	0.0	12	200%
Weather	52,704	10min	427.66	305	524	415	437	5%
ElecDem	230,736	30min	6,740	3,498	12,865	5,751	7,658	28%
Wind	432,000	2sec	363.69	-68	2,030	108	550	121%

are releasing this data with the paper. The target variable is the activate power of the wind turbine.

Table 1 lists their descriptive statistics. The relative InterQuartile Difference rIQD = $\frac{(\text{Q3}-\text{Q1})}{\text{MEAN}} \times 100$, reflects how spread out the middle 50% of the data is relative to the mean of the time series. This statistic will provide insights into the impact that lossy compression has on the different datasets.

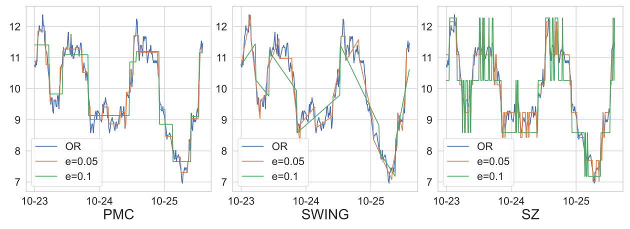
3.2 Error-Bounded Lossy Compression

We have chosen three well-known lossy compression algorithms that have received a lot of coverage in the scientific literature [27, 34, 53]. The three algorithms are PMC (specifically PMC-Mean) [33], SWING [11], and SZ [35]. These algorithms guarantee a relative pointwise error bound that preserves the data's structure and outliers if they are outside the error bound. This error bound provides users with fine-grained control over the loss of information and it can be meaningfully applied across datasets, e.g., $\epsilon = 0.1$ can be meaningfully applied to both the Weather and Solar datasets. For other compression methods like Discrete Cosine and Wavelet Transforms [1, 39], although widely used, supporting a user-defined error bound is not straightforward. This yields a significant limitation in controlling the error introduced during the compression process, a key requirement of our industry collaborators. Moreover, PMC and SWING learn constant and linear approximations which have been shown to represent time series more efficiently than higher-level polynomials [10]. Finally, SZ is one of the most popular lossy compression algorithms with multiple improvements over the years and stable releases. Descriptions of the original algorithms follow:

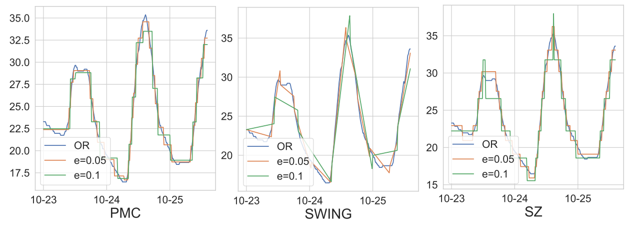
PMC: The algorithm adds data points to an adaptive window updating the mean value in the window. At the same time, the maximal absolute or relative (as in this paper) difference of the points to the mean value is computed. If the difference exceeds the error bound, the window without the latest point is turned into a segment represented by its mean value.

SWING: The algorithm adds data points to an adaptive window updating a linear approximation of the data points in the window. After adding a new point, the algorithm checks if the error between the actual data points and the linear approximation remains within a predefined threshold. If the error exceeds the error bound, the window is turned into a segment, and its values are compressed by the line that minimizes the mean square error.

SZ: The algorithm first splits the dataset into multiple non-overlapping equal-sized segments. For each segment, SZ evaluates different prediction models like classic Lorenzo, mean-integrated Lorenzo, and linear regression, which estimate the value of a new data point based on the values of its neighbors. Based on the prediction results, SZ dynamically selects the best-fit data predictor considering the data points within the segment. After prediction, the algorithm quantizes the difference between the actual data point and its predicted value. This quantization process translates the differences into a smaller set of discrete



(a) ETTm1



(b) ETTm2

Figure 1: PMC, SWING, and SZ compression output at different error bounds compared to the original time series (OR).

symbols, which are easier to compress. Afterward, the quantized values are compressed using Huffman encoding or some other kind of entropy encoder, e.g., arithmetic coding. Finally, SZ applies a general lossless compression, e.g., gzip.

Figure 1 shows an example of the output generated by PMC, SWING, and SZ on a segment of ETTm1 and ETTm2. Note that, the three methods generate very different outputs under the same error bounds of 0.05 and 0.1. At first glance, SZ seems to fit a constant line like PMC, however, this is due to the quantization step. Most likely, quantization is also causing the noticeable fluctuations in short time intervals of the decompressed data.

Implementations Used: We use ModelarDB's [27, 28] implementation of PMC and SWING, and Libpressio's [55] Python wrapper for SZ 2.1. Thus, all of the implementations used a piece-wise relative error-bound. ModelarDB's SWING computes the slope and intercept coefficients as the mean of the upper and lower bound of the linear approximation. In addition, SZ applies gzip as the final lossless compression. To ensure the compression methods can be directly compared we took two extra steps. First, we compress the timestamps for all the methods by storing the first timestamp as a 32-bit integer, the sampling interval as a 16-bit integer, and the length of the generated segments as a 16-bit integer. This extra step was needed as SZ cannot efficiently compress the timestamps, while PMC and SWING need them to reconstruct the time series. Second, we compress the PMC's and SWING's compressed representations using gzip since SZ applies gzip as the final step while the former two do not. Gzip is also applied directly to the raw dataset.

During the evaluation, we compress the datasets using 13 different error bounds (EB) unevenly distributed from 0.01 to 0.8. Concretely, EB = {0.01, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.65, 0.8}. We explore more values when the error bound is lower than 0.1, and as the error bound increases, we increase the distance between the values to cover higher error bounds while keeping the size of EB manageable. For each compression method and error bound, we generate an output file containing the compressed representation of the time series. Additionally, we add a header with the sampling interval, initial timestamp, and the number of data points in the segment to be able to decompress the data. Thus, the size of the raw and compressed data is the number of bytes in each of the generated *.gz* files.

3.3 Lossless Compression Baseline

Facebook's Gorilla (GORILLA) [43] is a compression method used as the default encoding for floating point measurements in many time series databases (TSMSs) [26]. Thus, we use it as a baseline to show the compression ratio that currently can be achieved with lossless compression. In the original paper, the data points are split into two hour blocks. However, for our datasets such as ETTm1, with 15 minutes sampling intervals, this will mean considering only 8 data points thus affecting the compression ratio of the algorithm. Considering this, our implementation of GORILLA compresses the whole time series as a single segment. Each value is compressed by XORing with the previous value followed by a variable-length binary encoding.

3.4 Forecasting Models & Hyperparameters

We have chosen seven prominent forecasting models ranging from classical statistical models to deep learning.

- **Arima** [5]: is the Autoregressive Integrated Moving Average model with Fourier terms as exogenous variables to model long seasonality.
- **Gradient Boosting (GBoost)** [7, 13]: an ensemble of basic predictors that, when combined, form a strong predictive model by iteratively correcting the errors of prior models. We use simple decision trees as the basic predictors.
- **GRU** [47]: widely used for time series forecasting, it is an encoder-decoder Gated Recurrent Neural Network.
- **NBeats** [42]: a deep neural architecture based on backward and forward residual links and a very deep stack of fully connected layers.
- **Transformer** [18]: an encoder-decoder architecture that has as its core feature a multi-head attention mechanism that learns intra-dependencies and inter-dependencies within the input time series.
- **Informer** [65]: an improvement over the Transformer model with a probabilistic sparse self-attention mechanism which achieves better time complexity and a generative style decoder to acquire long sequence output with only one forward step.
- **DLinear** [62]: a shallow neural architecture that decomposes the time series into trend and remainder, and employs a feed-forward layer per each component.

To conduct the hyperparameter search, we first explore the literature and reuse the deep learning model's suggested hyperparameter. Moreover, we follow a standard hyperparameters search strategy in case the model has not been tested on a dataset [14]. Specifically, we split the datasets into training/validation/testing using 70%/10%/20% for each set. Then, using the validation subset we conducted a grid search over the hyperparameters of the models: the number of units, layers, blocks, heads, and stacks. We search for values around the suggested hyperparameters to guide and limit the search space. The dropout hyperparameter was searched for all models within the values [0, 0.05, 0.1]. We use a learning rate equal to 0.001 and a weight decay equal to 0.0001 as default parameters for the Adam optimizer [31]. During training, we use an early-stop strategy on the validation subset with patience 3. Moreover, we apply a standard scaler to the input of the forecasting models and fix its size to 96 previous timestamps following previous work [65]. We set the forecasting horizon to 24 timestamps into the future. For Solar, the input size of DLinear is set to 720 as suggested for multivariate time series [62]. In the case of Arima, we use the Akaike Information Criterion (AIC) [19] to select the best model.

3.5 Evaluation Metrics

Compression Ratio: The compression ratio is defined as the ratio between the size of the raw dataset and the size of its compressed representation as follows:

$$CR = \frac{size\_of\_raw\_data}{size\_of\_compressed\_data} \tag{3}$$

Thus, a higher *CR* means the compression reduced the size of the data more. As mentioned before, the sizes of the raw dataset and its compressed representation are the numbers of bytes in the generated .gz files.

Transformation and Forecasting Error: To compute the transformation and forecasting error, we use three metrics: Root Mean Square Error (RMSE), Root Relative Squared Error (RSE), and Normalized Root Mean Square Error (NRMSE). RMSE is a classic metric used to compare time series. NRMSE and RSE facilitate the comparison between datasets that have different scales [3, 49]. Moreover, we use the correlation metric (R) to measure the similarity between the raw and transformed time series, and as an evaluation metric of the forecast accuracy.

$$NRMSE = \frac{RMSE(x,y)}{R_x} \tag{4}$$

$$RMSE = \sqrt{\frac{\sum_i^n (x_i - y_i)^2}{N}} \tag{5}$$

$$R = \frac{\sum_{i=1}^n ((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \tag{6}$$

$$RSE = \frac{\sqrt{\sum_i^n (x_i - y_i)^2}}{\sqrt{\sum_i^n (x_i - \bar{x_i})^2}} \tag{7}$$

where $R_x = max(x) - min(x)$, x is the raw time series, and $\bar{x}$ and $\bar{y}$ are the mean value of x and y which represents the predicted or transformed time series. The metrics RMSE, NRMSE, and RSE measure the distance between the two input time series, thus, lower values indicate that the time series are more similar. R interpretation is the opposite, the higher the value, the more correlated the time series thus closer to each other.

3.6 Evaluation Scenario

We consider one evaluation scenario based on the wind turbine example described in Section 1 to conduct our experiments. In the example, the time series is compressed directly in the wind turbine and transmitted to a central analysis system. Assuming there exists a forecasting model already trained on previously collected segments of the raw time series, it is desirable to understand the impact of lossy compression on forecasting accuracy. Alternative scenarios include retraining the model using the transformed data. However, performing such an evaluation requires evaluating different incremental machine-learning (ML) techniques which fall outside the scope of this paper. Algorithm 1 shows the procedure followed.

The evaluation starts by splitting the dataset into training, validation, and testing, and training the forecasting model on the training subset. We pre-computed the hyperparameters for each combination of model and dataset according to Subsection 3.4 using the validation subset. In Algorithm 1, $test.y$ and $\hat{test}.x$ refers to the set of all output sequences of the raw test subset and the set of all input of the transformed test subset, respectively. $T(test|\ C, \epsilon)$ refers to transforming the testing data given the lossy compression method C with error bound ϵ. The function $calculateMetrics()$ returns the RMSE, RSE, and NRMSE evaluation metrics. After transforming the testing subset, we obtain

Algorithm 1 Evaluation Procedure

```
Input: ModelID $F_{id}$, Dataset $X$
1:  function EvaluateScenario
2:      $results \leftarrow []$
3:      $train,\ val,\ test \leftarrow split(X)$
4:      $F \leftarrow trainModel(F_{id},\ train,\ val)$
5:      for $C \in [PMC,\ SWING,\ SZ]$ do
6:          for $\epsilon \in EB$ do
7:              $\hat{test} \leftarrow T(test|\ C, \epsilon)$ /*transform test*/
8:              $\hat{pred} \leftarrow F.predict(\hat{test}.x)$
9:              $results.append(calculateMetrics(test.y, \hat{pred}))$
10:         end for
11:     end for
12:     return $results$
13: end function
```

the model's prediction using as input the transformed data. The metrics are collected by comparing the raw data on the testing subset against the predictions of the model. To reduce the effect of the random initialization of the forecasting models, we run *EvaluateScenario* multiple times and report the mean values using different random seeds. We run the deep neural network models 10 times and the rest of the models only 5 times as they are less affected by random initialization.

Once the metrics are collected, we compare the results against the results obtained when using the raw data without applying any lossy compression, i.e., the baseline results. The baseline consists of performing the prediction and collecting the evaluation metrics using the raw testing subset *test*. For the baseline, we also compute 10 or 5 different baseline instances using different random seeds to reduce the effects of random initialization. Only the mean is reported.

3.7 Baseline Results

Table 2 shows the baseline results for the evaluation scenario. Overall, GRU's accuracy is the worst, especially on ETTm2, Solar, and ElecDem compared to the other models. The simpler models, DLinear and Arima, obtain very good results overall and even perform best in some datasets like ETTm1. In contrast, on Solar, these models perform worse in comparison to the complex models since they are unable to learn the correlation between the 137 series. Finally, NBeats obtains the best results on Solar and Wind while Informer and Transformer obtain the best results on ETTm2 and Weather, respectively. We use the results presented in Table 2 throughout the remainder of the paper to compute the TFE. Specifically, given an evaluation metric, a dataset, and a forecasting model, the respective entry in Table 2 constitutes the $D(F(X), y)$ factor in Eq. 2.

Table 2: Evaluation scenario baseline results (best in bold).

Model	**Metric**	ETTm1	ETTm2	Solar	Weather	ElecDem	Wind
Arima	R	0.95	**0.97**	0.82	0.87	0.78	0.98
	RSE	0.32	**0.24**	0.60	0.53	0.58	0.19
	RMSE	0.12	**0.15**	0.51	0.46	0.5	0.16
	NRMSE	0.054	**0.045**	0.164	0.08	0.081	0.036
Gboost	R	0.93	0.95	0.90	0.87	0.95	0.98
	RSE	0.39	0.34	0.45	0.50	0.34	0.17
	RMSE	0.14	0.21	0.39	0.45	0.29	0.15
	NRMSE	0.067	0.061	0.12	0.07	0.047	0.033
DLinear	R	**0.96**	0.96	0.91	**0.90**	0.94	0.98
	RSE	**0.30**	0.26	0.41	**0.46**	0.36	0.17
	RMSE	**0.10**	0.16	0.35	**0.41**	0.31	0.15
	NRMSE	**0.051**	0.049	0.095	**0.07**	0.051	0.033
GRU	R	0.94	0.87	0.78	0.89	0.58	0.98
	RSE	0.38	0.5	0.79	0.46	0.96	0.17
	RMSE	0.13	0.31	0.68	0.41	0.83	0.15
	NRMSE	0.065	0.091	0.184	0.069	0.14	0.033
Informer	R	0.94	**0.97**	0.93	0.89	**0.96**	0.99
	RSE	0.39	**0.24**	0.39	0.48	**0.29**	0.17
	RMSE	0.13	**0.15**	0.34	0.43	**0.25**	0.15
	NRMSE	0.067	**0.045**	0.091	0.07	**0.040**	0.032
NBeats	R	0.95	0.97	**0.94**	0.88	0.96	**0.99**
	RSE	0.36	0.26	**0.34**	0.48	0.29	**0.17**
	RMSE	0.12	0.16	**0.30**	0.43	0.25	**0.14**
	NRMSE	0.062	0.048	**0.080**	0.074	0.041	**0.032**
Transformer	R	0.95	0.97	0.93	0.82	0.96	0.99
	RSE	0.37	0.27	0.38	0.57	0.30	0.17
	RMSE	0.13	0.17	0.33	0.51	0.26	0.15
	NRMSE	0.063	0.049	0.089	0.086	0.042	0.032

4 RESULTS ANALYSIS

4.1 Research Questions

In this section, we define our three main research questions (**RQs**) and relevant related sub-questions that guided our analysis. Firstly, as we are evaluating the impact of lossy compression on time series forecasting, it is necessary to understand what the impact of lossy compression on time series is. In this regard, the three aspects that can be analyzed are the piecewise relative error bound, TE, and CR.

RQ1: How does lossy compression affect time series?

- **RQ1.1:** Which lossy compression method best preserves the data quality as the error bound increases?
- **RQ1.2:** Which lossy compression method provides the highest CR?
- **RQ1.3:** How are TE and CR related?

Secondly, and our main focus, is the impact of lossy compression on the accuracy of time series forecasting. Intuitively, as the error bound increases, the accuracy should decrease. However, determining how the accuracy decreases and which metrics can best explain the negative impact is very important. We also analyze the relationship between CR and TFE to determine how lossy compression can best be used with forecasting.

RQ2: How does lossy compression affect the accuracy of time series forecasting?

- **RQ2.1:** How much TE can be introduced before significantly affecting the forecasting accuracy?
- **RQ2.2:** Which time series characteristics best explain the impact of lossy compression on forecasting accuracy?
- **RQ2.3:** How high CR can be obtained without significantly affecting the forecasting accuracy?

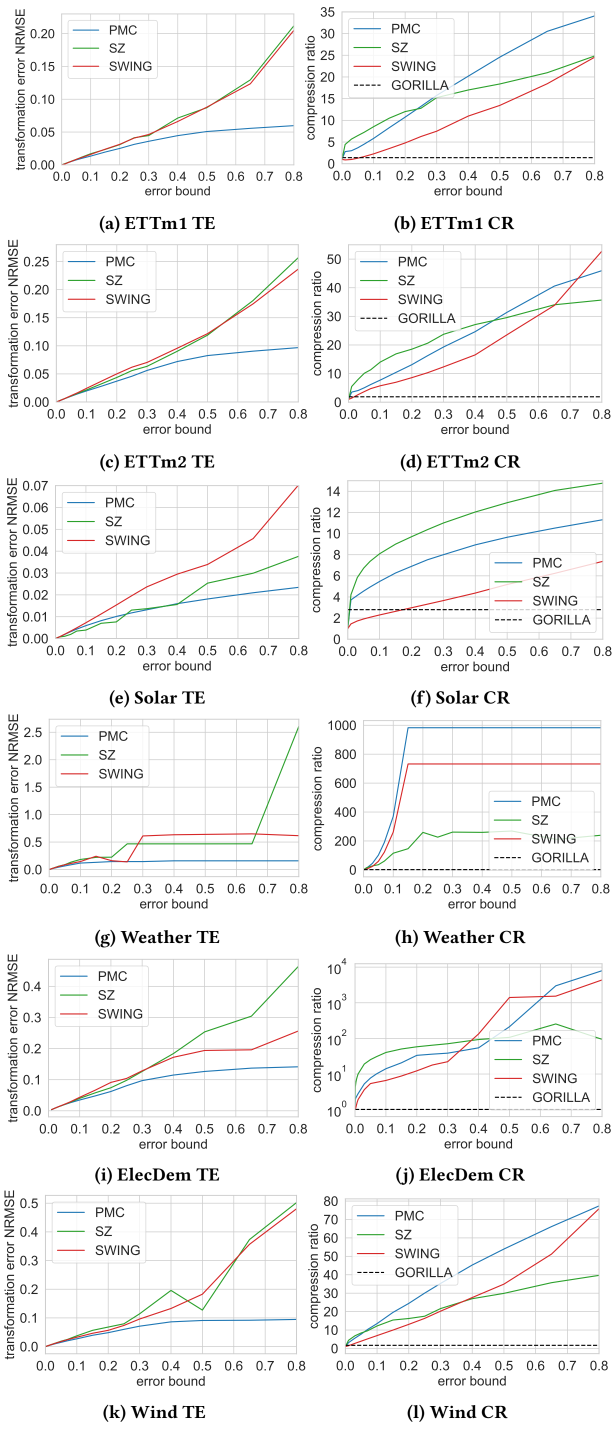
Finally, determining which forecasting model is most resilient to lossy compression can help users select the best model to combine with lossy compression.

RQ3: How does lossy compression affect the individual forecasting models?

- **RQ3.1:** Which factors influence the resilience of the forecasting models?
- **RQ3.2:** Which time series forecasting model is more resilient to the effect of lossy compression?

4.2 TE and CR Evaluation (RQ1)

Figure 2 shows the TE measured by NRMSE and the CR per lossy compression method for the different error bounds and datasets. Additionally, the figure shows GORILLA's CR as a baseline with 1.49x, 2.08x, 3.09x, 2.22x, 2.02x, and 1.73x for ETTm1, ETTm2, Solar, Weather, ElecDem, and Wind, respectively. Note, that the

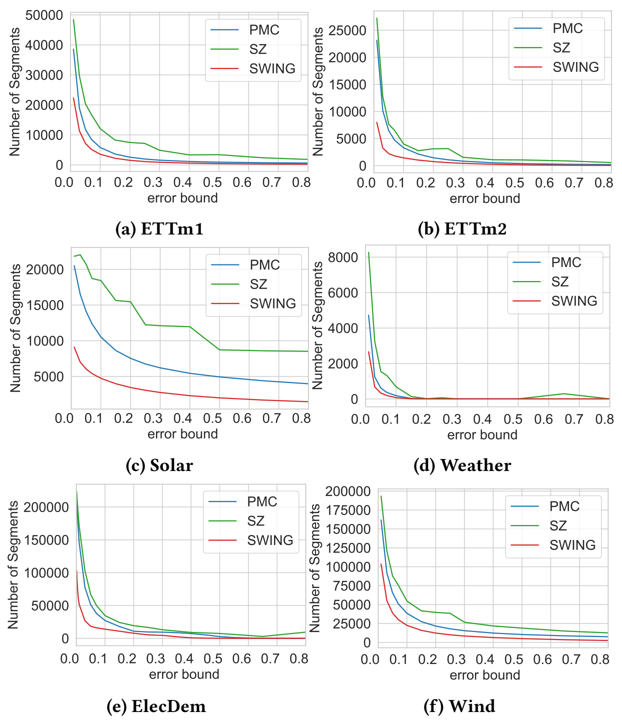


(a) ETTm1 TE (b) ETTm1 CR (c) ETTm2 TE (d) ETTm2 CR (e) Solar TE (f) Solar CR (g) Weather TE (h) Weather CR (i) ElecDem TE (j) ElecDem CR (k) Wind TE (l) Wind CR

Figure 2: TE and CR per dataset and lossy compression.

lossy compression methods provide much higher CRs than GORILLA even at an error bound of 0.01, with the single exception being SWING on Solar. An explanation for this is that during nighttime, the values are always 0. Thus, GORILLA's XOR and variable-length encoding can efficiently compress those segments using a single bit per value while SWING needs two coefficients.

Analyzing the CR among the lossy compression methods, SZ initially excels at lower error bounds due to its linear-scale quantization and Huffman encoding. However, as the error bound increases, PMC's simpler single-coefficient model proves more effective, surpassing both SZ and SWING in CR. For instance, for ETTm1, PMC and SZ create 11889 and 20434 segments at an error bound of 0.05, respectively. However, PMC only creates 899 segments at an error bound of 0.5, compared to SZ's 3397 segments. This trend is evident in Figure 3, which shows the segment counts across the datasets, indicating PMC's better effectiveness with higher error bounds. On the other hand, despite the flexibility of SWING with its two-coefficient model, as indicated by the lower number of segments in Figure 3, it is less efficient in terms of CR. The added storage overhead from its two coefficients becomes a disadvantage, especially after gzip compression. PMC's constant-value segments benefit more from gzip, resulting in a higher CR compared to SWING's slope-intercept pairs. These findings show that simple lossy compression methods like PMC can significantly increase their CR by incorporating lossless compression like gzip, thus avoiding the need for more complex lossy compression methods like SWING.



(a) ETTm1 (b) ETTm2 (c) Solar (d) Weather (e) ElecDem (f) Wind

Figure 3: Number of segments per dataset.

One anomaly that emerges is the exceptionally high CR observed on Weather, where CR exceeds 200 at an error bound of just 0.15, as depicted in Figure 2. This striking efficiency is primarily due to the dataset's characteristics, particularly, its small rIQD of 5% as summarized in Table 1. Such a characteristic allows compression to approximate the entire dataset with a small number of segments under the pointwise relative error bound, as shown in Figure 3. This contrasts sharply with the results observed on the Solar dataset, where the CR barely surpasses 14 even with an error bound of 0.8. In this case, rIQD is equal to 200%, indicating that even with an error bound of 0.8, the compression method cannot fit the entire series with only a few segments. These findings show the importance of contextualizing the error bound within the rIQD of the dataset to avoid misleadingly high CRs that can compromise the data quality.

For all error bounds less than or equal to 0.1 the lossy compression methods all have a similar TE. PMC exhibits a remarkable consistency as the error bound increases. Its strategy of only fitting the mean value across the segments seems to avoid big

Table 3: Linear regression coefficients [θ_1, θ_0], and standard error (SE). CR as a function of TE.

PEBLC		PMC		SWING		SZ	
		θ_1	θ_0	θ_1	θ_0	θ_1	θ_0
ETTm1	Coef	511.7	-0.4	126.1	0.97	104.4	6.7
	SE	27.6	0.9	5.6	0.41	13.2	1.0
ETTm2	Coef	409.9	-0.1	205.1	-0.17	128.2	10.1
	SE	25.6	1.3	8.3	0.79	17.8	1.7
Solar	Coef	367.5	3.0	91.5	1.6	288.6	5.8
	SE	23.6	0.3	4.5	0.1	43.2	0.7
Weather	Coef	7493.9	-203.3	973.3	166.9	76.9	119.6
	SE	806.8	98.3	238.6	93.1	40.9	31.1
ElecDem	Coef	2.6e+4	-968.9	1.1e+4	-620.9	369.6	34.3
	SE	1.2e+4	948.8	2941.1	378.2	113.2	20.9
Wind	Coef	662.2	-3.0	150.7	3.0	71.9	9.3
	SE	58.6	3.4	5.7	0.99	9.6	1.8

TE, as indicated by its sub-linear growth on all datasets as shown in Figure 2. In contrast, SWING's super-linear TE growth shown in Figure 2 can be due to larger errors in particular segments—especially in segments where the data does not follow a linear trend—skewing the overall NRMSE higher. Similar intuition can be applied to SZ which has more flexibility and can take full advantage of the maximum allowed error bound.

4.2.1 Quantifying the TE and CR Relationship: To quantify the expected increase of CR per unit of TE we use a linear regression model of the form: $CR = \theta_1 TE + \theta_0$. The slope θ_1 represents the increase in CR for each unit increase in TE. The intercept θ_0 indicates the expected CR when no compression is applied. Table 3 shows the coefficients and their standard error (SE). The results can be classified into two clusters: one composed of ETTm1, ETTm2, Solar, and Wind—less sensitive to compression—and another cluster with Weather and ElecDem—highly sensitive to compression. As previously analyzed for the CR on Weather, the main reason for this is the rIQD. Specifically, the first 4 time series have an rIQD between 75% and 200%, while the other two datasets have an rIQD of 5% and 28%, see Table 1. This means that for the first 4 datasets, most of the error bounds are below this value. In summary, the results suggest that if the error bounds are lower than the rIQD—the first cluster—users can expect an increase in CR of [3.6x-6.6x], [0.9x-2.1x], and [0.7x-2.8x] for every 0.01 increase of TE for PMC, SWING, and SZ, respectively. These results confirm that PMC generally offers the highest CR gains as TE increases. However, as illustrated by the θ_0 coefficients, SZ will generally obtain a higher CR for lower error bounds. The low SEs obtained for the coefficients on ETTm1, ETTm2, Solar, and Wind indicate a strong and reliable linear relationship. In contrast, for the other two datasets (Weather and ElecDem), with high SEs, the relation is unreliable, underscoring how the rIQD can impact the compression.

Finally, we answer RQ1 through its sub-questions as follows:

RQ1.1: *Which lossy compression method best preserves the data quality as the error bound increases?* **Answer:** For error bounds less or equal to 0.1, all lossy compression methods perform similarly in terms of TE. However, as the error bound increases, PMC best preserves the data quality of the time series.

RQ1.2: *Which lossy compression method provides the highest CR?* **Answer:** SZ provides the highest CR at low error bounds. As the error bound increases, the CR provided by SZ increases slower than PMC. SWING provides the lowest CR and in some cases, even a lower CR than GORILLA.

RQ1.3: *How are TE and CR related?* **Answer:** Overall, TE and CR have a positive linear correlation on datasets where the rIQD is higher than the error bound specified by the users. In contrast, on datasets with a low rIQD, the lossy compression methods can achieve very high TE and CR at very low error bounds making the relationship less predictable.

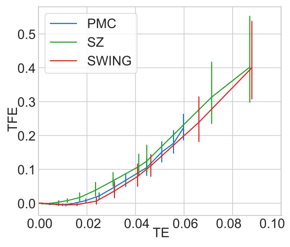
4.3 TFE Evaluation (RQ2)

To evaluate how lossy compression affects the forecasting accuracy, we first analyze the TE and TFE relation. Figure 4 shows the results. The horizontal line shows the average TFE while the vertical bars represent the 95% confidence intervals given by the different forecasting models. The smaller the vertical bars, the closer the results are across the different forecasting models. We have focused the results in the area with TFE <= 0.5, i.e., 50% of forecasting accuracy lost, to improve the visualization. We exclude GRU on Solar and ElecDem due to its significantly poorer baseline performance (Table 2) on these datasets, which renders a noticeable deviation from the typical pattern seen for the other forecasting models. Also, GRU remains largely immune to the effects of compression and even improves the forecasting accuracy as TE increases. This stark contrast from the trends on other datasets and forecasting models can skew interpretations, thus we exclude these results.

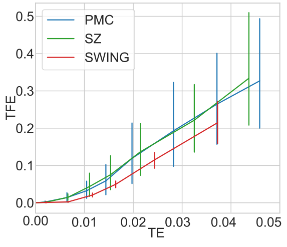
Overall, it is evident that minor TEs do not detrimentally influence forecasting accuracy. Interestingly, compression seems to be advantageous in several instances, suggesting that it helps eliminate noise or redundant information, thus enhancing the model's predictive performance. Specifically, for ETTm1, Solar, Weather, and Wind, we observe an improvement of the forecasting accuracy of up to 2% as indicated by the negative TFE. Moreover, every dataset shows a super-linear growth of TFE as TE increases, meaning that TFE decreases faster as TE increases. When evaluating the different compression methods, it becomes evident that SWING and PMC generally have a lower or equal TFE compared to SZ. This suggests that these two lossy compression methods preserve crucial data characteristics more effectively than SZ, thus leading to more consistent forecasting results despite decompression errors.

4.3.1 Time Series Characteristics Analysis. The variance in results across datasets, as depicted in Figure 4, suggests that distinct time series characteristics influence the compression methods. To explore this, we analyze 42 characteristics extracted using the R ts-feature package [23]. This analysis involves computing these characteristics on decompressed data across all lossy compression and error bounds and then computing the difference from the original data. To predict the TFE, we train a GBoost model and determine the significance of each characteristic using SHAP (SHapley Additive exPlanations) [36]. The model achieved an $R^2 = 0.9$ indicating a good fit of the train data. SHAP ranks the characteristics based on their contribution to the model's predictions. Figure 5 shows the most influential characteristics and their relationships with the predicted outcome based on SHAP values. We complement this analysis with the top characteristics based on the Spearman correlation to TFE in Table 4.

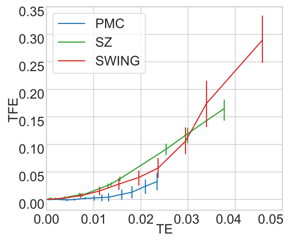
The SHAP analysis reveals that the most crucial characteristics to forecasting accuracy post-compression are those related to shifts in distribution (*max_kl_shift*, *max_level_shift*, *max_var_shift*, *mean*), autocorrelation (*seas_acf1*, *x_pacf5*), stationarity (*unitroot_pp*, *unitroot_kpss*), and seasonality (*seas_strength*). Notably, the *max_kl_shift* characteristic, representing the Kullback-Leibler



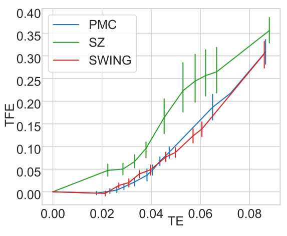
(a) ETTm1



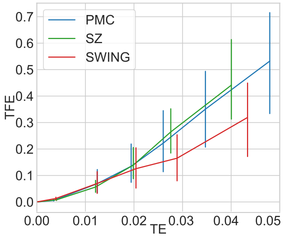
(b) ETTm2



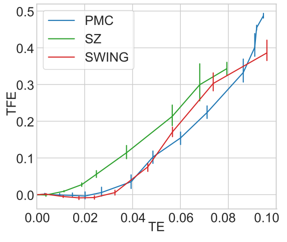
(c) Solar excluding GRU



(d) Weather



(e) ElecDem excluding GRU



(f) Wind

Figure 4: TE and TFE relation per dataset.

(KL) divergence across consecutive windows, stands out as the predominant characteristic. The Spearman correlation values in Table 4 support these results, with the KL divergence showing the highest positive correlation of 0.74. The overlap observed in both analyses, particularly for characteristics like *max_kl_shift* and *seas_strength*, underscores their robust influence on TFE. Other significant characteristics such as *diff1_acf1* and *diff2x_pacf5* from the Spearman analysis are likely related to *seas_acf1* and *x_pacf5* from the SHAP analysis, as they all capture temporal dependencies and cyclic patterns in the time series. Interestingly, the stationarity characteristics (*unitroot_pp* and *unitroot_kpss*) exhibit very low correlations to TFE, at 0.01 and 0.12 respectively, thus they are not shown in Table 4. This discrepancy in their perceived importance is due to SHAP's capability to capture complex interactions between characteristics that the correlation cannot.

By analyzing the SHAP values in the bee swarm plot in Figure 5 (left), one can determine that instances with higher KL divergence correspond to higher predicted TFE, denoted by the aggregation of red points on the positive side of the SHAP values. Conversely, instances featuring lower KL divergence correlate with improved forecasting accuracy, as depicted by blue points on the negative SHAP values. Similarly, higher shifts in mean and variance across consecutive windows, as indicated by *max_level_shift* and *max_var_shift*, are associated with higher predicted TFE. Conversely, lower shifts in these characteristics align with a reduction in TFE. This pattern suggests that when compression mitigates the distributional shifts, it effectively acts as a smoother, thereby preserving and potentially enhancing the accuracy of subsequent forecasts, a phenomenon substantiated by the trends observed in Figure 4. These results suggest that a lossy compression method should minimize distributional shifts of consecutive windows to retain forecasting accuracy.

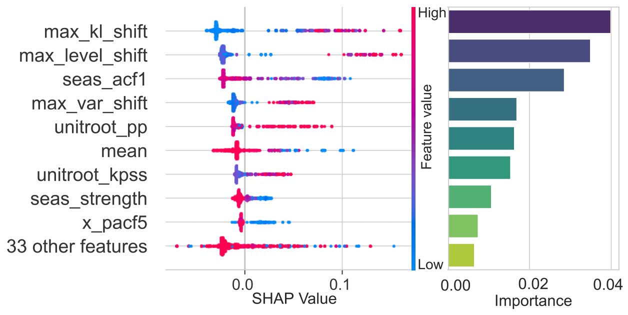


Figure 5: Top characteristics based on SHAP values.

Table 4: Top characteristics based on the correlation to TFE.

Characteristic	Correlation	Characteristic	Correlation
max_kl_shift	0.74	var	-0.40
seas_strength	-0.58	e_acf1	-0.38
flat_spots	0.57	beta	-0.37
diff1_acf1	-0.55	mean	-0.36
diff2x_pacf5	0.46	crossing_points	-0.34

In contrast, lower values of *seas_acf1* (seasonality autocorrelation at lag 1), *mean*, *x_pacf5* (partial autocorrelation at lag 5) and *seas_strength* (seasonal strength) correspond to a decrease in forecasting accuracy, as seen by blue dots with positive SHAP. This correlates with the negative correlation obtained for *seas_strength* and *mean* in Table 4. The compact clustering of SHAP values for *seas_strength* and *x_pacf5* suggests these characteristics have a stable impact on forecasting remaining consistent across the dataset despite compression. Essentially, the steadiness of these characteristics' SHAP values signifies the resilience of seasonal patterns and lagged correlations in the compressed data. Finally, *unitroot_pp* and *unitroot_kpss* cannot be straightforwardly interpreted as high values of these characteristics, indicative of non-stationarity, are associated with either high or low TFE, explaining the low correlation to TFE.

4.3.2 Inflection Point Analysis: One of the most interesting observations from Figure 4 is the presence of a distinct inflection point where an increase in TE begins to have a pronounced detrimental impact on forecasting accuracy. This characteristic bend or "elbow" in the curve is evident across most datasets. By extracting and analyzing the elbow points across different compression methods, we can systematically evaluate how each method impacts the forecasting accuracy relative to the decompression error. If one compression method consistently shows an elbow at a much higher TE than another, it suggests that this method produces errors that can be tolerated better before significantly affecting the forecasting accuracy. Similarly, we extract the error bounds, the CR, and the TFE at the elbow points to provide a comprehensive view of the lossy compression methods across all these performance metrics. This analysis is pivotal in determining the most efficient compression method for balancing error tolerance, forecasting accuracy, and CR, guiding users in selecting the most appropriate method for their specific needs. We use the Kneedle algorithm [48] to extract the elbow points in an automated manner. Table 5 shows the median of the performance metrics across forecasting models and the average (AVG) across the datasets.

Overall, PMC is a well-rounded option, delivering good compression and consistent results across datasets. Its slightly higher

Table 5: Elbows' median error bound (EB), TE, TFE, and CR.

		ETTm1	ETTm2	Solar	Weather	ElecDem	Wind	AVG
PMC	EB	0.2	0.1	0.3	0.02	0.07	0.15	**0.14**
	TE	0.025	0.02	0.013	0.03	0.026	0.039	**0.026**
	CR	10.7	7.7	8.0	25.8	10.1	19.6	13.65
	TFE	0.023	0.105	0.005	0.017	0.13	0.051	0.055
SWING	EB	0.15	0.07	0.28	0.015	0.04	0.1	0.11
	TE	0.024	0.016	0.022	0.031	0.016	0.033	0.023
	CR	3.5	4.8	3.5	10.1	4.5	7.0	5.56
	TFE	0.003	0.057	0.045	0.023	0.05	0.02	**0.033**
SZ	EB	0.2	0.07	0.2	0.03	0.05	0.05	0.10
	TE	0.031	0.015	0.008	0.067	0.02	0.018	**0.026**
	CR	12.0	11.3	9.7	23.5	25.1	8.2	**14.97**
	TFE	0.058	0.074	0.01	0.25	0.07	0.045	0.085

error bounds and TEs imply that PMC can deviate a bit more from the original data than SWING and SZ before significantly impacting forecasting accuracy. The TFEs at the detected elbows are relatively low across most datasets, suggesting that, for the amount of CR achieved (13.65x on average), the impact on forecast accuracy is tolerable (0.055 TFE on average). Its highest TFE is on ElecDem with 0.13, i.e., 13% of accuracy lost compared to the baseline while achieving a CR 10.1x higher than compressing with gzip. In summary, PMC offers a balanced approach, especially when there is a requirement for a moderate CR without a significant loss in forecasting accuracy.

SWING's results indicate that its compression can be the ideal choice when maintaining forecasting accuracy is the primary concern, even if it means compromising somewhat on compression. Across all datasets, SWING's TFE is consistently less than or equal to 0.05 (0.033 on average), and almost always lower than the rest of the lossy compression methods. Only on Solar, was SWING's TFE the highest at 0.045, while the rest all have less than or equal to 0.01. But even there its TE is the highest, indicating that more error was introduced before affecting the forecasting accuracy by that much.

Finally, while SZ is very efficient in terms of CR, achieving an impressive average of 14.97, it is important to note that this often comes at the cost of reduced forecasting accuracy as indicated by the higher TFEs. This trade-off is particularly noticeable on the Weather dataset with a TFE of 0.25 and a CR of 23.5. Users seeking high CRs will find SZ suitable, but only if they can tolerate some reduction in forecasting accuracy.

4.3.3 Time Series Characteristic Sensitivity. Building on our earlier analysis, we now aim to understand what is producing the elbow points analyzed. For this, knowing which characteristics start getting affected at the inflection point will provide more insights into why the forecasting accuracy starts degrading rapidly past this point. We set a threshold of TFE ≤ 0.1 as, in our findings, most datasets' elbows are below this mark. Table 6 shows the mean and standard deviation of the relative difference (%) between the original data and their compressed counterparts for the 5 most important characteristics: *max_kl_shift* (MKLS), *max_level_shift* (MLS), *seas_acf1* (SACF1), *max_var_shift* (MVS), and *unitroot_pp* (URPP).

The results reveal that while MKLS and URPP can be significantly impacted by compression, others like MLS, SACF1, and MVS remain consistently unaffected. PMC has the least average impact on these characteristics which aligns with its fundamental operation. By taking the mean value of segments, PMC inherently minimizes variations and level shifts in the compressed series,

Table 6: Mean and standard deviation of the relative difference (%) for the five most important characteristics when TFE $\leq$ 0.1.

dataset		MKLS	MLS	SACF1	MVS	URPP
ETTm1	PMC	30.0 (34)	0.3 (0.5)	0.7 (1.1)	1.3 (2.1)	12 (13.0)
	SWING	4.2 (8)	2.2 (3.2)	0.9 (1.0)	5.6 (8.4)	13 (12.8)
	SZ	35.9 (49)	0.3 (0.3)	1.1 (1.4)	1.8 (2.2)	40 (53.3)
ETTm2	PMC	81.4 (140)	0.4 (0.9)	0.6 (1.0)	1.3 (4.0)	0.6 (2.1)
	SWING	7.0 (8.3)	1.0 (1.2)	0.1 (0.2)	1.5 (2.1)	4.5 (4.3)
	SZ	63.6 (139)	0.8 (1.1)	0.5 (0.9)	1.3 (1.9)	3.9 (7.0)
Solar	PMC	60.2 (54)	0.8 (0.9)	0.3 (0.2)	0.2 (0.2)	3.6 (3.5)
	SWING	3.6 (7)	1.0 (0.9)	0.5 (0.8)	2.1 (3.4)	3.1 (4.5)
	SZ	23.0 (32)	0.2 (0.5)	1.0 (1.2)	1.8 (2.1)	6.7 (8.5)
Weather	PMC	133 (73)	1.3 (0.9)	0.4 (0.4)	2.5 (2.7)	5.2 (2.8)
	SWING	57.4 (47)	2.4 (1.3)	1.1 (1.1)	3.2 (2.3)	15.7 (8.4)
	SZ	151 (202)	2.7 (2.6)	2.0 (1.4)	4.4 (3.9)	52.6 (40)
ElecDem	PMC	49.3 (54)	0.6 (0.8)	0.7 (1.0)	0.6 (1.0)	3.4 (4.7)
	SWING	8.3 (12)	1.0 (2.5)	0.5 (1.2)	1.1 (2.7)	0.4 (0.6)
	SZ	8.4 (16)	0.5 (0.6)	0.5 (0.6)	0.9 (1.0)	1.6 (2.2)
Wind	PMC	88 (106)	0.2 (0.6)	0.3 (0.4)	1.1 (1.8)	7.4 (7.5)
	SWING	16.2 (14)	0.7 (1.0)	0.3 (0.3)	2.5 (3.5)	10 (9.4)
	SZ	28.0 (28)	0.5 (0.4)	0.2 (0.2)	1.6 (2.0)	7.8 (9.4)
AVG	PMC	73.6 (77)	**0.6 (0.8)**	**0.5 (0.7)**	**1.2 (1.9)**	**5.4 (5.6)**
	SWING	**16.1 (16)**	1.4 (1.7)	0.6 (0.8)	2.7 (3.7)	7.9 (6.7)
	SZ	51.6 (78)	0.8 (0.9)	0.9 (0.9)	1.9 (2.2)	18.8 (20)

ensuring that the fluctuations and deviations are almost negligible. Nevertheless, the remarkable stability across all datasets and compression methods of these three characteristics, indicates that the seasonality, mean, and variability of the series largely remain intact helping to preserve the TFE $\leq$ 0.1. Ultimately, when these characteristics show small deviations of even 1%, it is a sign that the forecasting models will not perform optimally, thereby making them key indicators to monitor.

Table 6 also shows that MKLS and URPP are pivotal in determining the exact impact on forecasting accuracy when TFE$\leq$0.1 for ML models. However, using these characteristics as indicators of the forecasting accuracy drop is not straightforward, particularly for MKLS. The sensitivity of the KL divergence metric to small probability values, notably in PMC's averaging approach, can exaggerate the perceived impact on forecasting accuracy. Such differences are more prominent for PMC than for SZ and SWING, leading to higher KL divergence values that do not reflect actual data value changes. On the other hand, URPP shows more uniformity across datasets, allowing users to set a threshold for alerts at even a 5% deviation, in line with PMC's average impact.

Finally, we answer RQ2 through its sub-questions as follows:

RQ2.1: *How much TE can be introduced before significantly affecting the forecasting accuracy?* **Answer:** By studying the inflection points at which the TFE starts rapidly increasing and the corresponding TE values for each combination of lossy compression and dataset, we can establish an average TE of 0.026, 0.023, and 0.026, with an average TFE of 0.055, 0.033, and 0.085, and an average error bound of 0.14, 0.11, and 0.11 for PMC, SWING, and SZ, respectively. **RQ2.2:** *Which time series characteristics best explain the impact of lossy compression on forecasting accuracy?* **Answer:** Our analysis reveals that the most important characteristics for predicting forecasting accuracy post-compression are those related to shifts in distribution (*max_kl_shift*, *max_level_shift*, *max_var_shift*, *mean*), autocorrelation (*seas_acf1*, *x_pacf5*), stationarity (*unitroot_pp*, *unitroot_kpss*), and seasonality strength

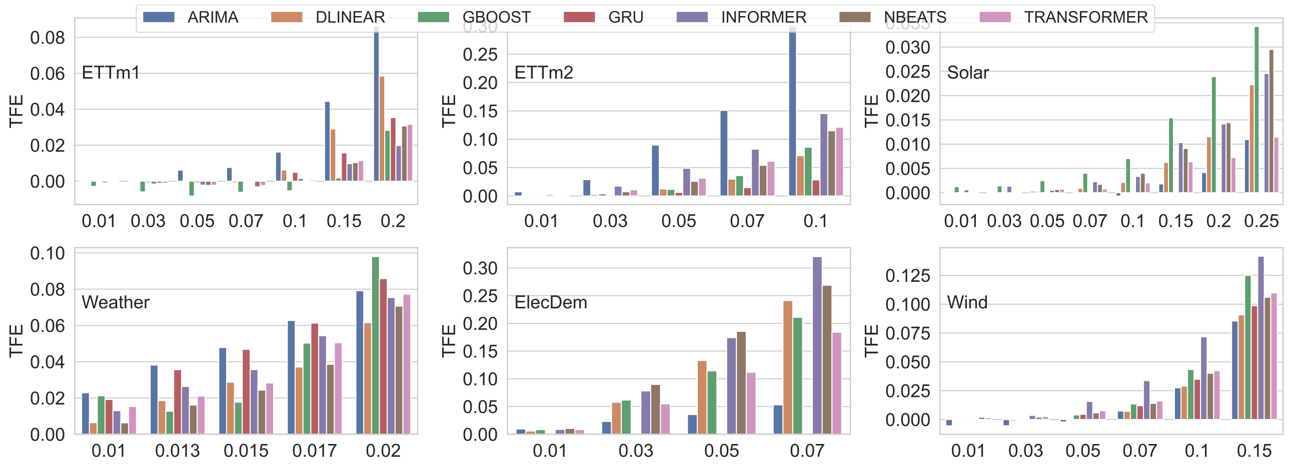


Figure 6: Average TFE per forecasting model.

Table 7: Best models based on NRMSE and TFE

	ETTm1	ETTm2	Solar	Weather	ElecDem	Wind
NRMSE	DLinear	Informer	NBeats	DLinear	Transformer	NBeats
TFE	GBoost	GRU	Arima	DLinear	Arima	Arima

(*seas_strength*). Notably, the *max_kl_shift* characteristic stands out as the predominant one. **RQ2.3:** *How high CR can be obtained without significantly affecting the forecasting accuracy?* **Answer:** We observed an average CR of 13.65 for PMC, 5.56 for SWING, and 14.97 for SZ without significantly affecting the forecasting accuracy. For specific combinations of lossy compression and dataset, the CR is 25x higher than gzip's, without significantly affecting the forecasting accuracy.

4.4 Forecasting Models Evaluation (RQ3)

To examine the resilience of the individual forecasting models, we first analyze their mean TFE at different error bounds per dataset. The maximum error bound was selected based on the mean error bound found in Table 5 per each dataset. Figure 6 shows the results. Additionally, Table 7 presents a summary of the best models based on forecasting accuracy measured by NRMSE and TFE, showing the top performers according to these metrics.

The results show that there is no single model that uniformly excels in both forecasting accuracy and compression resilience across all datasets. However, two general patterns can be extracted. 1) While complex models such as Transformers, Informer, and NBeats obtain the best NRMSE on four of the six datasets, they suffer a more significant performance drop when used on compressed data. This is evident by their absence at the top of the TFE rankings. In contrast, simpler models like Arima tend to maintain their performance better when used with lossy compressed data. 2) There is an inverse relationship between a model's baseline forecasting accuracy and its resilience to compression. Models with higher accuracy on raw data, such as Arima on ETTm2 and ETTm1, and NBeats on the Solar dataset, tend to be more sensitive to the error introduced by lossy compression. Conversely, models that exhibit lower accuracy on raw data, demonstrate stronger resistance to the errors induced by compression. For example, Informer and GRU, which produce poorer baseline forecasts on ETTm1 and ETTm2 (Table 2), demonstrate a stronger resistance to the added compression error.

These patterns have two main implications: 1) Models that achieve high accuracy on raw data, or in general complex models like Transformer, capture very subtle patterns of the time series. When the data is compressed and decompressed, those subtle patterns are among the first to be distorted, leading to a noticeable drop in accuracy for these models. Building upon the time series characteristic analysis in Section 4.3.1, the MKLS characteristic is likely indicative of these subtle patterns. 2) Models with lower accuracy on raw data, or simpler models like Arima, are capturing broader, more general patterns of the time series. These patterns are robust enough to resist the compression-decompression process, leading to relatively consistent performance. This is consistent with the SHAP values analysis of *seas_strength* and *x_pacf5*, which remains consistent across the datasets despite the use of lossy compression. This opens up interesting research directions outlined in Section 5.

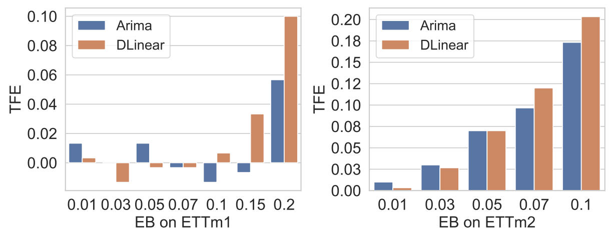


Figure 7: TFE results of Arima and DLinear training on decompressed data.

4.4.1 Training on Decompressed Data: In this section we conduct a set of experiments to understand how models respond to retraining on the decompressed data. The goal is to validate if the forecasting models can get accustomed to compression-induced error and improve their resilience by training on the decompressed data. We selected the Arima and DLinear models and the ETTm1 and ETTm2 datasets for this experiment. Arima and DLinear are the best baseline models on ETTm2 and ETTm1, respectively, as shown in Table 2. However, they also show more sensitivity to compression on those datasets. For our experiment, we train and infer in the decompressed data and record the accuracy using NRMSE between the predictions and the raw data. Figure 7 shows the TFE results per error bound.

The results show that Arima improved its resilience, this is more evident on ETTm2 where TFE was reduced from 0.3 in Figure 6 to 0.18 at the error bound of 0.2. Conversely, DLinear exhibits more sensitivity when training on decompressed data. While the difference on ETTm1 is marginal, ETTm2 showcases a substantial deterioration, with TFE increasing from 0.06 in Figure 6 to 0.2 at the error bound of 0.1. The decrease in resilience for DLinear suggests that compression alters key time series characteristics relevant to its operation. Specifically, this model decomposes the time series into trend and remainder components and learns to predict their future values. Then, when the model is trained on decompressed data, it can learn a different set of decompositions that are too distant from the original components, thus negatively impacting the DLinear model.

We explore this hypothesis by analyzing the impact on the trend and remainder components of ETTm1 and ETTm2. Specifically, we extract the values of these components on the original and decompressed data. We set the error bounds to 0.2 and 0.1 and compute the average RMSE between extracted components across all lossy compression methods. The RMSEs for the trend on ETTm1 and ETTm2 are 0.28 and 0.45, while the RMSEs for the remainder are 0.4 and 0.8, respectively. Note, that the RMSE for the remainder of ETTm2 is twice as high as the one from ETTm1 which is consistent with the observed impact on forecasting with TFEs 0.2 and 0.1, respectively. Moreover, the higher RMSEs for the remainder compared to the trend indicate that the compression is affecting short-term fluctuations more than the overall trend in the data. This aligns with Arima's higher resilience since the primary strength of this model lies in it capturing autocorrelations and trends, rather than predicting precise values in high-noise scenarios. Thus, even if some short-term fluctuations have been altered due to compression, Arima can still provide reasonably accurate forecasts based on that overall trend. Conversely, since DLinear is trained to capture both trend and short-term fluctuations, the distortion in the remainder is leading DLinear to make inaccurate predictions.

While we can discern the impacts of trend and remainder distortions on simpler models like Arima and DLinear, making similar deductions for their black-box counterparts like GRU and Transformer-based models is much more complex. The previous analysis suggested that the *max_kl_shift* can be the most significant characteristic when monitoring the performance of these models. However, validating this hypothesis would require diving into several elements, including, but not limited to, internal neuron activations, weight adjustments, gate behaviors, and attention mechanisms. This necessitates a more comprehensive, and potentially, different set of analytical techniques and simulations. We will address this question in future work and provide a general methodology that can be used for different analytics.

Finally, we answer RQ3 through its sub-questions as follows:

RQ3.1: *Which factors influence the resilience of the forecasting models?* **Answer:** Our results suggest that there are two main factors: 1) the model complexity and 2) the accuracy of the model on the raw data. Specifically, a simple model, that prioritizes the generalization of long-term trends will be more resilient than a complex model that prioritizes learning short-term fluctuations.

RQ3.2: *Which time series forecasting model is more resilient to the effect of lossy compression?* **Answer:** Arima stands out as a consistently resilient model, leading the TFE ranking in three out of the six datasets (Solar, ElecDem, and Wind). Arima also exhibits superior resilience when trained on decompressed data.

5 RESEARCH DIRECTIONS

This section outlines broader research directions in the field, intended for the wider scientific community to explore. Within this context, a critical area for advancement is the development of new lossy compression methods, specifically optimized to preserve forecasting accuracy. By leveraging the identified key characteristics like KL divergence and seasonal strength, these methods should aim to balance the CR with the preservation of forecasting accuracy. Further studies are also needed for different types of time series analytics, e.g., anomaly detection. For each of these analytics, it is crucial to identify and preserve characteristics that significantly influence them, thereby guiding the refinement of lossy compression methods to preserve these characteristics

An additional promising research direction is the development of ML models designed to predict the impact of lossy time series compression on various analytical tasks. Such models would effectively be learning the relationships between compression characteristics (e.g., compression ratio, impact on the time series characteristics) and the performance metrics of downstream analytics tasks (e.g., accuracy, precision). By doing so, they can guide the selection or optimization of compression methods based on the expected impact on analytical outcomes.

Finally, another interesting research direction involves combining models that are strong in forecasting with those that are more resilient. For example, create an ensemble model using Transformer which has good overall forecasting accuracy and Arima which is more resilient. This should improve the resilience and overall accuracy of forecasting models.

6 RELATED WORK

6.1 Time Series Forecasting

The field of time series forecasting has evolved from simple statistical models to sophisticated deep learning architectures. The development of the Arima model [5] marked a significant advancement by combining autoregressive (AR) and moving average (MA) components and handling non-stationary data (integrated, I). With the rise of machine learning, ensemble techniques like bagging and boosting [24], combining multiple models to improve forecast accuracy, offered a way to capture non-linear relationships without explicit model specification. The deep learning era has brought countless new models from Multilayer Perceptron (MLP) [46], and Radial Basis Function Neural Networks [6] which are very effective in capturing nonlinear relationships in the time series, Bayesian Neural Network which are particularly useful for estimating uncertainty and managing overfitting [37], to Transformer models [57] which are highly efficient capturing long-term dependencies in the time series [59].

Modern deep learning models can be divided into three main groups [54]. The first group consists of models that have extensively explored, combined, and modified *Recurrent and Convolutional Neural Networks* (RNNs and CNNs) [12, 32, 51, 52, 60, 64]. These models are capable of modeling short-term local dependencies among different time series. However, RNN models are not parallelizable and unable to handle long-term dependencies between the current and previous timestamps of a time series [50]. CNN models are limited by the reception field of the kernel and the locality of the features extracted.

The second group consists of *Transformer-based* [57] models adapted from NLP to learn long-term dependencies within time series. Transformer models rely on self-attention mechanisms to extract the correlations between paired elements in the time series making it permutation-invariant and anti-ordering. Examples of these models are Informer [65], Autoformer [59], Triformer [9], and FEDformer [66]. These models incrementally improved the accuracy of many datasets. However, recent papers suggest that these results have little to do with the temporal correlation learning capabilities of the Transformer model and that simpler linear models can obtain similar accuracy [62, 63].

The third group of models is based on the classic MLP architecture. This shift from Transformer models to MLP-based models is partly motivated by NBeats [42]. NBeats is the first pure MLP-based method for univariate time series forecasting that achieved state-of-the-art accuracy in the M4 competition [38]. More recently, MLP models like DLinear [62] and LightTS [63] support the claim that carefully designed MLP models are better at capturing the historical patterns of time series than Transformer models.

6.2 Time Series Compression

Time series compression algorithms can be divided into four categories based on their approach [8]: dictionary-based, functional approximation, sequential algorithms, and autoencoders.

The *dictionary-based* approach follows the principle that the time series can be represented as a sequence of segments from a dictionary. These segments are characterized by a sequence of "Atoms" which have a key in a dictionary used both in the representation of the time series and for querying its content. The dictionary can be created by domain experts who add typical patterns or by learning a training set. However, newly arriving data can introduce patterns or variations that were not previously included in the dictionary, thus constant updates of the dictionary are needed in real-life scenarios. Examples are: TRISTAN [40], CORAD [30], and A-LZSS [44].

The *functional approximation* approach divides the time series into segments and applies a function to approximate each of them. The segment size can be fixed beforehand or found by the algorithm as it compresses the data. For each segment, the compression algorithm learns the coefficients of a family of functions that best approximate the values of the segment. The functional approximation approach is especially suitable for error-bounded lossy compression as constraints on the coefficients of the functions are simple to add. Examples of functional approximation methods include PMC [33], Swing [11], and PPA [10].

The *sequential* approach combines several simple compression techniques sequentially. The most common encodings are Huffman, delta, run-length, and Fibonacci binary encoding [8]. The majority of the algorithms in this category are lossless like Spritz [4], GORILLA [43], and Chimp [34]. GORILLA is currently used as the default encoding for floating point measurements in many time series databases [26].

The *autoencoder* approach uses a symmetric pair of encoder and decoder neural network architectures. The basic idea is to reproduce the input time series in the output of the decoder by minimizing a distance metric. The output of the encoder, i.e., the embedded representation of the input, is the compressed representation. Examples of methods using this approach are DZip [16] and APRA [22].

6.3 Lossy Compression & Time Series Analytics

A few papers have made preliminary analyses of the impact of lossy compression on time series analytics including, change-detection, and time series classification. In [20] the impact of lossy compression on time series change-detection was analyzed. Specifically, the authors investigate how multiple combinations of lossy compression and change-detection algorithms perform on different datasets. The authors show that accurate change detection is possible even on heavily compressed data. In [41] the authors present a similar study for the classification problem using three lossy compression algorithms based on Wavelet Transform [2]. The paper shows an improvement from 10% to 50% in the classification accuracy and recall for different model configurations. Finally, [10] analyzes the impact of the PPA lossy compression algorithm on a single domain-specific (energy) forecasting task. Specifically, the authors perform a small experiment with a single dataset and exponential smoothing (ES) as the forecasting model. The experiments show that when guaranteeing the data is compressed within the absolute error bound of 25 Wh the forecasting accuracy remains unaffected despite achieving a compression ratio of 3x. Although these preliminary results suggest that lossy compression may not harm time series analytics, it is clear that much more in-depth analysis is needed.

7 CONCLUSIONS AND FUTURE WORK

With the rise of data-driven decision-making using massive amounts of sensors, high-frequency time series have become a crucial tool. However, managing such a massive amount of data can be challenging, as transferring or even storing the raw time series is often infeasible. Lossy compression algorithms provide a solution to this problem. However, using these algorithms introduces new issues, e.g., understanding their effect on the accuracy of time series forecasting models. In this paper, we answered the research questions: **RQ1:** How does lossy compression affect the time series? **RQ2:** How does lossy compression affect the accuracy of time series forecasting? **RQ3:** How does lossy compression affect individual forecasting models?

Our results suggest that it is possible to obtain high CR through lossy compression without having a significant negative impact on the forecasting accuracy and even in some cases improving it. Specifically, we obtained an average compression ratio of 13.65 for PMC, 5.56 for SWING, and 14.97 for SZ with an average impact on forecasting accuracy of 5.56%, 3.3%, and 8.5%, respectively. Among the compression methods, PMC offers a well-balanced approach, especially when there is a requirement for a moderate CR without a significant loss in forecasting accuracy. Moreover, we analyzed how the impact of lossy compression on 42 different time series characteristics, can be used to predict the impact of lossy compression on forecasting accuracy. From this analysis, the maximum Kullback-Leibler divergence between consecutive windows was the top characteristic, followed by the seasonality strength and other characteristics related to the autocorrelation function. Finally, Arima stood out as the most resilient model in half of the tested datasets and exhibited superior resilience when trained on decompressed data.

In future work, we will use synthetic data to further validate our findings. This approach will allow us to adjust the critical time series characteristics identified in this paper, and test the resilience of specific forecasting models to changes in these characteristics. Moreover, we will explore scenarios that include training forecasting models on decompressed data.

ACKNOWLEDGMENTS

This paper was supported by the MORE project funded by the EU Horizon 2020 program under grant agreement no. 957345.

REFERENCES

[1] Nasir Ahmed, T. Raj Natarajan, and K. R. Rao. 1974. Discrete Cosine Transform. *IEEE Trans. Computers* 23, 1 (1974), 90–93. https://doi.org/10.1109/T-C.1974.

223784

[2] Ali N Akansu and Richard A Haddad. 2001. *Multiresolution signal decomposition: transforms, subbands, and wavelets.* Academic Press, Inc. https://shop.elsevier.com/books/multiresolution-signal-decomposition/akansu/978-0-12-047141-6

[3] J.Scott Armstrong and Fred Collopy. 1992. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting* 8 (1992), 69–80. Issue 1. https://doi.org/10.1016/0169-2070(92)90008-W

[4] Davis W. Blalock, Samuel Madden, and John V. Guttag. 2018. Sprintz: Time Series Compression for the Internet of Things. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3 (2018), 93:1–93:23. https://doi.org/10.1145/3264903

[5] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons. https://www.wiley.com/en-us/Time+Series+Analysis%3A+Forecasting+and+Control%2C+5th+Edition-p-9781118675021

[6] David S. Broomhead and David Lowe. 1988. Multivariable Functional Interpolation and Adaptive Networks. *Complex Syst.* 2, 3 (1988). http://www.complex-systems.com/abstracts/v02_i03_a05.html

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD.* 785–794. https://doi.org/10.1145/2939672.2939785

[8] Giacomo Chiarot and Claudio Silvestri. 2023. Time Series Compression Survey. *ACM Comput. Surv.* 55, 10 (2023), 198:1–198:32. https://doi.org/10.1145/3560814

[9] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. 2022. Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting. In *Proceedings of IJCAI.* https://doi.org/10.24963/IJCAI.2022/277

[10] Frank Eichinger, Pavel Efros, Stamatis Karnouskos, and Klemens Bohm. 2015. A time-series compression technique and its application to the smart grid. *VLDB J.* 24, 2 (2015), 193–218. https://doi.org/10.1007/S00778-014-0368-8

[11] Hazem Elmeleegy, Ahmed K. Elmagarmid, Emmanuel Cecchet, Walid G. Aref, and Willy Zwaenepoel. 2009. Online Piece-wise Linear Approximation of Numerical Streams with Precision Guarantees. *Proc. VLDB Endow.* 2, 1 (2009), 145–156. https://doi.org/10.14778/1687627.1687645

[12] Hosein Eskandari, Maryam Imani, and Mohsen Parsa Moghaddam. 2021. Convolutional and recurrent neural network based model for short-term load forecasting. *Electric Power Systems Research* 195 (2021), 107173. https://doi.org/10.1016/j.epsr.2021.107173

[13] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232. https://doi.org/10.1214/aos/1013203451

[14] James Gareth, Witten Daniela, Hastie Trevor, and Tibshirani Robert. 2013. *An introduction to statistical learning: with applications in R.* Springer. https://doi.org/10.1007/978-1-4614-7138-7

[15] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. 2021. Monash Time Series Forecasting Archive. In *NeurIPS*, Vol. 1. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/eddea82ad2755b24c4e168c5fc2ebd40-Abstract-round2.html

[16] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. 2021. DZip: improved general-purpose lossless compression based on novel neural network modeling. In *DCC.* IEEE, 153–162. https://doi.org/10.1109/DCC50243.2021.00023

[17] S Edward Hawkins III and Edward Hugo Darlington. 2012. *Algorithm for compressing time-series data.* Technical Report. NASA Tech Briefs. https://ntrs.nasa.gov/citations/20120010460

[18] Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Koscisz, Dennis Bader, Frédérick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gaël Grosch. 2022. Darts: User-Friendly Modern Machine Learning for Time Series. *J. Mach. Learn. Res.* 23 (2022), 124:1–124:6. http://jmlr.org/papers/v23/21-1177.html

[19] Akaike Hirotugo. 1974. A new look at the statistical model identification. *IEEE Trans. Automat. Control* 19, 6 (1974), 716–723. https://doi.org/10.1109/TAC.1974.1100705

[20] Gregor Hollmig, Matthias Horne, Simon Leimkühler, Frederik Schöll, Carsten Strunk, Adrian Englhardt, Pavel Efros, Erik Buchmann, and Klemens Böhm. 2017. An evaluation of combinations of lossy compression and change-detection approaches for time-series data. *Inf. Syst.* 65 (2017), 65–77. https://doi.org/10.1016/J.IS.2016.11.001

[21] Tao Hong. 2020. Forecasting with high frequency data: M4 competition and beyond. *International Journal of Forecasting* (2020). https://doi.org/10.1016/j.ijforecast.2019.03.013

[22] Daniel Hsu. 2017. Time Series Compression Based on Adaptive Piecewise Recurrent Autoencoder. *CoRR* abs/1707.07961 (2017). http://arxiv.org/abs/1707.07961

[23] Rob Hyndman, Yanfei Kang, Pablo Montero-Manso, Mitchell O'Hara-Wild, Thiyanga Talagala, Earo Wang, and Yangzhuoran Yang. 2023. *tsfeatures: Time Series Feature Extraction.* https://pkg.robjhyndman.com/tsfeatures/

[24] Rob J Hyndman and George Athanasopoulos. 2021. *Forecasting: principles and practice, 3rd edition.* OTexts. https://otexts.com/fpp3/

[25] Robert F. Engle Jeffrey R. Russell. 2010. Analysis of high-frequency Data, In Handbooks in Finance, Handbook of Financial Econometrics: Tools and Techniques. Vol. 1. Elsevier, 383–426. https://doi.org/10.1016/B978-0-444-50897-3.50010-9

[26] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. [n.d.]. Time Series Management Systems: A 2022 Survey. In *Data Series Management and Analytics (Forthcoming).* ACM. Preprint available at: https://vbn.aau.dk/da/publications/time-series-management-systems-a-2022-survey.

[27] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. 2018. ModelarDB: Modular Model-Based Time Series Management with Spark and Cassandra. *Proc. VLDB Endow.* 11, 11, 1688–1701. https://doi.org/10.14778/3236187.3236215

[28] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. 2021. Scalable Model-Based Management of Correlated Dimensional Time Series in ModelarDB+. In *ICDE.* IEEE, 1380–1391. https://doi.org/10.1109/ICDE51399.2021.00123

[29] Søren Kejser Jensen, Christian Thomsen, and Torben Bach Pedersen. 2023. *ModelarDB: Integrated Model-Based Management of Time Series from Edge to Cloud.* Vol. 53. 1–33. https://doi.org/10.1007/978-3-662-66863-4_1

[30] Abdelouahab Khelifati, Mourad Khayati, and Philippe Cudré-Mauroux. 2019. Corad: Correlation-aware compression of massive time series using sparse dictionary coding. In *Big Data.* IEEE, 2289–2298. https://doi.org/10.1109/BIGDATA47090.2019.9005580

[31] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR.* http://arxiv.org/abs/1412.6980

[32] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *ACM SIGIR.* ACM, 95–104. https://doi.org/10.1145/3209978.3210006

[33] Iosif Lazaridis and Sharad Mehrotra. 2003. Capturing Sensor-Generated Time Series with Quality Guarantees. In *ICDE.* IEEE, 429–440. https://doi.org/10.1109/ICDE.2003.1260811

[34] Panagiotis Liakos, Katia Papakonstantinopoulou, and Yannis Kotidis. 2022. Chimp: Efficient Lossless Floating Point Compression for Time Series Databases. *Proc. VLDB Endow.* 15, 11 (2022), 3058–3070. https://doi.org/10.14778/3551793.3551852

[35] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2018. Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets. In *Big Data.* IEEE, 438–447. https://doi.org/10.1109/BIGDATA.2018.8622520

[36] Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* 2, 1 (2020), 56–67. https://doi.org/10.1038/S42256-019-0138-9

[37] Martin Magris and Alexandros Iosifidis. 2023. *Bayesian learning for neural networks: an algorithmic survey.* Vol. 56. 11773–11823 pages. https://doi.org/10.1007/S10462-023-10443-1

[38] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34, 4 (2018), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001

[39] Stéphane Mallat. 1989. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 11, 7 (1989), 674–693. https://doi.org/10.1109/34.192463

[40] Alice Marascu, Pascal Pompey, Eric Bouillet, Michael Wurst, Olivier Verscheure, Martin Grund, and Philippe Cudre-Mauroux. 2014. TRISTAN: Real-time analytics on massive time series using sparse dictionary compression. In *Big Data.* IEEE, 291–300. https://doi.org/10.1109/BIGDATA.2014.7004244

[41] Aekyeung Moon, Jaeyoung Kim, Jialing Zhang, and Seung Woo Son. 2018. Evaluating fidelity of lossy compression on spatiotemporal data from an IoT enabled smart farm. *Comput. Electron. Agric.* 154 (2018), 304–313. https://doi.org/10.1016/J.COMPAG.2018.08.045

[42] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR.* OpenReview.net. https://openreview.net/forum?id=r1ecqn4YwB

[43] Tuomas Pelkonen, Scott Franklin, Paul Cavallaro, Qi Huang, Justin Meza, Justin Teller, and Kaushik Veeraraghavan. 2015. Gorilla: A Fast, Scalable, In-Memory Time Series Database. *Proc. VLDB Endow.* 8, 12 (2015), 1816–1827. https://doi.org/10.14778/2824032.2824078

[44] James Pope, Antonis Vafeas, Atis Elsts, George Oikonomou, Robert J. Piechocki, and Ian Craddock. 2018. An accelerometer lossless compression algorithm and energy analysis for IoT devices. In *WCNC.* IEEE, 396–401. https://doi.org/10.1109/WCNCW.2018.8368985

[45] Martin Ringwelski, Christian Renner, Andreas Reinhardt, Andreas Weigel, and Volker Turau. 2012. The hitchhiker's guide to choosing the compression algorithm for your smart meter data. In *ENERGYCON.* IEEE. https://doi.org/10.1109/EnergyCon.2012.6348285

[46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536. https://doi.org/10.1038/323533a0

[47] Fathi M Salem. 2022. *Recurrent Neural Networks: From Simple to Gated Architectures.* Springer Nature. https://doi.org/10.1007/978-3-030-89929-5

[48] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a Kneedle in a Haystack: Detecting Knee Points in System Behavior. In *ICDCSW*. 166–171. https://doi.org/10.1109/ICDCSW.2011.20

[49] Maxim Vladimirovich Shcherbakov, Adriaan Brebels, Nataliya Lvovna Shcherbakova, Anton Pavlovich Tyukov, Timur Alexandrovich Janovsky, Valeriy Anatol'evich Kamaev, et al. 2013. A survey of forecast error measures. *World Applied Sciences Journal* (2013). https://doi.org/10.5829/idosi.wasj.2013.24.itmies.80032

[50] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306. https://doi.org/10.1016/j.physd.2019.132306

[51] Shun-Yao Shih, Fan-Keng Sun, and Hung-Yi Lee. 2019. Temporal pattern attention for multivariate time series forecasting. *Mach. Learn.* 108, 8-9 (2019), 1421–1441. https://doi.org/10.1007/S10994-019-05815-0

[52] Xianlun Tang, Yuyan Dai, Ting Wang, and Yingjie Chen. 2019. Short-term power load forecasting based on multi-layer bidirectional recurrent neural network. *IET Generation, Transmission & Distribution* 13, 17 (2019), 3847–3854. https://doi.org/10.1049/iet-gtd.2018.6687

[53] Seshu Tirupathi, Dhaval Salwala, Giulio Zizzo, Ambrish Rawat, Mark Purcell, Søren Kejser Jensen, Christian Thomsen, Nguyen Ho, Carlos E Muniz-Cuza, Jonas Brusokas, et al. 2022. Machine Learning Platform for Extreme Scale Computing on Compressed IoT Data. In *Big Data*. IEEE, 3179–3185. https://doi.org/10.1109/BigData55660.2022.10020540

[54] Jose F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martinez-Alvarez, and Alicia Troncoso. 2021. Deep learning for time series forecasting: a survey. *Big Data* 9, 1 (2021), 3–21. https://doi.org/10.1089/BIG.2020.0159

[55] Robert Underwood. 2022. libpressio. https://github.com/robertu94/libpressio.

[56] Kevin Michell Valencia, Werner Kristjanpoller, and Marcel C. Minutolo. 2022. Electrical consumption forecasting: a framework for high frequency data. *Neural Comput. Appl.* 34, 7 (2022), 5577–5586. https://doi.org/10.1007/S00521-021-06735-8

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[58] Chen Wang, Xiangdong Huang, Jialin Qiao, Tian Jiang, Lei Rui, Jinrui Zhang, Rong Kang, Julian Feinauer, Kevin Mcgrail, Peng Wang, Diaohan Luo, Jun Yuan, Jianmin Wang, and Jiaguang Sun. 2020. Apache IoTDB: Time-series database for Internet of Things. *Proc. VLDB Endow.* 13, 12 (2020), 2901–2904. https://doi.org/10.14778/3415478.3415504

[59] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. (2021), 22419–22430. https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html

[60] Wenyi Wu, Wenlong Liao, Jian Miao, and Guoli Du. 2019. Using gated recurrent unit network to forecast short-term load considering impact of electricity price. *Energy Procedia* 158 (2019), 3369–3374. https://doi.org/10.1016/j.egypro.2019.01.950

[61] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *ACM SIGKDD*. ACM, 753–763. https://doi.org/10.1145/3394486.3403118

[62] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting?. In *Conference on Artificial Intelligence, AAAI*. AAAI Press, 11121–11128. https://doi.org/10.1609/AAAI.V37I9.26317

[63] Tianping Zhang, Yizhuo Zhang, Wei Cao, Jiang Bian, Xiaohan Yi, Shun Zheng, and Jian Li. 2022. Less Is More: Fast Multivariate Time Series Forecasting with Light Sampling-oriented MLP Structures. *arXiv preprint arXiv:2207.01186* (2022). https://arxiv.org/pdf/2207.01186.pdf

[64] Jian Zheng, Cencen Xu, Ziang Zhang, and Xiaohua Li. 2017. Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network. In *CISS*. IEEE, 1–6. https://doi.org/10.1109/CISS.2017.7926112

[65] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI*. AAAI Press, 11106–11115. https://doi.org/10.1609/AAAI.V35I12.17325

[66] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *ICML (Proceedings of Machine Learning Research)*, Vol. 162. 27268–27286. https://proceedings.mlr.press/v162/zhou22g.html