# NORMALIZATION METHODOLOGIES FOR SHORT RANGE TIME SERIES FORECASTING IN PREDICTIVE SEQUENTIAL EVALUATION FRAMEWORKS

**Richárd Tuhári[1,*], Bálint Szilveszter Varga[1], Nóra Fenyvesi[1,2], József Stéger[1,3]**
[1] Artificial Intelligence National Laboratory,
Budapest, Hungary
[*] richardtuhari@protonmail.com
[2] Von Kármán Laboratory for Environmental Flows, Eötvös Loránd University, Budapest, Hungary
[3] Department of Physics of Complex Systems, Eötvös Loránd University, Budapest, Hungary

**ABSTRACT**

This work[1] aims to study the power and usability of various simple machine learning and deep learning models that work with time series of various data sources and attempt to make predictions. In the field of time series analysis predictive models are of utmost importance when dealing with tasks like weather forcasting, predicting measures of energy consumption, or grasp how certain economics pointers extrapolate in time. It is widespread to use traditional statistical methods like ARIMA or ETS, however machine learning (ML) and deep learning (DL) models, among them simple neural networks, e.g. LSTM, are promising alternatives. In this work, the main goal is to compare the errors and robustness of a few models and investigate how the data preprocessing may have a significant footprint on efficiency.

***K*eywords** Normalization · short range forecast · prequential evaluation frameworks

## 1 Introduction

Time series forecasting has evolved from the statistical models of the 20th century [1], through the hybrid models of the previous decade [2], to fully deep learning-based models. Although the problem is formally very easy to describe: given a variable $s$ changing over time, whose behavior we would like to predict. The actual complexity of the task strongly depends on the nature of the data. Different signal-to-noise ratios and characteristics describe weather, financial, power consumption, or even traffic data, so in many cases it is not even possible to choose a universally accurate model.

In recent years, the main focus has been on Long Time Series Forecasting (LTSF), where the target size grew an order of magnitude, from the order of tens to hundreds. This change of direction has, of course, brought with it new, previously less significant difficulties, which different models have dealt with in different ways.

One of the first, really significant models designed specifically for LTSF was N-HiTS[3], which modified the N-BEATS[4] architecture using MaxPooling and interpolation, allowing it to capture more of the long time dependencies and achieve higher accuracy, with less computational cost. Informer[5] and Reformer are transformer based models[6, 7], that focused on making self-attention sparse, reducing the $N^2$ difficulty, and the resources required to use these models. Other transformer models are Autoformer[8] and FEDformer[9], which used autocorrelation and frequency domain information in the attention mechanism. Subsequently, DLinear, an embarrassingly simple model, capable of higher accuracy than all the prior ones, proved that the success of the previous ones was not due to the attention mechanism, rather to the proper usage of the statistical methods in the first part of the models. Learning from that, SegRNN[10] and PatchTST[11] used a similar approach with two different architectures: in both cases, the main idea is to break the time series into segments or patches. For the forecast, the former uses a recurrent, the latter a transformer model.

[1] https://github.com/richardtuhari/ts-methodologies

Table 1: Study datasets and their most important descriptive quantites

| Label | sample rate | size ($N$) | Minimum | Median | Maximum | Mean | Variance |
|---|---|---|---|---|---|---|---|
| *ETTh1* | hour | 17420 | −4.1 | 11.4 | 46.0 | 13.3 | 8.7 |
| *ETTm2* | 15 mins | 69680 | −2.6 | 26.6 | 58.9 | 26.6 | 11.9 |
| *ETTh2* | hour | 17420 | −2.6 | 26.6 | 58.9 | 26.6 | 11.9 |
| *ETTm1* | 15 mins | 69680 | −4.2 | 11.4 | 46.0 | 13.3 | 8.6 |
| *weather* | 10 mins | 52696 | −6.4 | 10.2 | 34.8 | 10.8 | 7.5 |
| *ILINet* | week | 992 | 0.49 | 1.36 | 7.80 | 1.84 | 1.30 |
| *electricity* | hour | 26304 | 0.0 | 204.0 | 752.0 | 227.1 | 102.2 |
| *traffic* | hour | 17544 | 0.00 | 0.05 | 0.45 | 0.049 | 0.06 |
| *exchangerate* | day | 7588 | 0.48 | 0.76 | 1.10 | 0.77 | 0.14 |
| *AMZN* | day | 6516 | 0.07 | 6.44 | 186.57 | 31.6 | 48.06 |

Although the main focus was on creating increasingly accurate models, a factor of almost equal or even greater importance seemed to have been forgotten: methodology. Within methodology, we discuss two main issues: the choice of training samples and normalization.

The methodology presented in Autoformer[8], FEDformer[9] and several others is the following: the data is first segmented into three parts, namely train, validation, and test in a 7:1:2 ratio, and then the Z-score normalization is done for the entire dataset at once based on the train segment. Another way of normalization was to only subtract the last value of the sample. These approaches can work if the data do not contain any distribution shift, otherwise they cannot capture the changing trends. Kim et al. (2022)[12] proposed a solution for that with RevIN (Reversible Instance Normalization), which contains learnable affin parameters, that can predict distribution changes and ensure that the data remain normalized.

## Materials and methods

The aim of this work is to architect and study frameworks to be used in time series forecast. The idea is that at a given time (at present) the temporal evolution of a data source is well known, and it is represented by a sample series. The task is to predict the next few samples. After the prediction is over, the clock ticks and one gets to know the new sample and becomes aware of what the exact error has become. As the natural next step, the model can be re-trained including this new sample, and one reiterates with the forecast step. This methodology is also called the rolling predictive sequential evaluation or rolling prequential evaluation [13, 14].

A possible architecture to achieve this is shown in Figure 1.

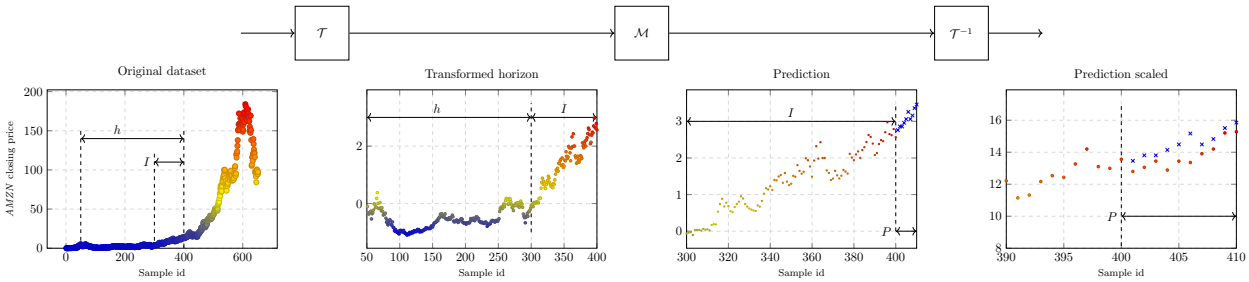


Figure 1: **Architecture.** The main components of the workflow are, the time series transformer $\mathcal{T}$, the model $\mathcal{M}$ and the inverse transformer $\mathcal{T}^{-1}$.

In our study, we rely on time series datasets that are commonly used in time series forecasting competitions. In Table 1, these time series are listed, and the most common aggregates are presented to describe them. Dataset *ETTH** [15] contains electricity transformer temperature data, *weather* is a collection of meteorological indicators collected by the Max Planck Institute, *ILINet* covers the rate of flu-like diseases throughout the US, *electricity* [16] involves electric power consumption, *traffic* holds traffic data from San Fransinsco, *exchangerate* is the evolution of daily currency rates of 8 countries, and *AMZN* contains Amazon's stock prices from 1997 until 2023.

Model performance largely depends on the statistical properties of the data, including their stationarity. To assess this, we apply both the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [17] and the Augmented Dickey-Fuller (ADF)

Table 2: Study datasets and their most important descriptive quantites

| Label | raw | smoothed | differenced |
|---|---|---|---|
| *ETT** | diff | diff | yes |
| *weather* | diff | diff | yes |
| *ILINet* | yes | diff | yes |
| *electricity* | diff | diff | yes |
| *traffic* | diff | diff | yes |
| *exchangerate* | not | not | yes |
| *AMZN* | not | not | yes |

test [18] to determine whether the raw dataset, its smoothed version, or its differenced form is stationary. The results of these tests lead to four possible interpretations:

- Both tests indicate non-stationarity $\longrightarrow$ The time series is considered non-stationary.
- Both tests indicate stationarity $\longrightarrow$ The series is regarded as stationary.
- KPSS suggests stationarity, but ADF does not $\longrightarrow$ The series is trend-stationary, meaning it becomes stationary once the trend is removed.
- KPSS indicates non-stationarity, while ADF suggests stationarity $\longrightarrow$ The differenced series has high chance to be stationary.

In Table 2. these features are collected.

Denote the studied time series as $s_i$. At time $t$ the system with memory $h$ is aware only of a slice of dataset $\mathbf{s}^{(t)} := \{s_{t+1-h}, s_{t+2-h}, \dots s_t\}$. The system memory is also called the history size in our notation. Time series $s$ may come from various sources and sample values may possibly cover a range of several orders of magnitude, a well-established preprocessing step is to standardize them. Data standardization is a necessary step in order the model can work with numbers in a range, where its output does not change hectically during the training phase with the parameter variation. In our architecture in Fig. 1 the transformer component $\mathcal{T}$ implements it. The transformer $\mathcal{T} : \mathcal{R}^h \longrightarrow \mathcal{R}^I$ works in the domain of $h$-slices and maps data to an input vector $\mathbf{x}^{(t)} := \{x_{1-I}^{(t)}, x_{2-I}^{(t)}, \dots x_0^{(t)}\}$, where $I$ is the model's input size, also called the horizon, and $x_0^{(t)}$ is the currrent *today* in our simulation.

In our problem statement, the model forecasts only a few $P$ target points $\{\hat{\mathbf{y}}^{(t)}\} := \{\hat{s}_{t+1}, \hat{s}_{t+2}, \dots \hat{s}_{t+P}\}$. The models used in our comparisons are initialized from a random state. The assumption that there are enough samples available $t > h$ in our cases always hold.

During the simulation, in each round, the model predicts the next $P$ points $\mathbf{y}^{(i)} := \{s_{i+1}, s_{i+2}, \dots s_{i+P}\}$ for every $S$ slice in the $h$ memory window with $I$ input size, and then is trained further to minimize the loss function. From now on, this collection of $S$ slices for a given $h$ memory will be referred to as $\{\mathbf{s}^{(i)}\}$ with $i \in [t-h, t]$ and used in our rolling prequential evaluation framework variants. Post-history loss values are collected to evaluate the accuracy of the methods examined. The goal of the methodology is to simulate the way a forecasting model is being used in a real-life application.

It is important to note that in our architecture a model prediction is transformed ($\mathcal{T}^{-1}$) back to the original space of the samples, whereas in many comparative studies, the loss calculations remain in the transformed domain.

Another key aspect of the methodology is the question of validation. Schnaubelt (2019) [19] measured the accuracy of several validation techniques shown in Figure 2 on generated data. The results indicate that the effectiveness of the validation techniques varies depending on the type of data-generating processes (DGP) used in the simulations. For globally stationary data, cross-validation methods performed well, however, as the data evolved over time, these methods struggled, leading to increased prediction errors. Specifically, forward-validation techniques, such as rolling-origin (roFV) and growing-window (gwFV), demonstrated superior performance in scenarios with non-stationary data, effectively adapting to the changing dynamics and yielding more accurate out-of-sample predictions.

In one of our frameworks, we employ a different validation scheme by integrating *k*-fold cross-validation into the learning process. We define a system memory (or history size) $h$, aligned with a timestep. At the beginning of each timestep, we load the best-performing model from the previous *k*-folds. Using the current memory, we then apply walk-forward cross-validation with predefined input size $I$ and target size $P$, sliding this window through the history.

Since cross-validation produces multiple models per timestep, we retain only the best-performing model for the next timestep, allowing it to be further trained—effectively acting as an in-training validation mechanism.

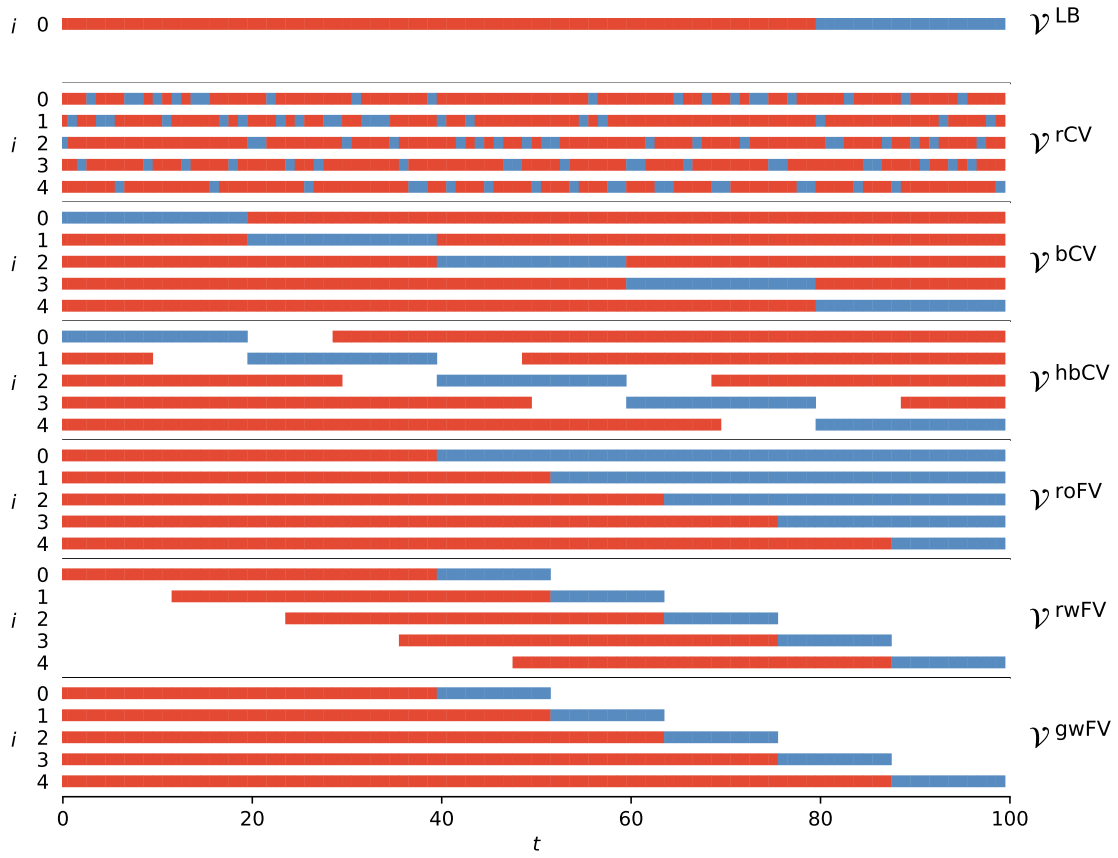


Figure 2: **Visualization of the examined validation methods.** The red parts of the data are used for training, the blue ones for validation. The methods shown are the following: $\mathcal{V}^{\text{LB}}$: last-block validation, $\mathcal{V}^{\text{rCV}}$: random cross-validation, $\mathcal{V}^{\text{bCV}}$: blocked cross-validation, $\mathcal{V}^{\text{hbCV}}$: h-blocked cross-validation, $\mathcal{V}^{\text{roFV}}$: rolling-origin forward-validation, $\mathcal{V}^{\text{rwFV}}$: rolling-window forward-validation. $\mathcal{V}^{\text{gwFV}}$: growing-window forward-validation.

To quantify how well a model performs on the given time series, the residuals are collected throughout the simulation. Both MSE and SMAPE losses were investigated, but here we present our findings based on the SMAPE between $\mathbf{y}^{(i)}$ and $\bar{\mathbf{y}}^{(i)}$. A simulation based on an dataset of size $N$ delivers an ensemble of $N-h$ error values, for which the cumulative error probability distributions are evaluated to help comparison tasks.

It is claimed that the preprocessing procedure may have a significant footprint on the overall model performance, thus a multitude of $\mathcal{T}$ and $\mathcal{T}^{-1}$ transformation operations are investigated.

A typical whitening operation is the Z-score transformation $\mathbf{Z}$. In this case the empirical mean and the standard deviation of $\mathbf{s}^{(t)}$ are calculated to shift and rescale its values. Mean being $\mu_t^{(h)} = 1/h \sum_j s_{t+j-h}$ and variance being $\sigma_t^{2\,(h)} = 1/h \sum_j \left(s_{t+j-h} - \mu_t^{(h)}\right)^2$ rescale the samples as $x_{j-h}^{(t)} := (s_{t+j-h} - \mu_t^{(h)})/\sigma_t^{(h)}$. Inspired by [12] an affine transformation with learnable parameters allow for further shifting and scaling $x^{(t)}$, denote this operation as $\mathbf{S}$. In this citation implementationwise the $\mathcal{T} = \mathbf{Z} \circ \mathbf{S}$ transformation is present. Choosing and teaching $\mathbf{Z} \circ \mathbf{S}$ is equivalent with teaching simply $\mathbf{S}$, however this latter choice converges poorer, and if teaching procedure quits after a prescribed number of epochs differences will show up. We also explore nonlinear transformations. Since Z-score values have a unit standard deviation, we prefer a nonlinear function that remains close to the identity function near zero within the unit interval. Given this criterion, $f = \tanh$ a natural choice.

Having these three operations and their combinations $\mathcal{T}$ could be one of $\mathbf{Z}$, $\mathbf{S}$, $\mathbf{Z} \circ \mathbf{S}$, $\mathbf{Z} \circ f$, $\mathbf{S} \circ f$ or $\mathbf{Z} \circ f \circ \mathbf{S}$.

The mapping of the model prediction values back into the original time sequence space by $\mathcal{T}^{-1}$ is defined by the chosen counterpart $\mathcal{T}$ that combines the inverse operations in a reverse order. It is important to note $\mathcal{T}^{-1}$ is a pseudo inverse if $f$ is used, i.e. when input values lie out of the $[-1, 1]$ range, then first they are shifted to $\epsilon - 1$ or $1 - \epsilon$ to avoid singularities.

When a $\mathcal{T}$ operator has learnable parameters, two cases were differentiated. In the one case $\mathcal{T}$ and $\mathcal{T}^{-1}$ share common parameters and during gradient calculations both operators contribute. In the other case, parameters are decoupled, only the forward transformation contributes during backpropagation, and the inverse transformation uses a value wise copy of $\mathcal{T}$'s learnable parameters. A decoupled version is marked by prime, e.g. $\mathbf{S}'$.

In previous studies, it is a common practice to involve the whole dataset in the standardization step, which implies target data points are transformed too. This naturally means that correlations will occur with the target values in the train set, and the model efficiency evaluated in such a scenario is biased. It leads to a source of possible artifacts, as in a real-life scenario future data points inclucing the target values are unknown in advance, so their contributions should not be taken into account if standardization parameters depend on the samples themselves. We argue that in such a scenario, a model will perform better, and it will have a footprint in the error distributions. To showcase this effect, a special version of $\mathcal{T}$ is also implemented and probed that uses samples from both the $h$-long history and the $P$-long future to calculate parameters. These operators are denoted by a subscript, e.g. $\mathbf{S}_{\text{trg}}$.

Besides this multitude of possible transformation a possible different approach is sketched in section Results.

In this study, the following models are compared. *LIN* is a simple 2-layer perceptron model for benchmarking purposes. *LST* stands for the long short-term memory model, *NBeats* is a block-model, where a block calculates a forecast and a backcast, to combine them with the input, a refined learning based on differences to pass on to a next block. *Dlinear* based on a classical trend-season decoupling, linear layers are applied on both channels to combine them in an output.

## Results

At first, information leakage is investigated. In Figure 3. the various transformations are summarized for the *AMZN* dataset and the baseline model. In the left panel of the cumulative loss probability distribution is shown, i.e.

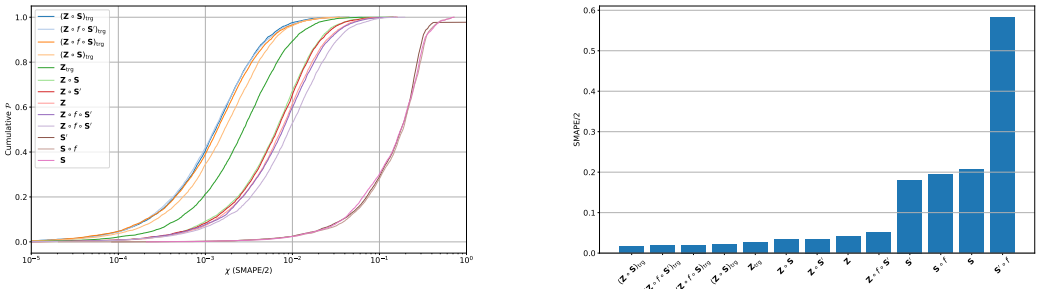


Figure 3: **Error distributions of the baseline model for various standardization schemes.** In the left the SMAPE distribution in the right extracted parameters are presented.

$\mathcal{P}(SMAPE/2 < x)$ where the division is an arbitrary choice to constraint the loss in the $[0, 1]$ closed interval. The distributions closely follow log-normal distributions, thus the choice of a logarithmic x-axis. Often a scalar is extracted from these distributions, like the median to make ordering possible. In this work we propose another approach that takes into account the log-normal behavior. A model function is fit to the data $m(\chi) = 1/2(\mathbf{erf}(\alpha\xi - \beta) + 1)$, where $\mathbf{erf}$ stand for the Gauss error function and $\chi$ is $\log_{10} x$. Parameter $\beta$ closely relates to the order of magnitude of the typical errors, and $\alpha$ accounts for the error's spread. In the right panel $10^{\beta}$ values for different $\mathcal{T}$ transformation strategies are shown in increasing order.

It is apparent those transformations $\mathcal{T}_{\text{trg}}$ that involve future unknown values in parameter estimation, tend to show smaller errors, which is expected, and what in real practice is not expected to be achieved. It is reassured that applying affine transformation after Z-score ($\mathbf{Z} \circ \mathbf{S}$) helps a model to perform better, however substituting it with a simple learnable scaler $\mathbf{S}$ performs poorer due to the limited epochs in the learning and relearning steps. Unexpected trend is that applying nonlinearity ($f$) in the transformation chain in most cases does not help. The figure also indicates that coupling the learnable parameters of $\mathbf{S}$ in most cases helps.

The effect of history size $h$ is investigated in Figure 4. Regardless of the chosen $\mathcal{T}$ a tendency appears and it turns out that the smaller the $h$ is the smaller the errors become. This is an interesting behavior, which tells us that knowing the generating distributions better in terms of mean and variance is not much of a help.

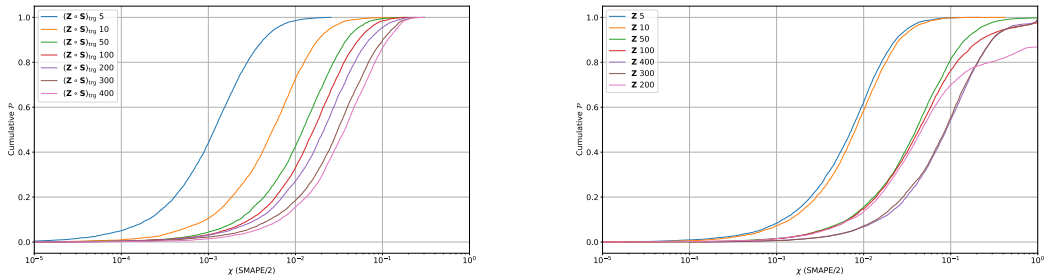


Figure 4: **The history size to choose has an impact on the model performance.** Two scenarios are depicted. The left panel shows loss distribution for traditional preprocessed Z-score, whereas the right panel is a biased transformation with learnable parameters coupled: $(\mathbf{Z} \circ \mathbf{S})_{\mathrm{trg}}$.

Summarizing the lessons of Figure 3 and 4 we get that Z-score ($\mathbf{Z}$) may not be the best natural transformation for this forecasting scheme. Applying learnable parameters will improve, but including an extra non-linearity at this stage does not necessarily help in the overall performance. For the datasets analyzed, the optimal choice is indeed $h = I$, i.e., as having a more stable approximator for the generating processes $\mu$ and $\sigma$ does not improve the model's performance.

The results of the experiment with the target bias ($\mathcal{T}_{\mathrm{trg}}$) raise the need for a solution that predicts the normalization parameters in the future in a subtle manner. When instead of $\mathcal{T} = \mathbf{Z} \circ \mathbf{S}$, latent affine transformation parameters are approximated, call them $\mu_{\mathrm{trg}}$ and $\sigma_{\mathrm{trg}}$, which could parametrize $\mathbf{S}$ better. One of our solutions for this problem is the following. First, two additional time series $\mu^{(t)}$, $\sigma^{(t)}$ are created that a model $\mathcal{S}$ learns to represent the mean and standard deviation of the current extended slice, containing both the input and the target. Here $\mathcal{S}$, in practice are two independent forecasting models to predict the mean and the standard deviation. During this training, the normalization values come only from the input slice $x^{(t)}$, and no direct information is leaked from the target. Finally, these trained models are used to generate $\mu_{\mathrm{trg}}$ and $\sigma_{\mathrm{trg}}$ to parametrize $\mathbf{S}$, consequently resulting in better forecasting accuracy. The proposed model's performance is detailed in Figure 7.

It is important to mention that the *AMZN* dataset is not stationary, see Table 2, however, its differential is. For a stationary series, it holds that the moments do not depend on the time, thus the performance of the architecture should not depend on $h$ in this case. From our numerical simulations, this hypothesis is confirmed.

In this section, we collect the lessons learned for the n-fold cross-validation framework. In Figure 5. the performance is tested for various model input sizes $I$. The big picture shows that with a smaller input size and fixed history size, the overall performance usually increases, and this holds for a wide range of data sources. This behavior is demonstrated for the baseline model *LIN* in the left panel, for the *LST* model in the middle panel and for *DLinear* model in the right panel. In the case of model *N-Beats* this ordering holds less. The dataset *traffic* has mark of constant samples at certain time which *LST* does not favor. This causes the model to coverge to a bad basin in many cases, and becomes stuck there as walking forward and this can be seen in the distribution's tendency to SMAPE/2 platoing often at 1. In the right panel, the parameters of log-normal fit is extracted to have a clearer view of the subtleties. The error bars in the figures come from the covariance matrix of least squares methods, and in the figure a ten times larger bar is used because uncertainty of this kind is small in our case. Datasets *exchangerate*, *weather* and *electricity* indicate that the optimal model input size parametrization in the range of few ten, whereas *AMZN* or *ILINet* favour smaller input sizes aroung five and seven.

In Figure 6. we revisit the information leakage problem. Errors naturally get higher if the datasets are properly preprocessed in the sense, that target samples are excluded from the calculations of the transformer parameters. For three models, the gaps between pairs of $\mathcal{T}_{\mathrm{trg}}$ and $\mathcal{T}_{\mathrm{src}}$ are shown for all datasets. The problem statement can at this point be reformulated. The distribution curve of a fairly designed architecture should lie between the extreme loss distributions. The goal then is to come up with one that tends closer to the leaked $\mathcal{T}_{\mathrm{trg}}$ curve. This gap is mostly dominated by the dataset's behavior, in our case *exchangerate*, *AMZN* and *electricity* allow larger room for model enhancement, while *ILINet* and *traffic* loss ditributions lie closer whether or not target bias is present.

In the left panel of Figure 7 the error series in the non-leaking $\mathcal{T}_{\mathrm{src}}$ case is plotted as $t$ evolves. In the same figure the preprocessor blocks output's ($\mu_{\mathrm{src}}$) of the same model, that $\mathcal{S}$ yields are plotted. Implementationwise $\mathcal{S}$ and $\mathcal{M}$ are the

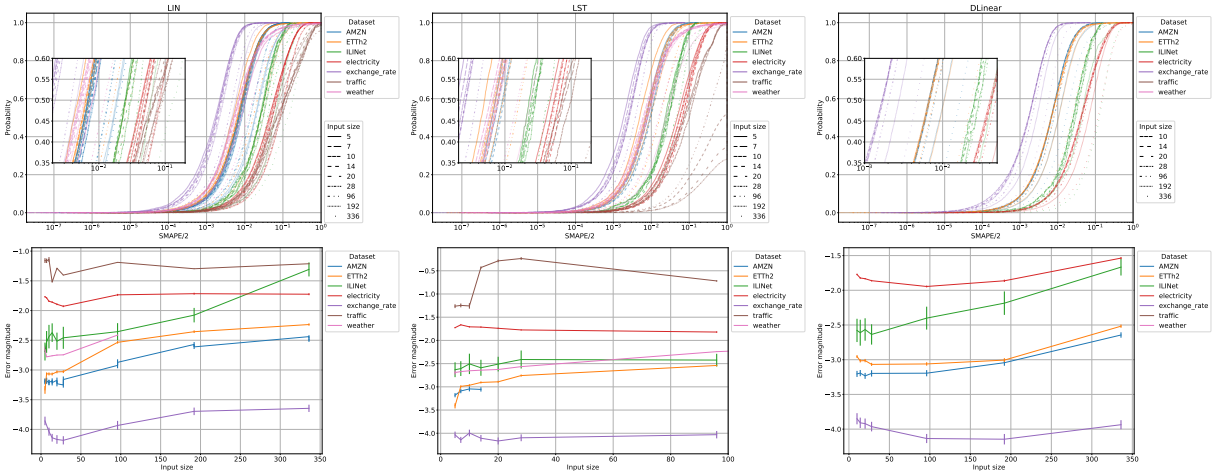


Figure 5: **Error distributions of the baseline model, the *LST* and *DLinear*.** The error distribution depends on the model input dimension $I$. For most time series analyzed, and for a large range it holds that the smaller the input size the smaller the error.

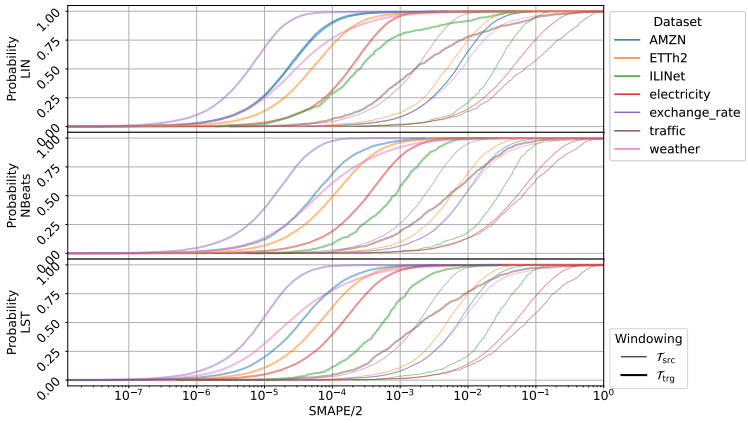


Figure 6: **Windowing effect on the error distributions.** For all timeseries and models in the experiment it is true that the windowing has impact in the performace. If the target samples are also considered during data standardization the error considerably drops.

same and the smaller fluctuation indicates the model learns this statistics more efficiently. Applying the whole chain of operations ($\mathcal{S}'$) on the samples the proposed model's error curve will nicely lie between the $\mathcal{T}_{\text{trg}}$ and $\mathcal{T}_{\text{src}}$ marginal curves. It is important to underline in this case a better performance is achieved without any information leakage. In the inset of the left figure the pairwise error values are also shown. As the model performs better, the points tend to be below the dashed diagonal.

It is worth noting, that when fitting $m(\chi)$ the error spread parameter $\alpha$ and the loss magnitude $\beta$ are not statistically independent parameters. Looking at the simulation results of Figure 5 in this aspect one can plot these parameters in a common $\alpha$-$\beta$ plane. Points well cluster together. There is a slight tendency appearing that those datasets that are not stationary and smoothing would not help to make them stationary like *AMZN* and *exchangerate* (see Table 2) tend to the bottom right part in this plane whereas differential stationary time series cluster rather in the top left direction in the $\alpha$-$\beta$ plane. This hypothesis is still investigated to confirm or reject.

## Conclusion

In time series prediction the choice of transformations has a signifficant effect in a model's output. In this paper we first verified and quantified the expected source-target-history dependency. We have simulated a multitude of datasets, transformation operators, models and frameworks to collect ensambles of loss values. The typical loss functions

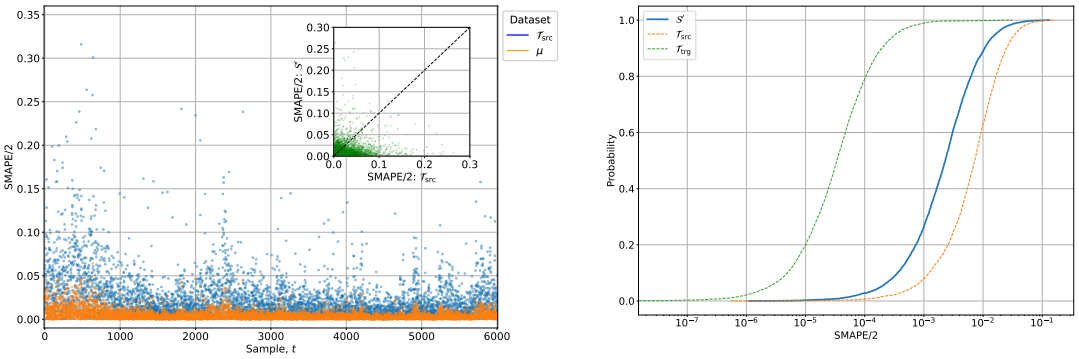


Figure 7: **The performance of the proposed model.**

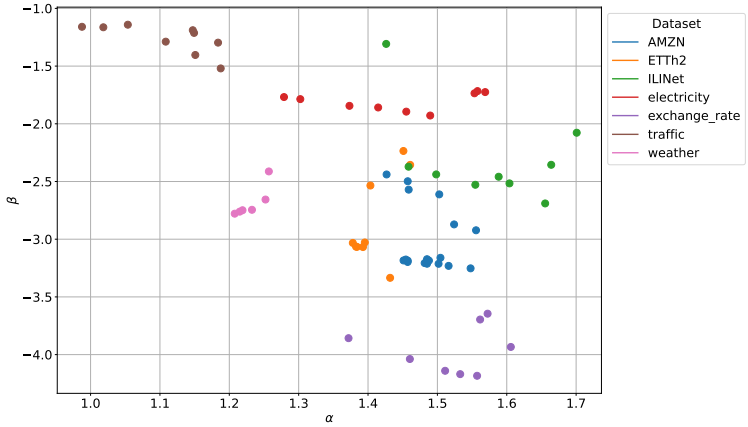


Figure 8: **The parameters of $m$ for various datasets and input sizes.**

were evaluated, including MSE and SMAPE. Framewroks included naïve walk forwarder, and in-train cross-forward validation. Information from the loss values are extracted by examining their cumulative probability distributions, which we find to well follow the log-normal distribution. Next Gaussian error functions are fit and their parameters describe a given problem in a compact manner. We identify the two extremes in the error distribution space, the lower bound defined by $\mathcal{T}_{\text{trg}}$ operation and an upper bound to find when transformation avoids the use of unseen samples. We came up with a model scheme that makes use of internal representation of standard statistical moments of sample slices to predict better. This model idea is beleived to offer a competetive solution in the problemset of short time predicion.

## Acknowledgments

This work is supported by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory.

## References

[1] G.E.P. Box and G.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day, 1970.

[2] G.Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[3] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting, 2022.

[4] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

[5] 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, 2021.

[6] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022.

[7] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2023.

[8] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[9] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, 2022.

[10] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting, 2023.

[11] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[12] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022.

[13] Juan I. González Hidalgo, Bruno IF Maciel, and Roberto SM Barros. Experimenting with prequential variations for data stream learning evaluation. *Computational Intelligence*, 35(4):670–692, 2019.

[14] David A. Bessler and Robert Ruffley. Prequential analysis of stock market returns. *Applied Economics*, 36(5):399–412, 2004.

[15] 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 *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press, 2021.

[16] Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

[17] Denis Kwiatkowski, Peter C. B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1-3):159–178, 1992.

[18] David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979.

[19] Matthias Schnaubelt. A comparison of machine learning model validation schemes for non-stationary time series data. Technical Report 11/2019, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institute for Economics, 2019.