Applied Intelligence (2024) 54:10490–10515
https://doi.org/10.1007/s10489-024-05715-4

Performance metrics for multi-step forecasting measuring win-loss, seasonal variance and forecast stability: an empirical study

Eivind Strøm[1] · Odd Erik Gundersen[1,2]

Accepted: 27 July 2024 / Published online: 20 August 2024

Abstract
This paper addresses the evaluation of multi-step point forecasting models. Currently, deep learning models for multi-step forecasting are evaluated on datasets by selecting one error metric that is aggregated across the time series and the forecast horizon. This approach hides insights that would otherwise be useful for practitioners when evaluating and selecting forecasting models. We propose four novel metrics to provide additional insights when evaluating models: 1) a win-loss metric that shows how models perform across time series in the dataset , allowing the practitioner to check whether the model is superior for all series or just a subset of series. 2) a variance weighted metric that accounts for differences in variance across the seasonal period. It can be used to evaluate models for seasonal datasets such as rush hour traffic prediction, where it is desirable to select the model that performs best during the periods of high uncertainty. 3) a delta horizon metric measuring how much models update their forecast for a period in the future over the forecast horizon. Less change to the forecast means more stability over time and is desirable for most forecasting applications. 4) decomposed errors that relate the forecasting error to trend, seasonality, and noise. Decomposing the errors allows the practitioners to identify for which components the model is making more errors and adjust the model accordingly. To show the applicability of the proposed metrics, we implement four deep learning architectures and conduct experiments on five benchmark datasets. We highlight several use cases for the proposed metrics and discuss the applicability in light of the empirical results.

Keywords Multi-step forecasting · Performance metrics · Deep learning · AI · DeepAR · TFT

1 Introduction

Time series forecasting has been a prominent research field since the early 1980s, when the *Journal of Forecasting* and the *International Journal of Forecasting* were founded. Between 1982 and 2005, over 940 papers were published, a summary of which is given by [24]. The modeling and forecasting of time series have been a key part of academic research due to its many important real-world applications, including: forecasting wind and solar power generation [41, 51], traffic [76], demand forecasting [12], trend analysis of climate change [60], data-driven medicine applications [72] and the forecasting of financial indices [14]. In most applications, it is important to produce forecasts for several time points in the future to allow decision-making based on predicted trends. *Multi-step* forecasting has historically, despite its many applications, been less studied compared to one-step forecasts [71]. Nonetheless, multi-step forecasting remains a critical part of several real-world applications and has begun to see an increase in attention, especially in the deep learning community [53].

A major contributor to advances in the field of time series forecasting has been large-scale empirical studies, designed to empirically evaluate methods and compare newly proposed models to the state-of-the-art [29]. However, despite having a long history, the problem of objectively evaluating the results of such studies remains an issue in the field. For example, the results of the M3 competition have been revisited and discussed on several occasions [20, 22, 58, 59]. [30] established early on that the ranking of the performance of various methods will vary depending on the performance

✉ Eivind Strøm
stromeivind@gmail.com

Odd Erik Gundersen
odderik@ntnu.no
https://www.ntnu.edu/employees/odderik

1 Department of Computer Science, Norwegian University of Science and Technology, Trondheim 7491, Norway

2 Aneo AI Research, Aneo AS, Trondheim 7031, Norway

metric used in the evaluation. Thus, research on developing robust and widely applicable metrics has been a prominent subject in the field [40]. Yet, [59] most recently raised concerns about the need for objective and unbiased ways to compare and test the performance of forecasting methods. Hence, the evaluation and testing of model performance is still a problem that poses the need for further research.

Despite academicians not agreeing on a single best performance metric, if such a metric exists, see e.g., [44, 45], several empirical studies have been conducted and evaluated by different performance metrics [38]. We find that most empirical studies on multi-step forecasting proceed by evaluating models on several time series or datasets and aggregate the errors across the time series and the forecast horizon, averaging some performance metrics. This approach of aggregating an error metric by averaging the results allows researchers to draw general recommendations and conclusions. However, general recommendations are of limited use during the actual evaluation to be performed by a practitioner in an industry setting. There are important questions, such as: "Does the model perform well on all datasets? How does the model perform in terms of the seasonal period? Does the model frequently update its estimates across the forecast horizon or are the forecasts relatively stable?" Such questions would be useful to answer in the evaluation and selection of models for industry applications, and we hypothesize that new metrics can provide the practitioner with additional insights.
Our main research question is: *Does evaluating forecasting models by averaging established error metrics neglect important insights on model performance, and can new metrics provide additional insights to the practitioner developing and evaluating models for industry applications?*

We extend the current literature on performance metrics for multi-step time series forecasting by proposing four novel metrics that provide the forecasting practitioner additional insights when evaluating models on a large number of datasets.

Firstly, a win-loss metric that shows how models perform across different time series in a dataset. The current approach of aggregating error metrics hides the fact that a model could be better on average on all time series, or only significantly better on a few and worse on others.

Secondly, we propose variance-weighted errors, which scale errors in relation to differences in variance across seasonal periods for highly seasonal datasets. Variance-weighted errors put more weight on correctly forecasting time periods that are difficult to forecast, for example during highly variable rush traffic periods or the time during the day when electricity consumption varies the most.

Thirdly, a delta horizon metric is proposed to measure how much multi-step forecasting models tend to update their estimates over the forecast horizon. A model could produce stable forecasts over the horizon, or update its estimates more aggressively, both of which could be favorable under different circumstances.

Lastly, we propose decomposed errors to relate forecasting errors to the decomposed time series factors: trend, seasonality, and noise. Decomposed error metrics can indicate which components of each dataset are difficult to predict far ahead in time.

In order to show the applicability of the proposed metrics, we conduct experiments on five benchmark datasets, which are commonly employed to evaluate the performance of multi-step models: electricity, traffic, solar, volatility, and wind. We compare four neural network architectures and one state space model in order to produce diverse forecasts to show the applicability of our proposed metrics. The scope of this study is not to reach conclusions about the performance of individual models, nor to obtain the best possible forecasts. Instead, we aim to showcase the applicability of our proposed metrics through experiments with datasets and models that are relevant in the most recent literature. The metrics, models, and experiments are all implemented in one open-source codebase to be fully reproducible and facilitate further research.

Our results show that important information is lost when current error metrics are averaged. The proposed win-loss metric highlights that one model will never be best for all time series, and that results could be improved by applying a combination of models. Variance-weighted errors are shown to be more robust for datasets with large variations over the seasonal period, increasing the impact of errors made during critical periods. Lastly, our delta horizon metric shows how the models differ in their magnitude of updates over the forecast horizon, even though average metrics indicate that performance over the horizon is similar.

The remainder of the paper continues by presenting a review of related work in Section 2. Section 3 presents the proposed evaluation metrics and the models that are applied in our experiments. The experimental setting is presented in Section 4, and obtained results are evaluated and discussed in Section 5. Lastly, Section 6 suggests directions for future research.

2 Literature review

Criteria for obtaining applicable and robust performance metrics have been defined and discussed in the literature: performance metrics should be reliable, valid, robust to outliers, scale-independent, and provide an informative summarization of the distribution of errors [7, 19]. Yet, performance metrics for forecasting have been highly debated throughout the existence of the field. It is well established that no single performance metric will be superior for all situations, and the identification of the "best" point forecast is highly dependent on the performance metric selected [6, 30, 57]. It is also argued that the type of metric selected is more important than the metric itself [26]. Nonetheless, several empirical studies have been conducted in attempts to compare and identify the most accurate models despite these observations [7, 38]. The most notable review on performance metrics was conducted by [40], identifying the problems of current metrics and proposing scaled errors and MASE as possible solutions.

Since [40], scaled errors have been used in several studies and forecasting competitions [42, 56, 59] and have been identified as one of the better performance metrics in terms of statistical properties [32]. Other metrics have also been proposed, such as the unscaled mean bounded relative absolute error (UMBRAE) by [17], who analyzes the proposed metric and its favorable statistical properties. However, we note that as the complexity of the metric increases (as is the case with UMBRAE), we find fewer studies applying the proposed metric.[1] Moreover, as we previously mentioned, the best point forecast will depend on the performance metric selected, because different point forecasts will minimize the expected errors for various metrics [44, 45]. Therefore, in this paper, instead of focusing on proposing or selecting the single best error metric, we propose metrics that provide additional insights to the forecast practitioner developing models in the industry.

The emphasis on conducting larger empirical studies to evaluate methods against established baselines and state-of-the-art has been fundamental to the development of new forecasting methods [29]. As such, the evaluation of forecasting models through empirical studies and forecasting competitions have been a considerable feature of the Journal of Forecasting and the International Journal of Forecasting since their inception [38]. The first large-scale empirical studies were conducted in the form of forecasting competitions, the first competition held by [55] in 1979 on a total of 1,001 time series. The competition was widely criticized for employing inappropriate error metrics [15]. Since then, several empirical studies comparing forecasting models, different methods of preprocessing and analysis of forecasting strategies have been conducted.

One of the main challenges in the field has been reducing the increasing forecasting error when forecasting for longer multi-step horizons. [11] describes five main strategies employed when producing multi-step forecasts,[2] highlighting their pros, cons, and differences in computational time. The review provides the practitioner with heuristic recommendations for what type of strategy is suitable under different circumstances and available computational resources before the actual evaluation starts. Our study aims to take this a step further, by providing the practitioner with new metrics that provide additional insights after a set of models have been chosen for evaluation. [71] extends the work of [11] by analyzing each strategy in terms of their bias-variance trade-off. Their results highlight how different factors, such as forecast horizon or time series length, impact the model bias and variance, and provide further evidence as to which strategy should be selected for different scenarios and horizons.

Multiple studies have reviewed and compared direct and indirect forecasting strategies for multi-step time forecasting based on data from the NN5 competition[4, 11].[3] The studies employed metrics such as MSE, MAPE and sMAPE to evaluate forecasting strategies, using time series from a single dataset. Although some results were presented with respect to the forecast horizon, the evaluation was mainly based on the aggregated metrics to draw general conclusions and recommendations when selecting a multi-step strategy. [8] does something similar: providing practical recommendations for the special case of hierarchical forecasting. [61] performed a large-scale empirical study, evaluating 11 methods on 95 different datasets. The results were evaluated in terms of a *multi-criteria* performance metric consisting of MSE and Theil's U coefficient, and prediction on change in direction (POCID), which, as claimed by the authors, allowed them to compare the investigated algorithms objectively [61]. The authors conclude with general recommendations and observations based on the aggregate of all results. Although general recommendations can be helpful when developing models for a new problem, they will not aid in the process of evaluation in a real-world industry setting where considerations other than average error are important. Our work provides suggestions for such tools in the form of metrics that can be applied in a variety of use cases.

Historically, neural networks have performed poorly in empirical studies, and their use for forecasting applications has been questioned on several occasions [1, 16,

[1] We have yet to find a paper employing UMBRAE as the selected performance metric.

[2] Recursive, Direct, DirRec, Multi-Input Multi-Output (MIMO), and DIRMO (combination of DIRect and miMO) are the main categories of strategies for multi-step forecasting.

[3] A paper describing the results from the NN5 competition arranged in 2008 was never published.

22]. Furthermore, forecasting competitions received few to no submissions with methods based on neural networks. Most notably, the M3 competition featured only one neural network, which nonetheless performed poorly [58]. As a follow-up, the NN3 competition was arranged using the same dataset to encourage additional submissions of neural network-based methods. None, however, outperformed the original M3 competitors [22]. [59] later compared modern machine learning algorithms with statistical methods on the original M3 dataset, again finding statistical methods to be superior. It was concluded that, with the longest time series being only 126 observations, neural networks and machine learning are simply not fit for short univariate time series. Despite these findings, deep learning for forecasting applications has gained popularity in recent years [53].

The deep learning revolution has been driven by noteworthy achievements in the fields of image processing [46], reinforcement learning [68], and natural language processing [25]. Despite having shown poor performance on short univariate time series, recent advancements have proven deep learning to be applicable for time series forecasting where complex data representations are necessary. Several architectures have been proposed for time series forecasting depending on the type of application. Sequence-to-sequence architectures based on recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) have become popular due to their ability to learn temporal patterns and long-range memory [31, 49, 79]. Temporal convolutional networks have been proposed as an alternative to RNNs, implementing long-range memory through dilated convolutions, and have been shown to outperform the canonical Seq2Seq models on several tasks [9, 23, 33]. Attention-based approaches [18, 47, 54, 63] and tensor factorization [77, 81] have been proposed for tackling multivariate time series with a large number of input series. Lastly, a recent trend is the incorporation of uncertainty estimates that have been of particular interest for sales and demand forecasting, and implemented in the form of quantile regression networks [54, 79] and hybrid models [64, 65, 67]. In this paper, we select four deep learning models based on different underlying architectures. Running experiments for our proposed metrics, we extend the current literature on evaluating deep learning models in the context of multi-step point forecasting.

The literature review highlights several gaps in the existing body of research on forecasting model evaluation. Despite extensive discussions on the criteria for robust and applicable performance metrics, the field continues to grapple with the challenge of identifying a universally superior metric due to the varying nature of forecasting scenarios. The literature also underscores the dependency of the "best" forecast on the chosen performance metric, suggesting a gap in research focused on metrics that cater to diverse forecasting goals and contexts rather than seeking a one-size-fits-all solution. Despite these challenges, deep learning has seen a resurgence in forecasting applications, driven by advancements across various domains. However, there is a notable lack of consensus on evaluating these models, particularly in the context of multi-step point forecasting. Our research aims to bridge these gaps by proposing new metrics that offer deeper insights into model performance across a spectrum of forecasting applications, thereby extending the current literature and providing valuable tools for practitioners in the field.

To summarize, we extend the current literature on performance metrics for multi-step time series forecasting by proposing four novel metrics, showing their applicability in different forecasting scenarios through an empirical deep learning study. We focus on deriving additional insights in the evaluation of forecasting models for which we have found a gap in the literature. Our results show that the proposed metrics can be applied to derive additional insights when comparing the performance of different models, showing the usefulness to practitioners and industry.

3 Methodology

In this section we present the methodology developed in order to answer our research question. Section 3.1 presents notation and terminology employed in this work. Section 3.2 presents the proposed performance metrics that provide additional insights when evaluating models. Lastly, Section 3.3 presents five multi-step forecasting models that have been selected for experiments: one state space model and four deep learning architectures. The models serve to obtain experimental results that can be analyzed with the newly proposed metrics.

3.1 Time series forecasting

The general goal of time series forecasting is to predict the target $y_{i,t}$ at time t given an entity i and prior observations $y_{i,t-k}$, where k is the lookback window. The entity i represents a logical grouping of temporal information which we refer to as a single time series. This could be a sensor producing data points at regular intervals or tracking the electricity consumed by a client or household. When we collect several entities of the same type we refer to the set of entities as a time series dataset. We express a one-step-ahead forecast as

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \mathbf{x}_{i,t-k:t}, \mathbf{s}_i), \tag{1}$$

where $\hat{y}_{i,t+1}$ is the forecasted value, $y_{i,t-k:t} = \{y_{i,t-k}, \dots, y_{i,t}\}$ are prior observations of the target, $\mathbf{x}_{i,t-k:t} = \{\mathbf{x}_{i,t-k}, \dots, \mathbf{x}_{i,t}\}$ are any exogenous inputs, s_i is static metadata, and $f(\cdot)$ is the prediction function learned by the model [53]. Although one-step-ahead forecasts are useful, several applications require predictions for multiple time steps into the future. We will refer to the process of forecasting several steps into the future as multi-step forecasting,[4] which can be written as

$$\hat{y}_{t+\tau} = f(y_{t-k:t}, \mathbf{x}_{t-k:t}, \mathbf{u}_{t-k:t+\tau}\mathbf{s}, \tau), \tag{2}$$

where $\tau \in \{1, \dots, T\}$ is the discrete forecast horizon, $\mathbf{u}_{t-k:t} = \{\mathbf{x}_{t-k}, \dots, \mathbf{x}_t, \dots, \mathbf{x}_{t+T}\}$ are known covariates such as date-time information and x_t are historically observed exogenous inputs [53]. Note that we have omitted the entity notation i in (2) for brevity.

Several methods have been proposed to perform multi-step forecasts, each with its pros and cons [11]. In general, there are two main approaches that are most commonly used: *recursive* methods and *direct* methods [53]. (1) Recursive methods work by iteratively feeding one-step predictions to the model, generating the next step until the required horizon has been forecasted. The recursive approach can be highly sensitive to noise, due to the error produced at each step potentially accumulating to large errors for longer horizons. As noted in [11], recursive methods are generally easy and inexpensive to train compared to other methods, requiring only single-valued outputs. (2) Direct methods were devised to avoid the accumulation of errors, by forecasting the entire horizon independently of other time steps. As previous forecasts are not used in the final output, the approach avoids the accumulation issue. On the other hand, the direct method can have difficulties learning complex dependencies between the forecasted variables, and typically require more computational resources [11]. The direct method is typically the method used when building neural networks, such as the Seq2Seq architecture [70].

A common way to benchmark time series forecasts is to compare them with the *naive method* for forecasting [37]. The naive method assumes that all future forecasts are equal to the previously observed value. When the time series is seasonal, we adjust the method by using the value observed at the same period in the previous season. The naive method can also be applied for multi-step forecasting, in which case the previously observed value is used as the forecast for the future T steps. The naive, seasonal naive, and multi-step naive methods are given by

$$\begin{aligned} \text{Naive}: \quad & \hat{y}_t = y_{t-1}, \\ \text{Seasonal naive}: \quad & \hat{y}_t = y_{t-m}, \\ \text{Multi-step naive}: \quad & \hat{y}_{t+T} = y_{t-1}, \end{aligned} \tag{3}$$

where m is the seasonal period and T is the maximum forecast horizon. Naive forecasts are useful for benchmarking models, and is the baseline commonly employed for computing *scaled errors* as a performance metric.

Assessing the performance of forecasting models is a challenging task for which several metrics have been proposed. [24] provides a list of the most commonly used metrics before 2006, later extended by [40]. The most common metrics for evaluating forecast performance are based on measuring the error between the predicted values, $\hat{y}_t$, and the ground truth values, y_t. We will refer to such metrics more specifically as *error metrics*. Note that the literature interchangeably refers to performance metrics as accuracy measures, performance measures, and evaluation metrics, to name a few. We will prefer the use of *performance metrics* when referring to any metric that assesses the performance of point forecasts in any given way.

Given a time series with n observations, let y_t denote the ground truth value at time t and $\hat{y}_t$ denote the forecasted value for y_t. Then, the forecast error at time t, e_t, is defined as $e_t = y_t - \hat{y}_t$. Provided a distribution of errors, the goal of selecting a metric is to provide an informative and clear summary of the error distribution while being aware that the selection of an appropriate metric will be highly dependent on the context in which it is used [7]. Table 1 provides an overview of the most commonly applied error metrics. Like [40], we will use the notation $\text{mean}(e_t)$ to denote the sample mean of errors $\{e_t\}$ over the period and time series of interest, depending on the context.

Most commonly applied metrics include *scale-dependent metrics* such as mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE). While being easy to compute and understand, they tend to be sensitive to extreme outliers and can produce biased results [5, 7]. *Percentage-based metrics* was proposed to be scale-independent, and thus can be used to compare forecast methods across datasets of differing scales. Mean absolute percentage error (MAPE) was used in the first M-competition, however, it has been criticized for producing anomalies when observed values are close to or equal to zero, and has a bias favoring forecasts that are below the observed values [17]. *Scaled error metrics* were introduced to address all of the aforementioned issues. [40] proposed the mean absolute scaled error (MASE) with the idea of scaling errors based on in-sample MAE from a benchmark method such as the naive method.

[4] Note that some literature refer to multi-step forecasting as *multi-horizon forecasting*. We prefer to use multi-step as the notion of horizon pertains to how far into the future one would like to forecast. Thus, multi-horizon can easily be confused with predicting over several different time horizons.

Table 1 Overview of common performance metrics based on forecast errors

Metric		Class	Equation
MSE	Mean Squared Error	Scale-dependent	$\text{mean}(e_t^2)$
RMSE	Root Mean Squared Error	Scale-dependent	$\sqrt{\text{MSE}}$
MAE	Mean Absolute Error	Scale-dependent	$\text{mean}(\lvert e_t \rvert)$
MAPE	Mean Absolute Percentage Error	Percentage based	$\text{mean}(\frac{\lvert e_t \rvert}{\lvert y_t \rvert})$
MASE	Mean Absolute Scaled Error	Scaled errors	$\text{mean}(\frac{\lvert e_t \rvert}{\text{mean}(\lvert y_t - y_{t-1} \rvert)})$

The error metrics are extended to the multi-step forecasting case by simply pooling the errors computed over the entire forecast horizon and computing the aggregate measure across all time steps to obtain a single value. This method of pooling and aggregating is commonly applied when evaluating a large number of models in empirical studies and forecasting competitions, see e.g., [11, 59, 61]. Although other performance metrics exist, they are typically custom-made for a specific context or situation in which more creative evaluations of forecasts are possible. These are typically industry settings, where notable examples are the evaluation of solar power forecasting [82], solar irradiance forecasting [73] and activity recognition [78]. In this paper, we aim to propose metrics that are applicable across a wide range of applications.

3.2 Metrics for evaluating multi-step forecasts

In order to establish a baseline for evaluation, we follow the general approach in the literature and compute the aggregate of an appropriate error metric to obtain a single score for each model and dataset. As recommended by [40], we use scaled errors that are independent of the scale of the data, have a well-defined mean and variance, and are symmetric by penalizing positive, negative, large, and small forecasting errors equally. Although [40] prefers the mean absolute scaled error (MASE) over root mean squared scaled error (RMSSE), it is often unclear which metric will be most preferable in different situations. For example, absolute errors are optimized for the median, which results in lower error scores for methods that forecast closer to the median value [66]. If we consider a dataset that contains sporadic ranges of $y_t = 0$, which are common in solar power generation data and sales data, then MASE will be favorable for methods that forecast smaller values. In contrast, squared errors are optimized for the mean and will prefer forecasts with larger values compared to MASE [43]. For this reason, a variant of RMSSE was selected for use in the recent M5 competition [56]. In our case, we employ a variety of datasets where the use of either metric could be warranted depending on the context. Therefore, we employ both the MASE and RMSSE metrics in our evaluation and showcase the implications of the different metrics when evaluating the results.

To obtain the MASE and RMSSE metrics, we first define the scaled errors prior to aggregating, i.e., prior to performing the mean and root mean across all observations. This is done to allow aggregation and grouping of the scaled errors across different dimensions. For example, we can compute the MASE per time series or per step in the forecast horizon. Additionally, we use the definition of scaled errors when deriving new metrics.

We begin by defining the absolute scaled error, ASE_t^i, of time series i at time t as

$$ASE_t^i = |q_t^i| \quad \text{where} \quad q_t^i = \frac{e_t^i}{\text{MAE}\big(Naive(i, T, m)\big)}. \tag{4}$$

In (4), q_t^i denotes the scaled error at time t for time series i, and $Naive(i, T, m)$ denotes forecasts obtained on the training data of time series i using the seasonal naive multi-step method with a T-step forecast horizon and seasonal periodicity m. Hence, the denominator consists of the mean absolute error (MAE) computed from the seasonal multi-step naive method on the training data. It is important to note that we scale the errors from each time series i *separately* because they can be of different scales. This is easily overlooked when pooling several hundreds of series as one single dataset, which is common practice when applying deep learning models. Moreover, note that we use seasonal naive multi-step forecasts as the baseline, in contrast to single-step forecasts. Naive single-step forecasts are more commonly applied for this purpose, likely due to their simpler computation, see e.g., [56]. We elect to use multi-step forecasts as this keeps the interpretability of scaled errors clearer. The naive multi-step forecasts accurately represent the benchmark that would be possible to obtain using the naive method, in contrast to naive single-step.[5] After obtaining absolute scaled errors, we compute the aggregated MASE metric as

$$MASE = \text{mean}\big(\text{ASE}_t^i\big), \tag{5}$$

where the mean operation is computed over all time series i and time steps t.

[5] Using the naive single-step forecast would imply that, forecasting across T-time steps, the model would iteratively have access to future information that a forecasting model would not.

Similarly, we define the squared scaled errors, SSE_i^t, of time series i at time t as

$$SSE_t^i = (q_t^i)^2 \quad \text{where} \quad q_t^i = \frac{e_t^i}{\text{RMSE}\big(Naive(i, T, m)\big)}, \tag{6}$$

q_t^i denotes the scaled error at time t and $Naive(i, T, m)$ denotes seasonal naive multi-step forecasts on the training set as above. Note that for squared scaled errors, the denominator is computed using the corresponding RMSE measure, as recommended by [40]. Again, the aggregated RMSSE metric is obtained by

$$RMSSE = \sqrt{\text{mean}\big(\text{SSE}_t^i\big)}, \tag{7}$$

where the root mean operation is computed over all time series i and time steps t.

MASE and RMSSE will be our baseline evaluation metrics and represent how empirical studies typically summarize and evaluate multi-step forecasts. To provide further insights that are neglected by only evaluating results in terms of aggregated metrics, we propose four novel metrics.

3.2.1 Win-loss metric

The aggregated MASE and RMSSE metrics tell us how well a model performs on an entire dataset overall, disregarding the information about which time series they perform well on. To identify how well models perform with respect to each time series, we propose a win-loss metric to rank models according to their win-loss counts.

Assume that we have a time series dataset $D = \{1, \ldots d\}$ where $i \in D$ denotes a single series in the dataset. Furthermore, assume that we evaluate a set of models $H = \{1, \ldots h\}$ on dataset D, where $g \in H$ denotes a single model in the set of models. Letting the aggregated error obtained by model g on time series i be denoted by $e_{g,i}$, we define the set of time series in which model g obtains the lowest error (best score) as W_g. The wins of model g is then given by

$$R_W = |W_g|, \tag{8}$$

where R_W represents the number of times model m obtains the best score, i.e., wins, on dataset D. Similarly, we define the set of time series in which model g obtains the highest error as L_g, whereby the losses can be written as

$$R_L = |L_g|, \tag{9}$$

where R_L is the number of times model g obtains the worst score on dataset D.

The win-loss metric provides further insights when evaluating models on datasets with a large number of time series. For example, consider the case where the aggregated error metric decisively indicates that one model is better than its peers. It is possible that it is better on average on all series, or conversely, it could perform well at some series and worse at others. It might turn out that it is optimal to use different models on different parts of the dataset. We will see concrete examples of this when we evaluate our results in Section 5.

Although one could combine wins and losses into a single metric, e.g., a ratio of wins to losses, we argue keeping the number of wins and losses in absolute terms provides the most information when analyzing results. Note that one drawback of the metric is that it is dependent on the underlying error metric to be computed for all models before the wins and losses can be calculated. Thus, it is not possible to use this type of metric for e.g., parameter optimization due to this dependency.

Note that the win-loss metric is not replacing the existing error metric that in many cases will be the deciding factor for model selection. Instead, it provides *additional* information to help the practitioner identify model strengths and weaknesses that can not be seen using an error metric alone.

3.2.2 Variance weighted errors

Seasonal time series that are driven by real-world phenomena, such as electricity consumption and traffic congestion, tend to be highly variable over the seasonal period. For example, traffic tends to be easy to predict during baseline hours and nighttime. The real uncertainty arises during rush hour around 08:00 AM and 16:00 PM. Electricity consumption is highly variable during the daytime, depending on external factors such as weather and consumer patterns. Furthermore, different parts of a time series might not be that relevant to forecast. It is not unreasonable to assume that practitioners are in most need of accurate forecasts when uncertainty is at its greatest, i.e., during rush hour traffic and peak electricity consumption. The standard error metrics do not account for these differences in uncertainty over the seasonal period when evaluating forecast performance. Therefore, we propose to include a proxy for uncertainty in the error metric.

For the purpose of including uncertainty in the error metric, we propose *variance-weighted errors*. By this approach, we weight the errors obtained at a particular time during the seasonal period by the variance of the target values at that same particular time. To illustrate this, Fig. 1 shows a plot of 30 consecutive days of measured traffic occupancy from a single time series. It should be evident that the variance around rush hour is significantly higher than during the middle of the day and during nighttime. Thus, we want to scale the errors of forecasts to closer match this distinct feature of the dataset. I.e., by increasing the importance of errors

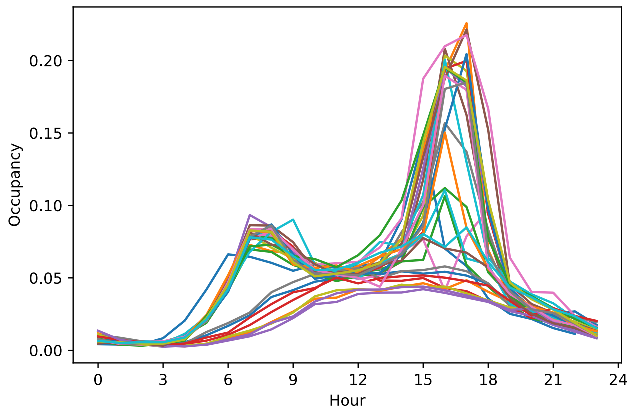

Fig. 1 An example of daily traffic occupancy plotted for 30 consecutive days

made under high uncertainty and decreasing the importance of errors when traffic is more stable.

To formalize this concept, assume we are given a dataset with a logical seasonal periodicity m and date-time covariates $c \in \{1, \ldots, m\}$.[6] The covariates represent a partitioning of the seasonal period. For example, given traffic measured at every hour, the seasonal period is $m = 24$ and each seasonal period constitutes a day with hours $c \in \{1, 2, \ldots, 24\}$. The covariate in this case is the hour of the day at which a forecast was made. Thus, the covariates c represent a logical partitioning of the seasonal period where one expects different behavior at different points of the period. We define the estimated variance, $\hat{V}_c$, of the target values y_t^c at covariate c as $\hat{V}_c = \text{Variance}(y_t^c)$, where $t \in \{c, c \cdot 1, c \cdot 2, \ldots, c \cdot \frac{n}{m}\}$. That is, $\hat{V}_c$ is the estimated variance of the targets occurring during covariate c. To obtain the variance weight, w_c, we normalize the estimated variance $\hat{V}_c$ by the sum of variances and scale by the seasonal periodicity:

$$w_c = \frac{\hat{V}_c}{\sum_{c=1}^{m} \hat{V}_c} \times m. \tag{10}$$

Therefore, by (10), $\sum_{c=1}^{m} w_c = m$, and each weight will act as a scaling factor according to how much the target varies during a particular covariate. In the special case where the variance is equal for all covariates, we have $w_c = 1$ for $c \in \{1, \ldots, m\}$ and the errors will not be changed by the weights. Note that we have omitted the superscript i notation representing the time series to simplify the notation.

[6] The Oxford Dictionary defines covariates as "independent variables that can influence the outcome of a given statistical trial, but which is not of direct interest."

Continuing, we define the *variance-weighted* MASE (MASE_{VW}) which we derive from the absolute scaled errors as defined in (4). Given weight w_c^i obtained on time series i at covariate c, we define the MASE_{VW} as

$$MASE_{VW} = \text{mean}\big(ASE_c^i \cdot w_c^i\big), \tag{11}$$

where ASE_c^i is the absolute scaled error obtained on time series i grouped by covariate c, and the mean operation is computed across all time series and covariates. Note that the reason for scaling prior to averaging is that the MASE_{VW} can be computed per time series or per covariate depending upon which variables the mean is performed over.

Similarly, we define the variance-weighted RMSSE (RMSSE_{VW}) as

$$RMSSE_{VW} = \sqrt{\text{mean}\big(SSE_c^i \cdot w_c^i\big)}, \tag{12}$$

where SSE_c^i and w_c^i are the absolute scaled error and variance weight obtained on time series i grouped by covariate c, respectively. The root mean operation is again computed across all time series and covariates.

3.2.3 Delta horizon metric

The two previous metrics address how models perform on individual time series in a dataset, and how forecasts across the seasonal period can mask the performance of models during critical time points. To gain further insight into how models behave with respect to the multi-step forecast horizon, we propose the *delta horizon* metric, or Δ_H. The Δ_H metric measures how much a multi-step forecasting model updates the forecasted value for y_t at time t as the time point approaches. For example, given a 24-step forecast at time t, the 24^{th} step is the forecasted value $\hat{y}_{t+24}$ for y_{t+24}. At $t+1$, the model produces a new forecast $\hat{y}_{t+24}$ for y_{t+24}, which is now a 23-step ahead forecast. In other words, the model updates its initial forecast for y_{t+24} as t approaches $t+24$. We might find that model X adjusts its forecasted value for y_t significantly over the forecast horizon, while model Y produces stable forecasts over the horizon, relative to X. This is what we aim to capture with the Δ_H metric.

To formalize the above, let $\hat{y}_t^\tau$ denote the τ-step ahead forecast for the observed value y_t and $\tau \in \{1, \ldots T\}$ where T is the forecast horizon. Then, we define the delta horizon metric as

$$\Delta_H = \text{mean}\big(|\hat{y}_t^\tau - \hat{y}_t^{\tau-1}|\big), \quad \text{for} \quad 2 \le \tau \le T, \tag{13}$$

where the mean operation is performed over both τ and t. Hence, the delta horizon metric measures the mean absolute difference between each update of $\hat{y}_t$ over the forecast horizon τ.

Similar to the win-loss metric, Δ_H is not proposed as a single metric to determine the best forecast. It provides additional context to the practitioner in the case where measuring the forecast stability is of high importance. Thus, the Δ_H is not a stand-alone metric and needs to be viewed in context together with an established error metric. We will see examples of this in Section 5.3 when discussing the experimental results.

3.2.4 Decomposed Error Metrics

The last metrics we propose are *decomposed errors*, which relate errors to the time series trend-cycle component, seasonality component, and remainder component (noise). A useful way of interpreting a univariate time series is to decompose it into three components by performing a *time series decomposition*. The additive and multiplicative decomposition models are written as

$$\begin{aligned} \text{Additive:} \quad & y_t = T_t + S_t + R_t, \\ \text{Multiplicative:} \quad & y_t = T_t \times S_t \times R_t, \end{aligned} \tag{14}$$

where y_t is the time series data point, T_t is the trend-cycle component, S_t is the seasonal component and R_t is the remaining component, all observed at time t [37].

As noted in [37], additive decomposition fits time series where the magnitude of seasonal fluctuations does not vary significantly with time. If the opposite is true, the multiplicative decomposition is more appropriate. Thus, multiplicative is typically used for, e.g., economic time series. When it comes to selection, it is up to the practitioner to select the best decomposition based on the time series characteristics. In this study, we showcase only the additive version to keep the number of performance metrics tractable while being suitable for the two highly seasonal datasets employed in this study: traffic and solar. Nonetheless, the multiplicative version can be derived by following the same approach.

There are several ways to perform the actual decomposition into T_t, S_t and R_t. We will use STL decomposition, which is a robust and versatile filtering procedure that allows the seasonal component to vary over time [21], to relate error metrics to trend, season, and noise. The purpose of this is to provide further insights to industry and practitioners when interpreting their results and provide insights that allow practitioners to build better models.

We begin by denoting the observed ground truth values y_t and the τ-step ahead forecast for the observed value y_t as $\hat{y}_t^\tau$ for $\tau \in \{1, \dots T\}$, where T denotes the maximum forecast horizon. Then, we define the *forecasted path* with forecast horizon τ as $p_\tau = \{\hat{y}_t^\tau, \hat{y}_{t+1}^\tau, \dots, \hat{y}_{t+n}^\tau\}$ where t and n are the starting point and length of the test series, respectively. Similarly, we define the path of the ground truth test series as $p_G = \{y_t, y_{t+1}, \dots, y_{t+n}\}$. Thus, p_1, p_T and p_G represents the path of all 1-step-ahead forecasts, T-step-ahead forecasts and ground truth test series, respectively. Note that we have omitted denoting the time series by index i to simplify notation.

With the above, we define decomposed errors as the error between each time series component resulting from the STL-decomposition of p_τ and p_G.[7] We let $\mathcal{T}_{t,p_\tau}$, S_{t,p_τ}, R_{t,p_τ} denote the trend-cycle, seasonal, and remainder components of p_τ respectively, and $\mathcal{T}_{t,p_G}$, S_{t,p_G}, R_{t,p_G} denote the components of p_G. Then, we define the decomposed trend, season, and remainder errors as

$$\begin{aligned} \text{Trend error:} \quad & e_{t,\mathcal{T}} = \mathcal{T}_{t,p_G} - \mathcal{T}_{t,p_\tau}, \\ \text{Seasonal error:} \quad & e_{t,S} = S_{t,p_G} - S_{t,p_\tau}, \\ \text{Remainder error:} \quad & e_{t,R} = R_{t,p_G} - R_{t,p_\tau}. \end{aligned} \tag{15}$$

In order to summarize the errors, we use the same approach as for scaled errors and calculate the RMSSE version of each decomposed metric as follows:

$$\begin{aligned} RMSSE_{\mathcal{T}} &= \sqrt{(q_{i,t,\mathcal{T}})^2} \quad \text{where} \quad q_{i,t,\mathcal{T}} = \frac{e_{i,t,\mathcal{T}}}{\text{RMSE}\big(Naive(i,T,m)\big)}, \\ RMSSE_S &= \sqrt{(q_{i,t,S})^2} \quad \text{where} \quad q_{i,t,S} = \frac{e_{i,t,S}}{\text{RMSE}\big(Naive(i,T,m)\big)}, \\ RMSSE_R &= \sqrt{(q_{i,t,R})^2} \quad \text{where} \quad q_{i,t,R} = \frac{e_{i,t,R}}{\text{RMSE}\big(Naive(i,T,m)\big)}, \end{aligned} \tag{16}$$

where $\text{RMSE}\big(Naive(i,T,m)\big)$ denotes the RMSE of the seasonal naive multi-step forecasts on the training set of time series i with seasonal period m, as when computed for scaled errors. Note that we include the i notation to show that each metric is scaled according to each individual time series. Decomposed errors can also be computed and aggregated by MASE instead of RMSSE, although we elect to compute only the RMSSE version to keep the number of metrics for the evaluation tractable.

The decomposed metrics indicate how accurately the model captures the trend, season, and remainder compo-

[7] For further details on how the STL decomposition is computed, we refer to [21, 37].

nents by each τ-step-ahead path, allowing practitioners and industry to see whether the error is mainly due to erroneous prediction of trend, season, or remainder (noise). We compute the decomposed errors of p_1 and p_T to compare how the models capture the time series components at the first and last prediction horizons. Henceforth, we will refer to the three metrics $RMSSE_{\mathcal{T}}$, $RMSSE_S$, $RMSSE_R$ as simply the "decomposed errors" for path p_1 and p_T.

Decomposed errors are, like win-loss and variance-weighted errors, not stand-alone metrics. They provide the practitioner with insights on performance against the trend, season, and noise component in the series. Thus, the metrics can not be used to select the best model alone. They do, however, provide the practitioner with novel insights as to how the model is performing against the underlying components to help the practitioner identify strengths and weaknesses to build better models.

3.3 Forecasting models

In order to showcase the applicability of the proposed metrics, we run experiments with five forecasting models: seasonal-ARIMA (SARIMA) [13], a generic sequence-to-sequence network (Seq2Seq) [70], a generic temporal convolutional network (TCN) [9], the DeepAR model [65], and the Temporal Fusion Transformer (TFT) [54]. The SARIMA model is commonly applied in the literature as a baseline for new forecasting models. Neural networks are commonly being trained and evaluated on large datasets using aggregated metrics, which is the typical use case where our proposed metrics can provide additional insights. Different architectures have been selected to provide more diverse forecasts that allow us to use the proposed metrics for a range of scenarios.

3.3.1 SARIMA

The SARIMA model is a generalization of the Box-Jenkins ARIMA model, which can accommodate data with both seasonal and non-seasonal features [13]. We specify a model by writing ARIMA(p, i, q), where p, i and q represent the autoregressive (AR) order, degree of integration and moving average (MA) order, respectively. The model can be written as

$$y'_t = c + \epsilon_t + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_p \epsilon_{t-p}, \tag{17}$$

where ϵ_t is white noise, y'_t is the first differenced series, $\phi_1, \ldots, \phi_n$ are the AR components, and $\theta_1, \ldots, \theta_p$ are the MA components [37].

Three of the datasets we employ in our experiments are highly seasonal, therefore, a SARIMA model is needed to account for seasonality. The SARIMA model extends ARIMA by including differences at lags equal to the seasonal period, m, to remove seasonal effects from the series. Note that, although more advanced versions of the ARIMA model exists, most comparison studies benchmark against the standard SARIMA model. For further details on estimating the SARIMA model, we refer to [37].

To select appropriate model orders, we use the stepwise algorithm by [39], designed to automate model selection when forecasting a large number of univariate time series. The selection of the optimal model order is based on minimizing the Akaike information criterion (AIC) [2]. Further details on the experimental application of the SARIMA model is provided in Section 4.

3.3.2 Sequence-to-sequence network

The first deep learning model we implement in our study is based on a general sequence-to-sequence architecture, also referred to as an encoder-decoder architecture. It was popularized by [70] and has historically produced strong results on natural language processing tasks [80]. As time series can be viewed as sequences of inputs and targets, several sequence-to-sequence architectures have been proposed for forecasting [64, 65, 77]. We implement a generic version of the sequence-to-sequence architecture which we will henceforth refer to as the Seq2Seq model.

In accordance with [70], we implement Seq2Seq by combining an encoder and a decoder in the form of two *long short term memory networks* [35] (LSTMs). LSTMs are better at exploiting long-range dependencies, making them suitable for time series modeling. The Seq2Seq model will be trained solely on the time series data, with no additional static metadata or date-time covariates as features.

3.3.3 Temporal convolutional network

Temporal convolutional networks are based on convolutional layers and will provide a novel architecture type for our experiments. Furthermore, TCNs have recently been shown to outperform canonical RNN and LSTM architectures on selected sequence-to-sequence tasks [9] and will be interesting to compare to the Seq2Seq model. Several versions of the temporal convolutional architecture have been proposed for different use cases [23, 33]. We follow the architecture proposed by [9], implementing a generic TCN that combines simplicity, autoregressive prediction, and long memory. For further details, we refer to their original paper [9].

Similar to the Seq2Seq model, we do not add any static metadata or date-time covariates to the TCN model. In the following two sections, we present the DeepAR and TFT models, which will incorporate static and date-time covariates as features.

3.3.4 DeepAR

DeepAR is a probabilistic model based on an autoregressive RNN architecture, developed by Amazon Research for probabilistic sales and demand forecasts [65]. DeepAR outperformed several models for demand forecasting, including Croston and ETS [36, 39], and outperformed the matrix factorization model [81] at point forecast accuracy on the electricity and traffic datasets, highlighting its applicability for point forecasting. The model has since been established as a benchmark when proposing new state-of-the-art deep learning architectures.

In accordance with [65], we use Gaussian likelihood, $\ell(y|\mu, \sigma)$, as the distribution model, where the Gaussian likelihood is parameterized by the mean and standard deviation, i.e., μ, σ, respectively. When predicting, we feed the model the input series $\mathbf{y}_{i,t-k:t}$ and sample $\hat{y}_{i,t} \sim \ell(\cdot|\theta_{i,t})$, before refeeding $\hat{y}_{i,t}$ to the model for the next prediction step and repeating the process to the end of the prediction horizon T, generating one sample path. Repeating the sampling process generates several paths representing a joint predicted distribution for the $\tau \in \{1, \dots, T\}$ prediction steps. The point forecast is then simply the mean of the predicted distributions. For more details on the model, we refer to the original paper by [65].

3.3.5 Temporal fusion transformer

The last model we implement is the TFT developed by Google Research [54], achieving state-of-the-art performance on multi-step forecasting compared to several recently proposed models [52, 64, 65, 79, 81]. The model is built on an attention-based RNN architecture and combines several components, including gating mechanisms, variable selection networks, covariate encoders, temporal attention, and quantile forecasts. The TFT will represent the state-of-the-art model in our experiments that combines elements from several archetypal network architectures, most notably the transformer architecture [75].

The TFT model, like the DeepAR model, produces prediction intervals, however, it does so by quantile regression rather than sampling from estimated distributions. Quantile forecasts are given as

$$\hat{y}_i(q, t, \tau) = f_q\big(y_{i,t-k:t}, \mathbf{z}_{i,t-k,t}, \mathbf{x}_{i,t-k,t+\tau}, \mathbf{s}_i, \tau\big), \qquad (18)$$

where $\hat{y}_i(q, t, \tau)$ is the predicted q^{th} quantile of the τ-step ahead forecast at time t and f_q is the prediction model [54]. Note that the TFT model separates time-dependent known and unknown covariates, treating them differently in the architecture. Known covariates, i.e., time of day or day of week at time t, are denoted as $x_{i,t}$ and unknown covariates such as additional external regressors are denoted as $z_{i,t}$. We refer to [54] for the complete TFT architecture description.

4 Experiments

In this section, we present details on the experiments we conducted in order to answer the research question. Section 4.1 provides an overview of the datasets employed. Section 4.2 and 4.3 presents the training procedure for each implemented model. To obtain realistic and accurate results, we closely replicate the experimental setup defined in each model's respective papers. Lastly, Section 4.4 provides details on the software and hardware used.

4.1 Data

To show the applicability of the proposed metrics under different contexts, we select five datasets for experimentation: electricity, traffic, volatility, solar, and wind. Each dataset has distinct characteristics and represents important real-world industry applications where forecasting models are rigorously used. Furthermore, the datasets have been routinely applied for model evaluation in previous research. Each dataset is split into three parts: training, validation, and testing, used for model learning, hyperparameter tuning and evaluation, respectively. An overview of each dataset is presented in Table 2.

Electricity The Electricity Load Diagrams dataset is collected from the UCI Machine Learning Repository [27] and is commonly used as a benchmark for forecasting models [52, 54, 64, 65, 77]. It exhibits high daily seasonality, and each time series varies significantly in magnitude, i.e., $y_t \in [0, 1 \times 10^5]$. Following [54, 65, 81], we use the period between 2014-01-01 to 2014-08-07 for training, 2014-08-08 to 2014-08-31 for validation and the immediate following week for testing.

Traffic The PEMS-SF dataset is collected from the UCI Machine Learning Repository [27] and is typically used as a benchmark alongside electricity, again see e.g., [52, 54, 64, 65, 77]. The dataset features high daily seasonality in addition to peak-hour traffic spikes. Like previous studies, we use the data prior to 2008-06-15 for training and validation, where the last 10% is used as validation. The week immediately following validation is used for testing [54, 65, 81].

Volatility The volatility dataset is collected from the OMI realized library [34] comprising of daily realized volatility

Table 2 Overview of datasets, dataset metadata and static parameters employed for experiments

	Electricity	Traffic	Volatility	Solar	Wind
Number of series	369	963	31	137	28
Total observations	2198k	3997k	151k	7200k	306k
Frequency	Hourly	Hourly	Daily	10 min.	Daily
Seasonal period	24	24	5	144	30
Target domain	$\mathbb{R}_{0\geq}$	[0, 1]	$\mathbb{R}$	$\mathbb{R}_{0\geq}$	[0, 1]
Training samples	500k	500k	100k	2000k	50k
Input length	168	168	252	240	210
Output length	24	24	5	24	30

computed from the intraday data of 31 stock indices where each index is treated as a time series. The volatility dataset is noisy with no definite seasonality and contains fewer observations than the previous datasets. It was used in the study by [54] to contrast with the strongly seasonal electricity and traffic datasets and show the interpretability advantages of the TFT model. Data before 2016 is used for training, between 2016 and 2017 for validation, and 2017 to 2019-06-28 for testing [54].

Solar The solar power dataset is provided by NREL[8] and has been used for model evaluation in [47, 52]. The dataset exhibits daily seasonality and intermittent periods of zero power production during nighttime, adding an interesting dynamic. The phenomenon of sporadic periods of $y_t = 0$ can also be found in sales forecasting, which was the task selected for the recent M5 competition [56]. The data is split with training data prior to November 2006, validation data during November, and testing data during December of the same year.

Wind The wind power dataset is collected from Kaggle,[9] measuring the percentage wind power output of 29 countries, and was used in the study of [52]. The wind dataset is extremely noisy with very slight yearly and monthly seasonality. Furthermore, in contrast to the other datasets, wind power is completely independent of known time inputs and will serve as an interesting comparison to the other datasets. We use data prior to 2014 as training, between 2014 and 2015 as validation and the year of 2015 for testing in accordance with [52].

All datasets are preprocessed by adding date-time information to each time series, i.e., date-time covariates. The date-time covariates will serve several purposes: (1) they are used as known inputs by the DeepAR and TFT model. (2) seasonal date-time covariates such as hour of day is needed to compute the variance weighted errors. Further details on preprocessing is provided in Appendix A.1.

[8] https://www.nrel.gov/grid/solar-power-data.html

[9] https://www.kaggle.com/sohier/30-years-of-european-wind-generation

4.2 Fitting the SARIMA models

We are unable to find adequate details on how previous research has implemented and fitted SARIMA models, despite the large base of papers that compare their models to an SARIMA baseline [52, 54, 64, 83]. At most, papers only provide implementations for *their* proposed model, while stating that they use the `auto.arima` function from the R library for the SARIMA baseline[39]. Therefore, we design a custom framework for fitting the SARIMA to the datasets.

We find the most reasonable approach to be fitting a single SARIMA model for each individual time series in each dataset. The fact that we fit a single model to each series means that the SARIMA models have no opportunity to learn from other time series in the dataset, unlike the deep learning models.

For fitting the models, we use a rolling window approach, fitting the model to a moving window consisting of the most recent data as described in [37]. This is done for three reasons: One, it means the models will only be fitted on the most relevant data. Two, the model will react to any changes in the data distribution as the rolling window moves in time. Three, it significantly reduces the required computational time for selecting and fitting the most appropriate model. Like previous research we use `auto.arima`, and limit the search to a model of order SARIMA(5, 2, 5)(5, 2, 5), using stepwise search with approximation [39]. The search is performed on the most recent moving window prior to the test period. When forecasting, the moving window is moved forward and the model is updated and refit before forecasting T-steps ahead.[10] When fitting the seasonal component of the model, we use seasonal period m as defined in Table 2. Details on the lookback window used for the rolling window approach are provided in Appendix A.2.

[10] Note that refitting the model during the prediction period is what any practitioner would do in practice. Although this looks like target leakage, it is just a version of time series cross validation performed on the test set and is detrimental to the state-space modeling approach, see [37, Chapter 5.10].

To summarize our approach, for each time series in the dataset, we use `auto.arima` to find the optimal model order and update the model iteratively on the moving window as we produce T-step ahead forecasts for each time step. This approach keeps the problem tractable (considering we have a total of 1528 time series to fit), allowing us to fit a large number of models without having to manually intervene on each fit.

4.3 Training the deep learning models

We follow the general approach of previous research when training the deep learning models. Training windows are created and sampled following [65]. Denoting a dataset by $\{\mathbf{y}_{i,1:N}\}_{i=1}^{M}$ where M is the number of different time series and N is number of available observations per time series, training instances are sampled with a fixed lookback window of length k and forecast horizon of length T. Sampled windows then consists of input observations $\mathbf{y}_{i,t-k:t} = \{y_{i,t-k}, \ldots, y_{i,t}\}$ used to forecast the values $\hat{\mathbf{y}}_{i,t:t+T} = \{\hat{y}_{i,t}, \ldots, \hat{y}_{i,t+T}\}$ for different starting times t. The total training data set consists of H sliding windows $\{\mathbf{y}_{i,t-k:t+T}\}_{t \in H, i \in M}$. We ensure that the same selection of samples is used and loaded when training each deep learning model. Hence, each model will see the same training and validation data. Sample size per dataset is summarized in Table 2. In accordance with [54, 65], we use 500k samples for training on electricity and traffic and 100k samples for volatility. In accordance with [52], we use 50k samples on wind. Lastly, there is no predefined scheme for the solar dataset. Hence, we elect to use 2000k samples, as the dataset is considerably larger, and we find this to work well in preliminary experimentation. Covariates for each model are selected for the DeepAR and TFT model as in previous research [54, 65], the specification for which is presented in Table 3. Lastly, all models are trained with early stopping on the validation loss, with a *patience* of 5 epochs in accordance with [54].

We use the same hyperparameters from previous research where this is available. [54] provides hyperparameters for the TFT model on traffic, electricity, and solar. For DeepAR we use the hyperparameters for electricity and traffic as stated in the original paper [65]. For the other datasets and models where previous research is not available, we use the Optuna hyperparameter optimization framework [3]. For details on the hyperparameter search grid and the final parameters per model, see Appendix A.1.

Table 3 Overview of the covariates added to the DeepAR and TFT model for each dataset

Covariates	Electricity	Traffic	Volatility	Solar	Wind
Time series ID	✓	✓	✓	✓	✓
Time from start	✓	✓	✓	–	–
Hour of day	✓	✓	–	✓	–
Day of week	✓	✓	✓	–	–
Day of month	–	–	✓	–	–
Month of year	–	–	✓	–	✓
Week of year	–	–	✓	–	–

4.4 Implementation

Original code for the implemented models, hyperparameter tuning, proposed metrics, and evaluation framework is available and open source for full reproducibility of our results. PyTorch was used to implement the Seq2Seq model and TCN model [62]. The DeepAR and TFT models have been implemented and configured using the PyTorch Forecasting library [10]. To make it possible to train several neural network architectures within the same framework, we used the PyTorch Lightning for orchestration [28]. The SARIMA model was implemented in the R language, utilizing the `auto.arima` function [39].[11]

The NTNU IDUN computing cluster was used for all model training [69]. The deep learning models have been tuned and trained utilizing a single Nvidia Tesla V100 GPU. Hyperparameter tuning required up to 5 days of run-time per model, and the training of models up to 12 hours per model. The SARIMA models in R were fitted utilizing a single CPU, requiring run-time of up to 5 days. Note that neural networks are typically slower to train compared to SARIMA models, given that they also require more data and are more complex to fit. However, the neural networks have the advantage of high parallelizability when training on a GPU, also being computationally faster than a single CPU. The speed of fitting the SARMIA models could be significantly improved by parallelization, given that we have to fit a separate model for each single time series in each dataset.

5 Results and discussion

In this section, we present and evaluate the experimental results. First, we evaluate our results when aggregating a single error metric which is the approach applied in the literature. Furthermore, we show how the proposed win-loss metric provides important information that cannot be obtained by the error metric alone. Second, we show how differences in the seasonal period can impact the choice of model using variance weighted errors. Third, we evaluate the results in relation to the forecast horizon and discuss the applicability of the proposed delta horizon metric. Lastly, we

[11] Although implementations of `auto.arima` exist in Python, we found the R implementation to be significantly faster, while also being most commonly used package in the literature.

use the decomposed error metrics to gain additional insights that relate the errors to the concepts of trend, season, and noise.

5.1 Evaluation by the win-loss metric

We begin by evaluating results in terms of aggregated errors, i.e., averaging errors across both prediction horizon and time series in the dataset. This results in a single value per metric for each model and dataset. This approach represents how models are evaluated and compared in the literature. Table 4 presents aggregated errors in terms of the MASE and RMSSE metrics.

In general, we find that TFT, Seq2Seq, and TCN are the best-performing models. They consistently perform well on all datasets, obtaining relatively similar scores (within 0.03 points excluding traffic and solar MASE). In contrast, the worst-performing model tends to be SARIMA, although the model performs reasonably well on volatility. Furthermore, we find that DeepAR is considerably worse on electricity and volatility, but is the second best on solar (RMSSE) and wind (MASE). The TFT model is considered state-of-the-art and is the best-performing model on three out of five datasets in terms of MASE. Surprisingly, we find that less complex models such as Seq2Seq and TCN outperform the TFT model on datasets where it should have an inherent advantage.

Another interesting finding is the considerable difference between MASE and RMSSE, most notably on the traffic and solar datasets. As discussed in Section 3.2, MASE favors forecasts towards the median in contrast to RMSSE which favors forecasts towards the mean [43, 66]. The solar dataset is skewed towards 0 as the sun shines only during daytime, leaving the nightly periods (i.e., roughly half of the dataset), with 0 in power output. Thus, MASE will assign lower values to models that produce forecasts closer to 0. On solar, we observe that TFT performs best in terms of MASE and Seq2Seq in terms of RMSSE. This highlights the importance of selecting an appropriate error metric. For solar it seems reasonable to use RMSSE, as forecasting the mean (i.e., higher values and actual power production in this case), should be favored rather than correctly forecasting that there will be little to no sun during the night. We observe a similar effect on traffic, where TFT is 0.037 points better than TCN by MASE, and only 0.005 points better by RMSSE.

Note that when we evaluate using a single error metric, it only provides an indication of which models perform better *on average*. However, we obtain little insight about *where* the models are outperforming. Table 5 presents results using the win-loss metric, showing the number of times a model is the best (or worst) performing model per time series in the dataset. In other words, errors are not averaged across all time series, but rather evaluated per series while counting wins and losses for each model.

First, we observe that the TFT model is the most frequent winner on electricity, although the TCN model scored best in terms of aggregated error. This could stem from a difference in losses, where TCN is worst on only 3 series, in contrast to TFT being worst on 17 (22) series when measured by MASE (RMSSE). Another possibility is that the TCN model performs well on average on the majority of series without trending towards the extreme predictions, as it has fewer wins, but also fewer losses compared to TFT. This can be a desirable property, because it is typically better to avoid major errors while continuously predicting reasonably well rather than providing both extremely accurate and inaccurate forecasts.

Another interesting finding is the win-loss metric for the SARIMA model. On electricity, SARIMA is frequently both the best and the worst model (80 wins and 97 losses in terms of MASE). This fact is entirely neglected by the aggregated errors, in which SARIMA seems to be a bad model. However, as the win-loss metric reveals, it is in fact the best model on 22% of the series in the dataset. This would provide the practitioner with important insights: (1) the SARIMA model

Table 4 Results in terms of MASE and RMSSE aggregated across the prediction horizon and each time series

	Electricity		Traffic		Volatility		Solar		Wind	
	MASE	RMSSE	MASE	RMSSE	MASE	RMSSE	MASE	RMSSE	MASE	RMSSE
SARIMA	0.289	0.358	🟥 0.329	🟥 0.463	0.682	0.660	🟥 0.361	🟥 0.459	🟥 0.93	🟥 0.878
DeepAR	🟥 0.322	🟥 0.380	0.227	0.362	🟥 0.704	🟥 0.678	0.274	0.332	0.817	0.773
Seq2Seq	0.274	0.330	0.177	0.303	🟩 0.669	🟩 0.647	0.259	🟩 0.327	0.826	0.767
TCN	🟩 0.252	🟩 0.312	0.167	0.298	0.678	0.652	0.318	0.354	0.827	0.769
TFT	0.260	0.330	🟩 0.130	🟩 0.293	0.689	0.664	🟩 0.251	0.345	🟩 0.796	🟩 0.761

Note: The best and worst scores are marked green and red respectively

could be inappropriately specified for a subset of series where the model is significantly underperforming, given that it is the best model on a different part of the dataset. (2) if (1) is not the case, then SARIMA is simply very well suited for certain series, and unfit for others. This suggests that the practitioner can reach much better aggregated error scores by identifying which type of series SARIMA performs well on, and use a different model for the difficult to predict series. It could also be a good opportunity to combine forecasts of different models to improve the accuracy.

Continuing, the win-loss metric also shows that SARIMA is the worst model on nearly all traffic series, likely due to SARIMA fitting the general seasonal pattern of the series while being unable to adequately fit the rush hour traffic spikes. Again, the practitioner should check the specification of the model before concluding that it is unusable for this type of data.

On the solar dataset, we see the same discrepancy between evaluations using MASE and RMSSE. For solar, TFT is the best model on 76 series when measured by MASE and only 19 when measured by RMSSE. This was also reflected in the aggregated errors. We elect to continue the evaluation in the following sections using RMSSE not to complicate the evaluation and the number of metrics. Furthermore, it seems reasonable to prefer forecasts towards the mean in cases such as the solar dataset, as was argued in [56].

One limit of the win-loss metric is that it only provides additional information when used in unison with the typical aggregated error metric. For example, looking at only win-loss, we do not know at what level the S2S model performs. It is both best and worst on a small subset of series (56 out of 369 series in total for MASE), and we do not have any information about what happens in the cases where S2S neither wins or loses. For that, we still need to look at the MASE and compare it against the other models.

5.2 Weighting Errors by Variance

We have hypothesized that aggregating errors across the prediction horizon and time series can potentially make models that are good at predicting what is already known appear to be better than models that are good at predicting uncertain events. This is especially important for time series with characteristics rooted in real-world phenomena, such as power consumption, traffic flows and solar power generation, which also are important industrial applications. To account for such differences in variance over the seasonal period, we analyze the results in terms of the proposed variance-weighted errors.

Firstly, we focus on the first row of Fig. 2. It shows plots of the RMSSE metric aggregated per hour for the electricity, traffic, and solar datasets. Unsurprisingly, we observe that there are significant differences in mean error depending on what time of day is forecasted. Power consumption is difficult to predict during the day and evening, traffic is difficult to predict at 07:00 AM and 16:00 PM during rush hour, and

Table 5 Results in terms of the win-loss metric calculated based on MASE and RMSSE for each time series

	Electricity		Traffic		Volatility		Solar		Wind	
	Wins	Losses	Wins	Losses	Wins	Losses	Wins	Losses	Wins	Losses
SARIMA										
- MASE	80	97	2	🟥 853	1	1	0	🟥 94	0	🟥 28
- RMSSE	67	109	3	🟥 850	1	3	0	🟥 127	0	🟥 28
DeepAR										
- MASE	21	🟥 209	27	101	4	🟥 19	49	1	1	0
- RMSSE	25	🟥 192	49	103	2	🟥 20	🟩 79	1	6	0
S2S										
- MASE	13	43	5	3	🟩 17	0	12	0	1	0
- RMSSE	24	43	107	1	🟩 22	0	39	0	8	0
TCN										
- MASE	109	3	22	5	3	5	0	42	3	0
- RMSSE	110	3	131	5	4	1	0	4	3	0
TFT										
- MASE	🟩 146	17	🟩 907	1	6	6	🟩 76	0	🟩 23	0
- RMSSE	🟩 143	22	🟩 673	4	2	7	19	5	🟩 11	0

Note: The best and worst scores are marked green and red respectively

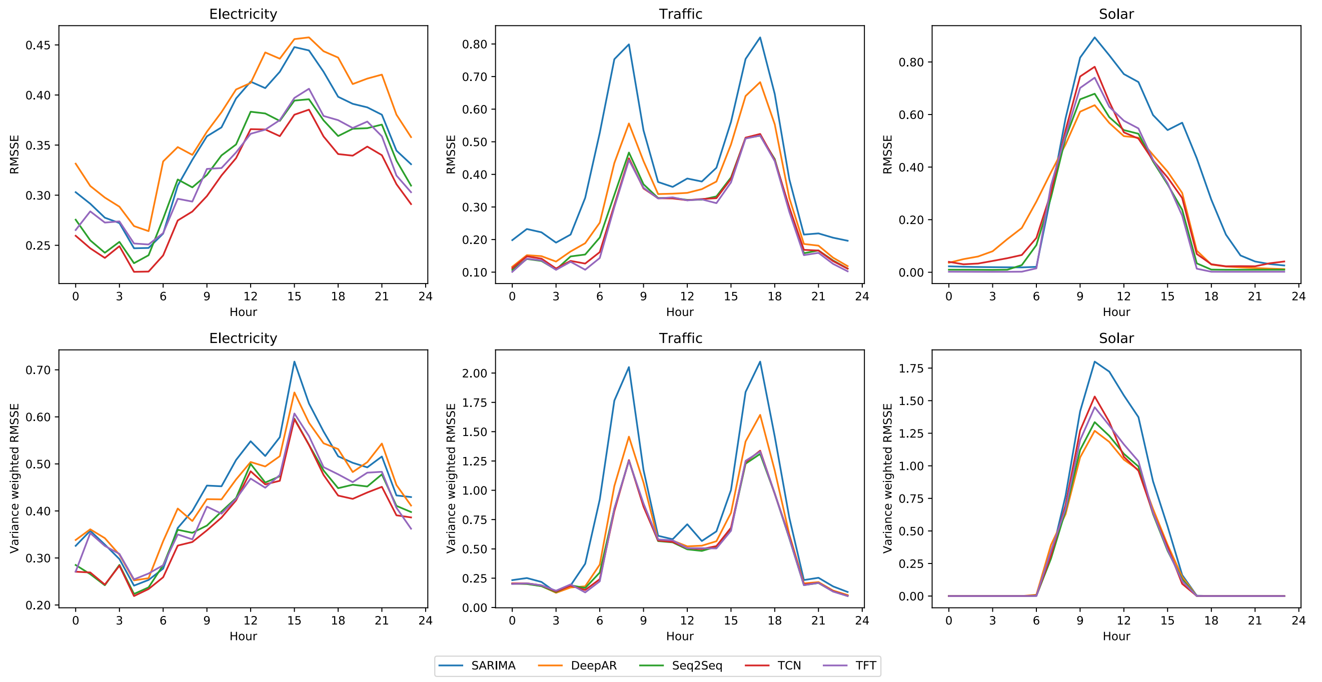

Fig. 2 Results in terms of RMSSE (first row) and variance weighted RMSSE (second row) computed per hour of day for the electricity, traffic, and solar datasets

solar power generation is zero during night. Interestingly, Fig. 2 shows that, on average, models are very consistent and follow the same error pattern. A model with a lower error score tends to be better at forecasting across all hours. In other words, we do not find that one model is very good at predicting peak traffic while another model is better for baseline traffic. This, however, could be due to the law of large numbers and averaging across all series. As we found in the previous section, although one model could be best in terms of aggregated errors, another model can still have significantly higher wins with lower average error. Thus, we do not know whether there are significant differences on each time series prior to aggregation. We do, however, find one significant difference on the solar dataset. That is, DeepAR is the worst model at predicting nighttime power production which we already know is 0, and thus irrelevant. At the same time, DeepAR is the best model when predicting solar generation during peak hour. In other words, the model is punished for making bad predictions for something we already know, outweighing the improvements it makes during critical hours. A simple solution would be to just disregard the periods when the sun is down, however, this would not be as easily applicable to datasets such as electricity or traffic.

To account for the differences in uncertainty we proposed variance-weighted errors, weighting errors in relation to how much variance there is at the predicted hour. The second row

Table 6 Results in terms of aggregated RMSSE and RMSSE$_{\mathrm{VM}}$ for each model and dataset

	Electricity		Traffic		Volatility		Solar		Wind	
	RMSSE	RMSSE$_{VW}$	RMSSE	RMSSE$_{\mathrm{VM}}$	RMSSE	RMSSE$_{\mathrm{VM}}$	RMSSE	RMSSE$_{\mathrm{VM}}$	RMSSE	RMSSE$_{\mathrm{VM}}$
SARIMA	0.358	0.461	0.463	0.990	0.739	0.745	0.459	0.779	0.878	0.880
DeepAR	0.380	0.450	0.362	0.748	0.759	0.767	0.332	0.560	0.773	0.775
Seq2Seq	0.330	0.406	0.303	0.638	0.724	0.732	0.327	0.576	0.767	0.769
TCN	0.312	0.394	0.298	0.642	0.730	0.737	0.354	0.618	0.769	0.770
TFT	0.330	0.415	0.293	0.642	0.744	0.751	0.345	0.612	0.761	0.762

Note: The best and worst scores are marked green and red respectively

of Fig. 2 plots RMSSE_{VM} for each prediction hour. Most notably, we see that errors on the solar dataset between 18:00 PM and 06:00 AM have been scaled to 0 as this period has no variance in power output. Furthermore, the error during peak output has increased significantly in magnitude, as more weight is put on the period where power output is highly variable. The same result is visible, although not as pronounced, for electricity and traffic. Differences between models where the target is relatively predictable (i.e., low target variance) have been reduced, whereas more weight is put on periods where the target is unpredictable, i.e., during peak traffic and electricity hours.

Table 6 shows RMSSE_{VM} measure for all datasets in addition to RMSSE as previously presented in Table 4 for comparison. Note that electricity, traffic and solar is weighted by the variance at each hour, while volatility and wind is weighted by the variance at each day of week due to having daily frequency. As expected, we observe significant increases in error magnitude for electricity, traffic and solar (up to 28.8%, 119% and 77.4% respectively). The datasets are highly seasonal and partly driven by known phenomena. In contrast wind and volatility shows only a marginal increase in magnitude (up to 1.11% and 0.30% respectively), as their variance is close to constant and independent of the seasonal period. As a consequence, we find that the best models change from TFT to Seq2Seq on traffic and from Seq2Seq to DeepAR on solar, which agrees with our discussion of DeepAR in Fig. 2. In this case, the practitioner should consider selecting the Seq2Seq model when forecasting for traffic or solar, if the goal is to have the best forecast during uncertain periods.

In addition to providing a more realistic representation of the forecast error, we argue that RMSSE_{VM} more accurately represents performance in relation to the naive forecasting method. Scaled errors have the benefit that they can be interpreted as being better than the reference method when the error is less than one and worse when greater than one [40]. However, for multi-step forecasting, the naive multi-step forecast will likely produce very bad forecasts for periods that are easy to forecast, e.g., baseline traffic or before sunrise and after sunset. Thus, when critical periods are weighted higher, the metric more closely represents the improvement over the reference method as models no longer have the benefit of major metric improvements when forecasting uninteresting periods. In the next section, we examine differences in how the models perform across the prediction horizon.

We note that for this experiment, all models have been trained to minimize a standard loss function. Thus, we generally see the same error pattern across the seasonal period, further exaggerated by the law of large numbers as noted previously. An open question is whether it would be possible to parameterize models by optimizing for variance-weighted errors to ensure that models are more accurate during high variance periods. We propose investigating this as a measure to increase the stability and robustness of models. Furthermore, using different loss functions or different forecasting methods other than the Recursive or Direct method could yield larger differences and better showcase the usability of this metric. We suggest these potential paths for future work.

5.3 Evaluation over the forecast horizon

When evaluating multi-step forecasts, an important aspect that is not considered explicitly when looking at aggregated errors is the forecast horizon. In this section, we evaluate our results in terms of RMSSE over the forecast horizon and the delta horizon metric, Δ_H.

Figure 3 shows RMSSE values per prediction horizon $\tau \in \{1, \ldots, T\}$ for each dataset. As we observed in Fig. 2, models trained on the same dataset tend to follow the same pattern in error distribution. Models that perform better in terms of aggregated error also perform better across the prediction horizon compared to a model with higher errors. We find little evidence of individual differences. Differences are most noticeable for electricity, where TFT is the worst model at prediction time $t = 0$, but better relative to other models later in the prediction horizon. Conversely, SARIMA is second best at time $t = 0$, but second worst after a few time steps. In general, error tends to be monotonically increasing across the horizon, with minor exceptions for electricity and wind. For volatility, the increase in error over the horizon appears to be close to linear, which is what we should expect for a highly noisy dataset. Interestingly, on the wind dataset, models are no worse at predicting wind 4 days in advance than 30 days in advance. As long as the models derive a reasonable prediction for the average wind level a few days in advance, the same prediction will yield similar accuracy for the rest of the horizon. What this shows is that, in general and on an aggregated level, models tend to keep their performance advantages across the entire forecast horizon. The exception is during the first few time steps. According to these results, we do not find evidence for the preference of one model over another to predict well on specific parts of the horizon. Most of the time, the practitioner can select the best model overall. However, despite largely similar and indifferent behavior in terms of errors across the horizon, we find large differences in the Δ_H metric.

Table 7 shows the mean Δ_H metric for each model and dataset. Recall that Δ_H measures how much the model adjusts its estimate of y_t as t approaches over the forecast horizon. First, we find that although TCN and TFT perform similarly in terms of aggregated RMSSE (0.312 and 0.330, respectively), they are significantly different in terms of Δ_H (12.8274 and 4.9080, respectively). In other words, TCN is

Table 7 Results for the Δ_H metric for each model and dataset

	Electricity	Traffic	Volatility	Solar	Wind
SARIMA	5.9976	0.0009	0.1168	0.3890	0.0062
DeepAR	11.0895	0.0022	0.1036	0.3360	0.0058
Seq2Seq	10.4219	0.0009	0.1070	0.3104	0.0030
TCN	12.8274	0.0023	0.0863	0.3170	0.0044
TFT	4.9080	0.0003	0.1182	0.2434	0.0020

Note: The highest and lowest scores are marked green and red, respectively

highly sensitive over the horizon and frequently updates its initial estimates. In contrast, TFT is more stable and follows the initial forecasted value more closely over the horizon, effectively making fewer adjustments. This can be of high importance to the practitioner under different use cases. Typically, you want stable forecasts and a model that will make less abrupt changes to its initial forecasts [74]. This is because the cost of making new decisions based on updated forecasts can be high, and a stable forecast will reduce the need to make new decisions and incur additional costs. If stability is important, the practitioner should choose the TFT model with a significantly lower Δ_H, even though RMSSE is slightly higher. On the other hand, stable forecasts also imply that the model is less reactive to new information. Reactivity can be favorable in situations where arriving information is what drives the time series (e.g., volatility forecasting which is highly impacted by newsfeeds). Thus, higher Δ_H can be useful in applications such as stock market predictions, where the cost of updating your positions is low and impact of recent news on the future stock price is high. In the end, the practitioner needs to consider the pros and cons of their specific use case.

In addition to looking at the aggregate Δ_H metric, we can plot the metric across the prediction horizon to show at which points each model makes the most adjustments (on average). Figure 4 shows the Δ_H metric plotted across the forecast horizon. We observe that there are significant differences as to how and where the models update their forecasts. We note that Δ_H of the SARIMA model produces a strictly increasing function over the horizon, which makes sense as it is a recursive model. On the other hand, direct deep learning models occasionally update forecasts more at the beginning and end of the horizon, as seen on electricity and traffic.

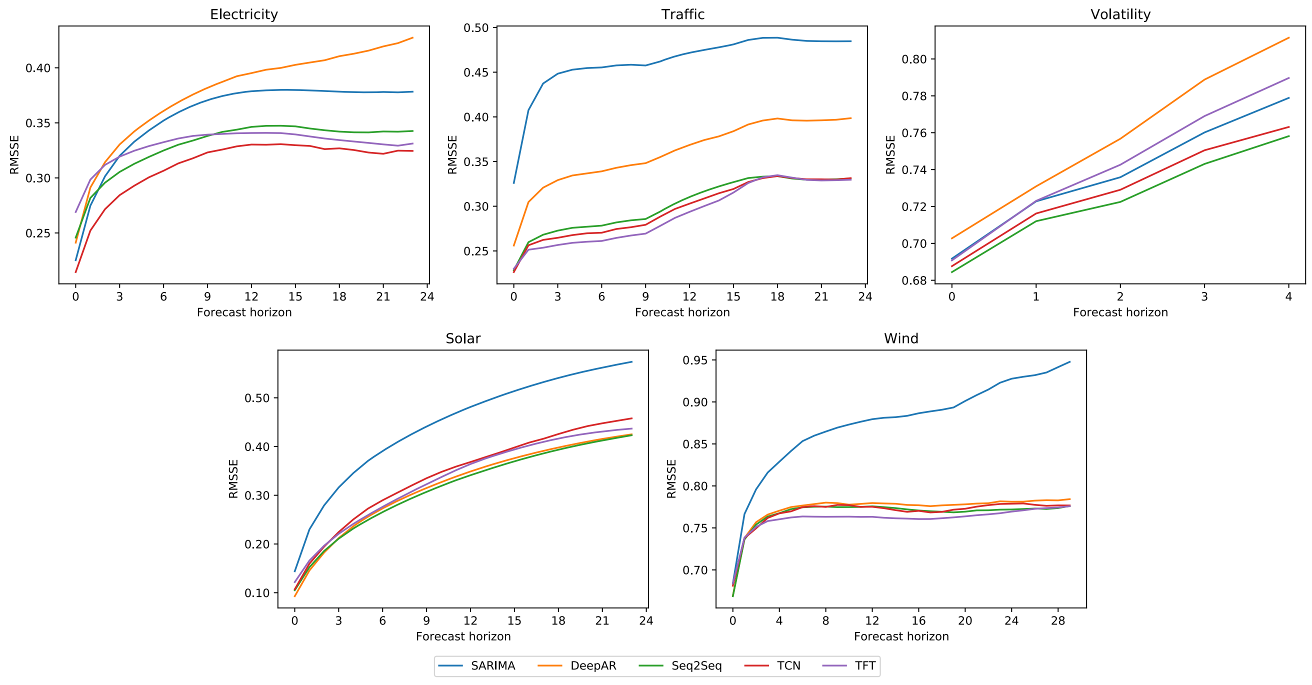

Fig. 3 Results in terms of RMSSE computed per forecast horizon for all datasets

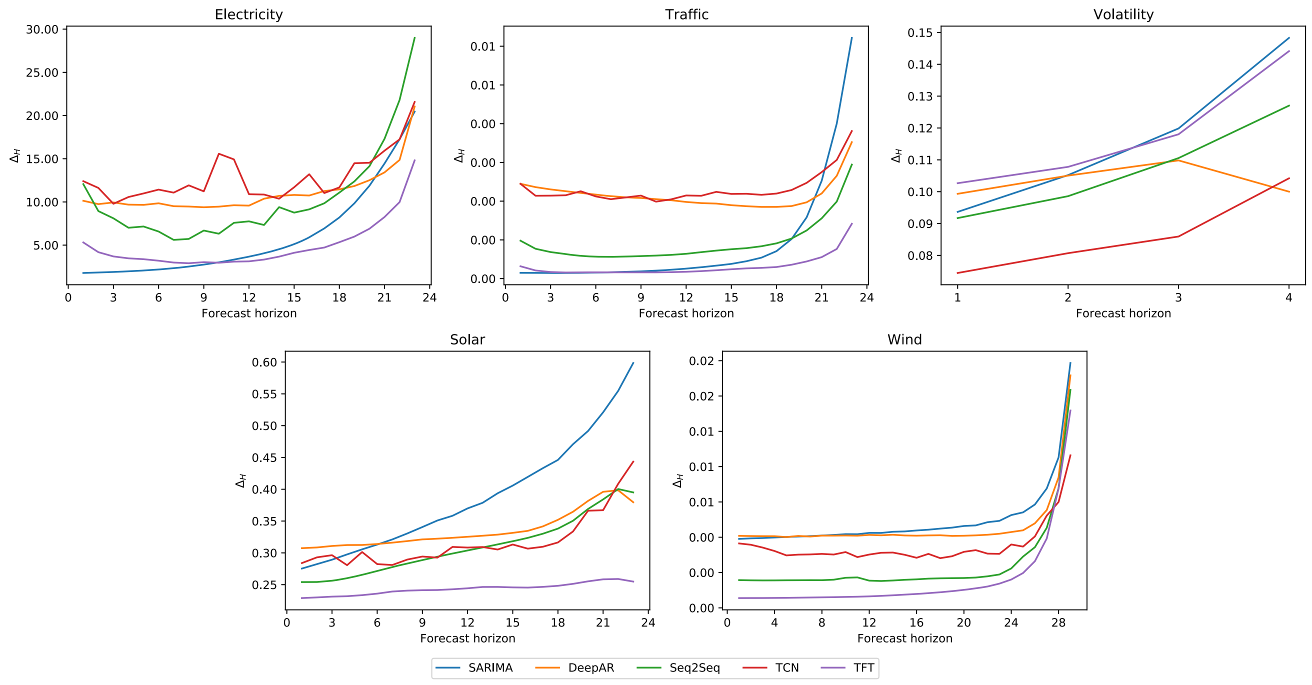

Fig. 4 Results in terms of the delta horizon metric computed per forecast horizon for all datasets

Table 8 Results in terms of decomposed errors for p_T and p_1

	Electricity		Traffic		Volatility		Solar		Wind	
	p_T	p_1	p_T	p_1	p_T	p_1	p_T	p_1	p_T	p_1
SARIMA										
- Trend	0.241	0.049	0.210	0.079	0.511	0.254	0.168	0.027	0.589	0.112
- Season	0.179	0.141	0.276	0.192	0.249	0.333	0.312	0.062	0.355	0.363
- Noise	0.190	0.154	0.258	0.218	0.440	0.474	0.440	0.124	0.634	0.561
DeepAR										
- Trend	0.321	0.059	0.156	0.059	0.554	0.331	0.153	0.010	0.281	0.119
- Season	0.211	0.185	0.225	0.171	0.257	0.310	0.318	0.059	0.400	0.358
- Noise	0.163	0.130	0.171	0.144	0.433	0.450	0.219	0.070	0.587	0.542
Seq2Seq										
- Trend	0.214	0.062	0.111	0.046	0.482	0.263	0.129	0.013	0.277	0.103
- Season	0.183	0.181	0.155	0.144	0.253	0.330	0.314	0.074	0.391	0.360
- Noise	0.175	0.138	0.162	0.142	0.438	0.460	0.248	0.073	0.589	0.543
TCN										
- Trend	0.205	0.049	0.105	0.042	0.490	0.291	0.132	0.012	0.263	0.113
- Season	0.157	0.141	0.145	0.135	0.262	0.320	0.369	0.074	0.385	0.369
- Noise	0.174	0.141	0.169	0.148	0.43	0.453	0.234	0.075	0.599	0.549
TFT										
- Trend	0.208	0.126	0.101	0.062	0.519	0.276	0.127	0.020	0.239	0.185
- Season	0.175	0.165	0.146	0.138	0.252	0.327	0.326	0.085	0.404	0.36
- Noise	0.167	0.144	0.154	0.141	0.445	0.465	0.254	0.084	0.587	0.541

Note: Noise refers to the remainder error as presented in Section 3

Similar to the win-loss metric, the drawback of Δ_H is that it can not be used in the evaluation alone. The error metric is still needed to determine which model is best in terms of average error. However, the practitioner can gain more information by applying the Δ_H and selecting the best model in terms of both error and stability. In the case of TCN and TFT, one can argue that the practitioner should use the TFT model, which produces significantly more stable forecasts although it has a 5.5% lower RMSSE compared to TCN.

5.4 Error decomposition

Lastly, we present results in terms of the decomposed error metrics as proposed in Section 3. Table 8 shows decomposed errors in terms of trend, season and remainder for the path the T-step-ahead forecasts), p_T, and for the path the 1-step-ahead forecasts, p_1. We will refer to the remainder component as noise, although it can contain other elements than white noise as it comprises everything that is not captured by the trend and seasonal components.

First, we observe that in general the trend error decreases over the forecast horizon (from p_T to p_1). Furthermore, we observe a decrease in the season error and noise error. However, the decrease is substantially lower than what we observe in the trend error. In other words, this indicates that the improvements in accuracy as the forecast horizon decreases stem primarily from improvements when predicting the correct trend. The improvements when predicting seasonality and noise tend to be minor in comparison. If we compare results on electricity and traffic (seasonal data) to volatility and wind (noisy data), we observe that the noise error is significantly higher relative to trend and season on the noisy data. Conversely, on seasonal data, the error of trend and season is higher relative to noise (when observing the p_T values). Thus, the error metrics indicate which components of each dataset are difficult to predict far ahead in time. On seasonal data, this tends to be the long-term trend, while season and noise tend to be as difficult to predict tomorrow as it is in the future. The same is true for the noisy data, however, the trend component is smaller relative to the noise component. We observe that the noise component in general rarely decreases over the horizon, which is to be expected, as true noise in theory is unpredictable.

One considerable disadvantage of the decomposed errors is interpretability between models. A prominent example of this is that TFT on electricity produces trend error 0.126, which is worse relative to the other models between 0.062 and 0.049. Yet, TFT is the second-best model in terms of aggregated RMSSE. In other words, TFT likely performs better during the other time steps between the first (step T) and last (step 1) forecasted value, which is not captured by the metrics. A solution could be to include the average of all steps in addition to the first and last steps, however, this also increases the number of metrics.

6 Conclusion and future work

In this paper, we investigate ways to provide additional insights when evaluating multi-step forecasting models. We propose four novel metrics for evaluating multi-step forecasting models, and conduct empirical experiments in order to evaluate and show the applicability of the proposed metrics.

First, we show that important information is lost when errors are aggregated. Our proposed win-loss metric captures parts of this lost information. The TCN model is deemed best on electricity in terms of MASE (0.252) and is the best model on 109 series. In contrast, TFT is second best in terms of MASE (0.260), yet it is the best model on 146 of the series in the dataset. This information would be important in industry applications, as it does not make sense to apply TCN on *all* the series even though it is the best model in terms of MASE. Furthermore, one would probably not consider SARIMA to be that good relative to TCN and TFT (MASE of 0.289). However, SARIMA is in fact the best model on 80 of the series. Thus, the best model is highly dependent on the individual time series.

Second, we find that variance-weighted errors can be useful for datasets where there are large variations over the seasonal period and where performance during uncertain periods should be weighted higher. The applicability of the metric is especially visible on solar and traffic, where Seq2Seq and DeepAR are the best models in terms of RMSSE_{VW} due to better performance where variance is high. However, we also find that the difference between models over the seasonal period tends to be minor and that models generally follow the same error pattern. In other words, we do not find that some models perform especially well during uncertain periods. Thus, the metric might only be applicable for special cases.

Third, as with the seasonal period, we find that models in general follow the same error pattern over the forecast horizon. If model X is better than model Y on average, the same conclusion tends to hold for all forecast horizons. Thus, a metric that measures performance over the forecast horizon does not seem to be useful. We do, however, find significant differences in how much each model updates its forecasted value for a point in time over the horizon. The proposed delta horizon metric allows practitioners to differentiate between models that produce stable forecasts over the forecast horizon and models that react more aggressively to arriving information.

Lastly, we provide additional insights into how the errors relate to trend, season, and noise. The decomposed error metrics indicate that improved performance over the forecast horizon is largely due to improvements in forecasting the correct trend. This is true for both seasonal and noisy datasets, however, we find the noise component to account for a larger portion of the error on noisy datasets, which agrees with intuition.

In what follows, we provide some directions for future work. One direction is to perform experiments and test the applicability of metrics on shorter time series and test series. With fewer observations, the law of large numbers might not be as prevalent considering that we have observed smooth error patterns over the seasonal period and forecast horizon.

Another possibility is to run experiments using other types of models. As we have observed, the deep learning models produce largely the same error patterns. Therefore, training the deep learning models with other loss functions could provide interesting differences, e.g., the DILATE loss function which explicitly differentiates between shape and time [50]. Furthermore, testing whether our conclusions hold for other state space models and forecasting techniques would be a possible extension of this work.

Finally, as mentioned in Section 5, it would be interesting to run experiments using variance-weighted errors for parameter optimization. We hypothesize that optimizing by variance-weighted errors would produce models that predict better during time periods where uncertainty is highest, which is typically where we want accurate forecasts. At the same time, we argue that the other metrics are not suited for this purpose for different reasons. I.e., the win-loss metric is not applicable due to being dependent on the errors of the other models, and the delta horizon metric should only be used in combination with a standard error metric.

7 Appendix A Details on experimental setup

7.1 A.1 Data preprocessing

All datasets are preprocessed by adding date-time information to each time series, i.e., date-time covariates. We closely follow the preprocessing steps applied in previous studies for each dataset. Therefore, all our results should be comparable to previous research and of relevance for future research within the field. To ensure replication of the results on electricity, traffic and volatility by [54], we use their source code made available at the Googe Research GitHub repository[12] to obtain and process the aforementioned datasets. The solar dataset is obtained at the GitHub repository[13] of [48]. Lastly, the wind dataset is obtained directly from the Kaggle source. All datasets are processed by adding the following covariates:

1. *Group id*: Each time series in the dataset is represented by a group id.
2. *Time index*: Each series has a time index representing the order of observations. This is either hours, days or 10-minute intervals from start depending on the dataset frequency.
3. *Known inputs*: The electricity, traffic and solar datasets are additionally added hour and day of week variables. To volatility and wind, we add day of week, day of month, week of year and month.

Table 2 provides an overview of the datasets, including the number of series, observations in total, frequency, seasonality and target domain. Additionally, it provides dataset specific parameters which we present in the following section on training procedure.

7.2 SARIMA rolling window

The rolling window approach requires specifying a constant lookback window, k. The ideal solution would be to select the lookback window based on a time series cross-validation approach [37], and therefore select k for each time series individually. However, this would significantly increase the computational resources required considering the size of the datasets. To keep the problem tractable, we performed preliminary experiments and found the following static lookback windows to produce the best results: $k = 240$ for electricity, $k = 720$ for traffic, $k = 756$ for volatility, $k = 240$ for solar, and $k = 300$ for wind. The search was performed in terms of including multiples of the seasonal period m in the lookback window, to ensure that the models are fitted on an adequate number of seasonal periods.

7.3 Hyperparameter tuning

Table 9 shows the selected search space for each hyperparameter and model. Note that we use the same search space for hyperparameters that are common across models for consistency. The search was conducted running 250 trials for each search, using a batch size of 64, a maximum of 25 epochs and pruning callbacks for poor trials [3]. To keep running time tractable, we subset the training and validation sets according to the following fractions, respectively: Electricity, 0.15 and 0.33. Traffic, 0.15 and 0.33. Solar, 0.10 and 0.20. Volatility, 1.00 and 1.00. Wind 1.00 and 1.00. Table 10-13 shows the optimized hyperparameters for every deep learning model on every dataset.

12 https://github.com/google-research/google-research/tree/master/tft

13 https://github.com/laiguokun/multivariate-time-series-data

Table 9 Overview of search ranges for Optuna hyperparameter optimization

	DeepAR	Seq2Seq	TCN	TFT
Learning rate	$[1 \times 10^{-6}, 1 \times 10^{-2}]$	$[1 \times 10^{-6}, 1 \times 10^{-2}]$	$[1 \times 10^{-6}, 1 \times 10^{-2}]$	$[1 \times 10^{-6}, 1 \times 10^{-2}]$
Num. layers	[2, 6]	[1, 3]	[2, 6]	-
Hydden size	[16, 256]	[16, 256]	[16, 256]	[16, 256]
Dropout	[0.1, 0.9]	[0.1, 0.9]	[0.1, 0.9]	[0.1, 0.9]
Max gradient norm.	$[1 \times 10^{-2}, 1 \times 10^{-2}]$	$[1 \times 10^{-2}, 1 \times 10^{-2}]$	$[1 \times 10^{-2}, 1 \times 10^{-2}]$	$[1 \times 10^{-2}, 1 \times 10^{-2}]$
Kernel size	–	–	[4, 16]	–
Attention head size	–	–	–	{1, 4}

Table 10 Final hyperparameters for the DeepAR model

	Electricity	Traffic	Volatility	Solar	Wind
Learning rate	1×10^{-3}	1×10^{-3}	1×10^{-5}	3×10^{-4}	4×10^{-4}
Num. layers	3	3	3	2	2
Hidden size	40	40	208	48	256
Dropout	0.1	0.1	0.3	0.3	0.9
Max gradient norm.	1×10^{-2}	1×10^{-2}	10	11	0.2
Batch size	64	64	64	128	64

Table 11 Final hyperparameters for the Seq2Seq model

	Electricity	Traffic	Volatility	Solar	Wind
Learning rate	3×10^{-3}	1×10^{-4}	3×10^{-5}	1×10^{-3}	7×10^{-3}
Num. layers	3	2	1	3	2
Hidden size	96	240	224	64	144
Dropout	0.1	0.8	0.3	0.5	0.4
Max gradient norm.	0.1	0.3	3×10^{-2}	2	20
Batch size	64	64	64	128	64

Table 12 Final hyperparameters for the TCN model

	Electricity	Traffic	Volatility	Solar	Wind
Learning rate	1×10^{-4}	8×10^{-5}	2×10^{-6}	5×10^{-4}	9×10^{-6}
Num. layers	5	4	5	4	6
Hidden size	256	224	224	176	64
Dropout	0.1	0.6	0.9	0.7	0.2
Max gradient norm.	3×10^{-2}	0.9	0.6	48	3×10^{-2}
Batch size	64	64	64	128	64
Kernel Size	4	10	10	6	12

Table 13 Final hyperparameters for the TFT model

	Electricity	Traffic	Volatility	Solar	Wind
Learning rate	1×10^{-3}	1×10^{-3}	1×10^{-2}	4×10^{-6}	6×10^{-3}
Hidden size	160	320	160	224	48
Dropout	0.1	0.3	0.3	0.2	0.7
Max gradient norm.	1×10^{-2}	1×10^{2}	1×10^{-2}	13	0.7
Batch size	64	128	64	128	64
Attention head size	4	4	1	1	1

Acknowledgements The authors would like to sincerely express their gratitude to the reviewers for their time and effort in reviewing the manuscript. Their valuable comments and suggestions have been instrumental in improving the quality of the work.

Author Contributions Eivind Strøm: Conceptualization, methodology, software, validation, and writing – original draft. Odd Erik Gundersen: Conceptualization, methodology, supervision, and writing – review and editing.

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital).

Availability of data and materials Source code will be made available on Github/Zenodo prior to publication. All dataset sources have been extensively documented in the article and are openly available on Zenodo, exactly as used in this article: https://zenodo.org/record/6970019

Code availability Original source code is available at: https://github.com/eivistr/metrics-for-multistep-forecasting-study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Adya M, Collopy F (1998) How effective are neural networks at forecasting and prediction? A review and evaluation. Journal of Forecasting, 17(5-6):481–495, 9. ISSN 0277-6693. https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<481::AID-FOR709>3.0.CO;2-Q
2. Akaike H (1998) Information Theory and an Extension of the Maximum Likelihood Principle. Selected Papers of Hirotugu Akaike. Springer, New York, New York, NY, pp 199–213
3. Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2623–2631, New York, NY, USA, 7. ACM. ISBN 9781450362016. https://doi.org/10.1145/3292500.3330701
4. An NH, Anh DT (2016) Comparison of Strategies for Multi-step-Ahead Prediction of Time Series Using Neural Network. Proceedings - 2015 international conference on advanced computing and applications. ACOMP 2015:142–149. https://doi.org/10.1109/ACOMP.2015.24
5. Armstrong JS (2001) Evaluating forecasting methods. In: Principles of forecasting, Springer, pp 443–472
6. Armstrong JS, Fildes R (1995) Correspondence on the selection of error measures for comparisons among forecasting methods. J Forecast 14(1):67–71, 1 ISSN 02776693. https://doi.org/10.1002/for.3980140106
7. Armstrong JS, Collopy F (1992) Error measures for generalizing about forecasting methods: Empirical comparisons. Int J Forecast 8(1):69–80, 6 ISSN 01692070. https://doi.org/10.1016/0169-2070(92)90008-W. https://linkinghub.elsevier.com/retrieve/pii/016920709290008W
8. Athanasopoulos G, Kourentzes N (2023) On the evaluation of hierarchical forecasts. Int J Forecast 39(4):1502–1511 ISSN 0169-2070. https://doi.org/10.1016/j.ijforecast.2022.08.003. https://www.sciencedirect.com/science/article/pii/S0169207022001121
9. Bai S, Kolter JZ, Koltun V (2018) An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv, 3 2018. ISSN 23318422. http://arxiv.org/abs/1803.01271
10. Beitner J (2020) PyTorch Forecasting: Time series forecasting with PyTorch. https://github.com/jdb78/pytorch-forecasting
11. Taieb SB, Bontempi G, Atiya AF, Sorjamaa A (2012) A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Syst Appl 39(8):7067–7083, 6. ISSN 09574174. https://doi.org/10.1016/j.eswa.2012.01.039http://dx.doi.org/10.1016/j.eswa.2012.01.039
12. Böse J-H, Flunkert V, Gasthaus J, Januschowski T, Lange D, Salinas D, Schelter S, Seeger M, Wang Y (2017) Probabilistic demand forecasting at scale. Proceedings of the VLDB Endowment, 10(12):1694–1705, 8. ISSN 2150-8097. https://doi.org/10.14778/3137765.3137775
13. Box GEP, Jenkins GM (1968) Some Recent Advances in Forecasting and Control. Appl Stat 17(2):91. ISSN 00359254. https://doi.org/10.2307/2985674. https://onlinelibrary.wiley.com/doi/10.2307/2985674

14. Bustos O, Pomares-Quimbaya A (2020) Stock market movement forecast: A Systematic review. Expert Syst Appl 156, ISSN 09574174. https://doi.org/10.1016/j.eswa.2020.113464
15. Chatfield Chris (1988) Apples, oranges and mean square error. Int J Forecast 4(4):515–518, 1. ISSN 01692070. https://doi.org/10.1016/0169-2070(88)90127-6. https://linkinghub.elsevier.com/retrieve/pii/0169207088901276
16. Chatfield Chris (1993) Neural networks: Forecasting breakthrough or passing fad? Int J Forecast 9(1), 1–3, 4. ISSN 01692070
17. Chen C, Twycross J, Garibaldi JM (2017) A new accuracy measure based on bounded relative error for time series forecasting. PLOS ONE, 12(3):3. ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0174202. https://dx.plos.org/10.1371/journal.pone.0174202
18. Choi E, Bahadori MT, Kulas JA, Schuetz A, Stewart WF, Sun J (2016) RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. Adv Neural Inform Process Syst (Nips):3512–3520:8. ISSN 10495258
19. Clements MP, Hendry DF (1993) On the limitations of comparing mean square forecast errors. J Forecast 12(8):617–637, 12. ISSN 02776693. https://doi.org/10.1002/for.3980120802. https://onlinelibrary.wiley.com/doi/10.1002/for.3980120802
20. Clements MP, Hendry DF (2001) Explaining the Results of the M3 Forecasting Competition. Int J Forecast 17:550–554. ISSN 0169-2070
21. Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990) STL: A Seasonal-Trend Decomposition Procedure Based on Loess. J Off Stat 6:3–73
22. Crone SF, Hibon M, Nikolopoulos K (2011) Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. International Journal of Forecasting, 27(3):635–660, 7. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2011.04.001
23. Dauphin YN, Fan A, Auli M, Grangier D (2016) Language Modeling with Gated Convolutional Networks. 34th International conference on machine learning, ICML 2017, 2:1551–1559, 12
24. De Gooijer JG, Hyndman RJ (2006) 25 years of time series forecasting. Int J Forecast 22(3):443–473, 1. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2006.01.001. https://linkinghub.elsevier.com/retrieve/pii/S0169207006000021
25. Devlin J, Chang M-W, Lee K, Toutanova K (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL HLT 2019 - 2019 conference of the north american chapter of the association for computational linguistics: human language technologies-proceedings of the conference, 1:4171–4186, 10
26. Petropoulos F, Koutsandreas D, Spiliotis E, Assimakopoulos V (2022) On the selection of forecasting accuracy measures. J Oper Res Soc 73(5):937–954. https://doi.org/10.1080/01605682.2021.1892464
27. Dua D, Graff C (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
28. Falcon W, Borovec J, Wälchli A, Eggert N, Schock J, Jordan J, Skafte N, Ir1dXD, Bereznyuk V, Harris E, Murrell T, Yu P, Præsius S, Addair T, Zhong J, Lipin D, Uchida S, Bapat S, Schröter H, Dayma B, Karnachev A, Kulkarni A, Komatsu S, Martin B, Schiratti J-B, Mary H, Byrne D, Cristo E, cinjon, Bakhtin A (2019) PyTorch Lightning. https://github.com/PyTorchLightning/pytorch-lightning
29. Fildes R, Ord K (2004) Forecasting Competitions: Their Role in Improving Forecasting Practice and Research. In: A companion to economic forecasting, chapter 15, pp 322–353. Wiley, Ltd,. ISBN 9780470996430. https://doi.org/10.1002/9780470996430.ch15
30. Fildes R, Hibon M, Makridakis S, Meade N (1998) Generalising about univariate forecasting methods: further empirical evidence. International Journal of Forecasting, 14(3):339–358, 9. ISSN 01692070. https://doi.org/10.1016/S0169-2070(98)00009-0
31. Fox I, Ang L, Jaiswal M, Pop-Busui R, Wiens J (2018) Deep Multi-Output Forecasting. In: Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining, pp 1387–1395, New York, NY, USA, 7 ACM. ISBN 9781450355520. https://doi.org/10.1145/3219819.3220102
32. Philip Hans Franses (2016) A note on the Mean Absolute Scaled Error. Int J Forecast 32(1):20–22, 1. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2015.03.008
33. Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN (2017) Convolutional Sequence to Sequence Learning. 34th International conference on machine learning, ICML 2017, 3:2029–2042, 5
34. Heber G, Lunde A, Shephard N, Sheppard K (2009) Oxford-Man Institute's realized library. https://realized.oxford-man.ox.ac.uk/
35. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Comput 9(8):1735–1780, 11. ISSN 0899-7667. https://doi.org/10.1162/neco.1997.9.8.1735. https://direct.mit.edu/neco/article/9/8/1735-1780/6109
36. Hyndman R, Koehler AB, Ord JK, Snyder RD (2008) Forecasting with Exponential Smoothing: The State Space Approach. Springer Series in Statistics, Springer, Berlin Heidelberg. 9783540719182
37. Hyndman R, Athanasopoulos G (2021) Forecasting: Principles and Practice. OTexts, Australia, 3rd edn
38. Hyndman RJ (2020)A brief history of forecasting competitions. Int J Forecast, 36(1):7–14, 1. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2019.03.015
39. Hyndman RJ, Khandakar Y (2008) Automatic Time Series Forecasting: The forecast Package for R. J Stat Softw, 27(3):22. ISSN 1548-7660. https://doi.org/10.18637/jss.v027.i03
40. Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. Int J Forecast, 22(4):679–688. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2006.03.001
41. Inman RH, Pedro HTC, Coimbra CFM (2013) Solar forecasting methods for renewable energy integration. Prog Energy Combust Sci, 39(6):535–576,12. ISSN 03601285. https://doi.org/10.1016/j.pecs.2013.06.002
42. Kang Y, Hyndman RJ, Smith-Miles K (2017) Visualising forecasting algorithm performance using time series instance spaces. Int J Forecast, 33(2):345–358, 4. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2016.09.004
43. Kolassa S (2016) Evaluating predictive count data distributions in retail sales forecasting. Int J Forecast, 32(3):788–803, 7. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2015.12.004
44. Kolassa S (2020) Why the "best" point forecast depends on the error or accuracy measure. Int J Forecast, 36(1):208–211, 1. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2019.02.017
45. Kolassa S (2023) Do we want coherent hierarchical forecasts, or minimal MAPEs or MAEs? (We won't get both!). Int J Forecast, 39 (4):1512–1517. ISSN 0169-2070. https://doi.org/10.1016/j.ijforecast.2022.11.006. https://www.sciencedirect.com/science/article/pii/S0169207022001492
46. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems, vol 25. Curran Associates, Inc
47. Lai G, Chang W-C, Yang Y, Liu H (2018a) Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, number July, pages 95–104, New York, NY, USA, 6. ACM. ISBN 9781450356572. https://doi.org/10.1145/3209978.3210006. https://dl.acm.org/doi/10.1145/3209978.3210006
48. Lai ZR, Dai DQ, Ren CX, Huang KK (2018) A peak price tracking-based learning system for portfolio selection. IEEE Trans Neural Netw Learn Syst, 29(7):2823–2832. ISSN 21622388. https://doi.org/10.1109/TNNLS.2017.2705658

49. Laptev N, Yosinski J, Li LE, Smyl S (2017) Time-series Extreme Event Forecasting with Neural Networks at Uber. In: International conference on machine learning (ICML), pp 1–5
50. Le Guen V, Thome N (2019) Shape and Time Distortion Loss for Training Deep Time Series Forecasting Models. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd, Fox E, Garnett R (eds), Adv Neural Inf Process Syst, vol 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/466accbac9a66b805ba50e42ad715740-Paper.pdf
51. Lei M, Shiyan L, Chuanwen J, Hongling L, Yan Z (2009) A review on the forecasting of wind speed and generated power. Renew Sust Energ Rev, 13(4):915–920, 5. ISSN 13640321. https://doi.org/10.1016/j.rser.2008.02.002. https://linkinghub.elsevier.com/retrieve/pii/S1364032108000282
52. Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang YX, Yan X (2019) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Adv Neural Inf Process Syst, 32(NeurIPS). ISSN 10495258
53. Lim B, Zohren S (2021) Time-series forecasting with deep learning: a survey. Philosophical transactions of the royal society a: mathematical, physical and engineering sciences 379(2194):20200209
54. Lim B, Arık SÖ, Loeff N, Pfister T (2021) Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. Int J Forecast, 37(4):1748–1764. ISSN 0169-2070. https://doi.org/10.1016/j.ijforecast.2021.03.012. https://www.sciencedirect.com/science/article/pii/S0169207021000637
55. Makridakis S, Andersen A, Carbone R, Fildes R, Hibon M, Lewandowski R, Newton J, Parzen E, Winkler R (1982) The accuracy of extrapolation (time series) methods: Results of a forecasting competition. J Forecast 1(2):111–153. https://doi.org/10.1002/for.3980010202
56. Makridakis S, Spiliotis E, Assimakopoulos V (2020) The M5 accuracy competition: Results, findings and conclusions. Int J Forecast
57. Makridakis S (1993) Accuracy measures: theoretical and practical concerns. Int J Forecast, 9(4):527–529, 12. ISSN 01692070. https://doi.org/10.1016/0169-2070(93)90079-3. https://linkinghub.elsevier.com/retrieve/pii/0169207093900793
58. Makridakis S, Hibon M (2000) The M3-Competition: results, conclusions and implications. Int J Forecast, 16(4):451–476, 10. ISSN 01692070. https://doi.org/10.1016/S0169-2070(00)00057-1. https://linkinghub.elsevier.com/retrieve/pii/S0169207000000571
59. Makridakis S, Spiliotis E, Assimakopoulos V (2018) Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLOS ONE, 13(3), 3. ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0194889. https://dx.plos.org/10.1371/journal.pone.0194889
60. Mudelsee M (2019) Trend analysis of climate time series: A review of methods. Earth Sci Rev, 190(December 2018):310–322. ISSN 00128252. https://doi.org/10.1016/j.earscirev.2018.12.005
61. Parmezan ARS, Souza VMA, Batista GEAPA (2019) Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. Inf Sci, 484:302–337, 5. ISSN 00200255. https://doi.org/10.1016/j.ins.2019.01.076
62. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in neural information processing systems 32, pp 8024–8035. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
63. Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell GW (2017) A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI-17, pp 2627–2633. https://doi.org/10.24963/ijcai.2017/366
64. Rangapuram SS, Seeger MW, Gasthaus J, Stella L, Wang Y, Januschowski T (2018) Deep State Space Models for Time Series Forecasting. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Adv Neural Inf Proc Syst, vol 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf
65. Salinas D, Flunkert V, Gasthaus J (2017) DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. Int J Forecast, 36(3):1181–1191, 4. ISSN 01692070. https://doi.org/10.1016/j.ijforecast.2019.07.001. https://linkinghub.elsevier.com/retrieve/pii/S0169207019301888
66. Schwertman NC, Gilks AJ, Cameron J (1990) A Simple Noncalculus Proof That the Median Minimizes the Sum of the Absolute Deviations. Am Stat 44(1):38–39. https://doi.org/10.1080/00031305.1990.10475690
67. Seeger MW, Salinas D, Flunkert V (2016) Bayesian Intermittent Demand Forecasting for Large Inventories. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R (eds), Advances in neural information processing systems, vol 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/03255088ed63354a54e0e5ed957e9008-Paper.pdf
68. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, 1. ISSN 0028-0836. https://doi.org/10.1038/nature16961
69. Själander M, Jahre M, Tufte G, Reissmann N (2019) EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure
70. Sutskever I, Vinyals O, Le QV (2014) Sequence to Sequence Learning with Neural Networks. Adv Neural Inform Process Syst 4(January):3104–3112, 9. ISSN 10495258
71. Taieb SB, Atiya AmirF (2016) A Bias and Variance Analysis for Multistep-Ahead Time Series Forecasting. IEEE Trans Neural Netw Learn Syst 27(1):62–76, 1. ISSN 2162-237X. https://doi.org/10.1109/TNNLS.2015.2411629. http://ieeexplore.ieee.org/document/7064712/
72. Topol EJ (2019) High-performance medicine: the convergence of human and artificial intelligence. Nature Medicine, 25(1):44–56, 2019. ISSN 1546170X. https://doi.org/10.1038/s41591-018-0300-7
73. Vallance L, Charbonnier B, Paul N, Dubost S, Blanc P (2017) Towards a standardized procedure to assess solar forecast accuracy: A new ramp and time alignment metric. Solar Energy 150:408–422, 7. ISSN 0038092X. https://doi.org/10.1016/j.solener.2017.04.064https://linkinghub.elsevier.com/retrieve/pii/S0038092X17303687
74. Van Belle J, Crevits R, Verbeke W (2023) Improving forecast stability using deep learning. International Journal of Forecasting, 39(3):1333–1350. ISSN 0169-2070. https://doi.org/10.1016/j.ijforecast.2022.06.007https://www.sciencedirect.com/science/article/pii/S016920702200098X
75. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is All you Need. In: Guyon I, Von Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds), Advances in neural information processing systems, vol 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
76. Vlahogianni EI, Karlaftis MG, Golias JC (2014) Short-term traffic forecasting: Where we are and where we're going. Trans Res Part

C: merg Technol 43:3–19, 6. ISSN 0968090X. https://doi.org/10.1016/j.trc.2014.01.005
77. Wang Y, Smola A, Maddix D, Gasthaus J, Foster D, Januschowski T (2019) Deep Factors for Forecasting. In: Chaudhuri K, Salakhutdinov R (eds), Proceedings of the 36th international conference on machine learning, vol 97 of Proceedings of machine learning research, pp 6607–6617. PMLR, 4. https://proceedings.mlr.press/v97/wang19k.html
78. Ward JA, Lukowicz P, Gellersen HW (2011) Performance metrics for activity recognition. ACM Trans Intell Syst Technol 2(1):1–23, 1. ISSN 2157-6904. https://doi.org/10.1145/1889681.1889687https://dl.acm.org/doi/10.1145/1889681.1889687
79. Wen R, Torkkola K, Narayanaswamy B, Madeka D (2017) A Multi-Horizon Quantile Recurrent Forecaster. NIPS, 11. ISSN 23318422
80. Young T, Hazarika D, Poria S, Cambria E (2017) Recent Trends in Deep Learning Based Natural Language Processing. IEEE Computational Intelligence Magazine, 13(3):55–75, 8. ISSN 1556-603X. https://doi.org/10.1109/MCI.2018.2840738
81. Yu HF, Rao N, Dhillon IS (2016) Temporal regularized matrix factorization for high-dimensional time series prediction. In: Advances in neural information processing systems, pp 847–855
82. Zhang J, Florita A, Hodge B-M, Lu S, Hamann HF, Banunarayanan V, Brockway AM (2015) A suite of metrics for assessing the performance of solar power forecasting. Solar Energy, 111:157–175, 1. ISSN 0038092X. https://doi.org/10.1016/j.solener.2014.10.016
83. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In: The Thirty-fifth AAAI conference on artificial intelligence, AAAI 2021, virtual conference, AAAI Press, vol 35, pp 11106–11115

Eivind Strøm is a Senior Data Scientist at the Tech Build and Design Unit of Boston Consulting Group, BCGX. His work primarily focuses on machine learning, forecasting, and optimization. Strøm holds two MSc degrees from the Norwegian University of Science and Technology (NTNU), completed in 2020 and 2021.

Odd Erik Gundersen is an adjunct associate professor at NTNU and the Chief AI Officer at Aneo AS. Gundersen received both his MSc (2004) and PhD (2014) from NTNU. He has applied AI in industry since 2006, working mostly with time-series data. Currently, he is investigating how AI can be applied in the renewable energy sector and for driver education.