Traditional vs. Machine-learning methods for forecasting sandy shoreline evolution using historic satellite-derived shorelines

Calkoen, Floris; Luijendijk, Arjen; Rivero, Cristian Rodriguez; Kras, Etienne; Baart, Fedor

DOI
10.3390/rs13050934

Publication date
2021

Document Version
Final published version

Published in
Remote Sensing

Citation (APA)
Calkoen, F., Luijendijk, A., Rivero, C. R., Kras, E., & Baart, F. (2021). Traditional vs. Machine-learning methods for forecasting sandy shoreline evolution using historic satellite-derived shorelines. *Remote Sensing*, *13*(5), 1-21. Article 934. https://doi.org/10.3390/rs13050934

remote sensing

MDPI

Article

Traditional vs. Machine-Learning Methods for Forecasting Sandy Shoreline Evolution Using Historic Satellite-Derived Shorelines

Floris Calkoen [1,2,*], Arjen Luijendijk [1,3], Cristian Rodriguez Rivero [2], Etienne Kras [1] and Fedor Baart [1]

[1] Deltares, Boussinesqweg 1, 2629 HV Delft, The Netherlands; arjen.luijendijk@deltares.nl (A.L.); etienne.kras@deltares.nl (E.K.); fedor.baart@deltares.nl (F.B.)
[2] Faculty of Science, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands; c.m.rodriguezrivero@uva.nl
[3] Department of Hydraulic Engineering, Faculty of Civil Engineering and Geosciences, Delft University of Technology, P.O. Box 5048, 2600 GA Delft, The Netherlands
* Correspondence: floris@calkoen.nl

Abstract: Forecasting shoreline evolution for sandy coasts is important for sustainable coastal management, given the present-day increasing anthropogenic pressures and a changing future climate. Here, we evaluate eight different time-series forecasting methods for predicting future shorelines derived from historic satellite-derived shorelines. Analyzing more than 37,000 transects around the globe, we find that traditional forecast methods altogether with some of the evaluated probabilistic Machine Learning (ML) time-series forecast algorithms, outperform Ordinary Least Squares (OLS) predictions for the majority of the sites. When forecasting seven years ahead, we find that these algorithms generate better predictions than OLS for 54% of the transect sites, producing forecasts with, on average, 29% smaller Mean Squared Error (MSE). Importantly, this advantage is shown to exist over all considered forecast horizons, i.e., from 1 up to 11 years. Although the ML algorithms do not produce significantly better predictions than traditional time-series forecast methods, some proved to be significantly more efficient in terms of computation time. We further provide insight in how these ML algorithms can be improved so that they can be expected to outperform not only OLS regression, but also the traditional time-series forecast methods. These forecasting algorithms can be used by coastal engineers, managers, and scientists to generate future shoreline prediction at a global level and derive conclusions thereof.

Keywords: forecasting shorelines; shoreline-trajectories; sandy beaches; coastal engineering; coastal oceanography; time-series forecasting; data science; machine learning; deep learning

Citation: Calkoen, F.; Luijendijk, A.; Rivero, C.R.; Kras, E.; Baart, F. Traditional vs. Machine-Learning Methods for Forecasting Sandy Shoreline Evolution Using Historic Satellite-Derived Shorelines. *Remote Sens.* **2021**, *13*, 934. https://doi.org/10.3390/rs13050934

Academic Editors: Jorge Vazquez, Rafael Almar, Dennis Wilson and Jean-Marc Delvit

Received: 30 December 2020
Accepted: 25 February 2021
Published: 3 March 2021

1. Introduction

Sandy beaches form an essential part of coastal zones as they play a key role in the ecosystem, while providing socio-economic values and services at the same time. Despite the ecological and social importance of these ecosystems, sandy beaches are increasingly under anthropogenic pressure. Sea levels have already risen throughout the twentieth century and are expected to rise increasingly this century, effectively constituting another serious threat to stability of coastal ecosystems [1]. Shoreline retreat can particularly be harmful to coastal communities, natural preserves and infrastructure. It is, therefore, important that effective management of sandy shores includes sustainable multiple use that does not comprise the future [2]. Effective management further requires optimal long-term sustainable use of the sandy coasts and maintenance of the most natural environment possible [2]. To meet these demands, coastal managers have an increasing need for accurate shoreline predictions, which empowers them to assess vulnerability and protect coastal infrastructure, human safety and habitats [3].

Traditionally, dynamic coastal zones have been hard to monitor because monitoring involved time-consuming field work and/or expensive imagery stations at multiple loca-

tions. However, at present these disadvantages can to some extent be overcome by using Earth-observing satellite data. During recent years, remote sensing data have played an increasingly important role in achieving data-driven understanding of beach dynamics on various time- and spatial scales [4–7].

Typically, shoreline evolution is predicted at local level by process-driven models, e.g., [3,8,9]. Although these models have provided a step from accidental to rather intentional coastal engineering they are relatively labor intensive and computationally expensive—especially when such studies aim to cover mesoscale or larger areas. In a blind-tested comparison of shoreline prediction models, ref. [10] considered ML methods, but note that in general the models often fail to produce reliable forecasts outside the in-sample window. At longer time-scales ambient shoreline change-rates present in Satellite-derived Shorelines (SDS) data have been used to assess which shorelines are under extinction threat [11]. However, extrapolating OLS regression through the SDS to forecast 2100 shorelines might be an unreasonable simplification of shoreline dynamics. Southgate [12] already showed other time-series modeling methods are substantially superior to OLS regression through a signal of shoreline-positions, although his study was limited to forecasting five years ahead.

In this study, we take advantage of the extensive amount of data, computer resources and a wide variety of time-series forecasting models that nowadays exist. We compare eight time-series modeling approaches to forecast shoreline evolution from SDS. The results show that most classical and ML methods are able to outperform OLS baselines and that this advantage is steady over increasing forecasting horizon. To the best of our knowledge this is the first time that shoreline evolution forecasting is approached by learning patterns across a collection of SDS time-series with ML techniques. To illustrate the potential of these methods we present some probabilistic shoreline evolution forecasts from 2016 until 2030. We also provide insight to how ML models can be further improved so that they can also outperform classical time-series forecast methods.

1.1. Monitoring Satellite-Derived Shoreline Changes

Although the shoreline is strictly defined as the intersection of water and land surfaces, the dynamic nature of this boundary and its dependence on the temporal and spatial scale at which it is being considered, results to use a range of shoreline indicators [13]. In this study, we use an indicator based on image-processing techniques that has been one of the key factors that led to several coastal monitoring applications [5]; some of them directly targeting shoreline monitoring [4,6,7]. All these applications relied on supervised classification of composite satellite imagery, usually by means of some computationally cheap decision tree algorithm. They are able to provide excellent results in satellite imagery classification, especially when spectral indices are used as input features. Water and wet surfaces characteristically show strong absorption in the near-infrared and short-wave infrared spectrum, while dry soils, dry sand and vegetation reflect a large proportion of the radiation in these spectral ranges. Therefore, these spectral bands and indices derived thereof, have a high potential for measuring moisture levels in soils and vegetation and are now widely used indicators for water body delineation and wetland mapping. Derivation of the SDS used in this study relied especially on the Normalized Difference Water Index (NDWI).

The quality of (historical) shoreline change studies is proportional to the temporal and spatial resolution of the satellites providing data. With advances in remote sensing technology over the last decades both temporal and spatial resolution have steadily increased [14]. The long running record of the Landsat program and its global coverage make the data exceptionally useful for large scale studies on multi-annual to multi-decadal time frame, especially since it has been freely available from 2008 onward [15]. The spatial resolution of the sensors used in the Landsat program has generally been 30-m. Hagenaars et al. [16] found that the accuracy of SDS detection is within sub-pixel precision (15-m) for composite imagery.

1.2. Forecasting Time-Series of Shoreline Positions

Historical shoreline positions, albeit being derived from different types of shoreline indicators, have a long history for being used to determine erosion rates and derive shoreline predictions thereof (e.g., [11,17–20]). Typically, erosion rates are extracted by means of a least-squares regression and extrapolated into the future. Although several statistical methods of higher complexity have been considered to predict future shorelines from a collection of historical shoreline positions—such as binning [20], basis functions [21,22] and time-series forecast methods [12,23]—OLS presently still forms the (scientific) basis of Dutch coastal management [23] and long-term coastal erosion rates in the US [24]. However, whereas many naturally evolving shorelines can follow a linear trend for a certain period, some shorelines are influenced by processes where a straight trend line does not capture the relevant processes [25]. Examples include seasonality, storm induced retreat, sand waves, coastal interventions, accelerated coastal retreat due to gas or groundwater extraction below the coastal zone and accelerated Sea-level Rise (SLR). In such circumstances other forecast algorithms might be more suitable as they are able to take these processes into account.

Traditionally time-series forecasting has been dominated by statistical methods (e.g., [26]) such as Error Trend Seasonality (ETS) and Autoregressive Integrateded Moving Average (ARIMA) because they are relatively simple, well understood, robust and effective on many problems, so that they can be used by non-expert users [27,28]. Whereas ETS models focus on describing error, trend and seasonality in data, ARIMA models aim at the auto-correlations present in the time series. Since these parameters are determined at individual time-series level, forecasting larger quantities of sites easily becomes too laborious. Hyndman and Khandakar [26] specifically addressed this issue and introduced a framework (`RForecast`) to automate ETS and ARIMA forecasts.

Until recently, simple forecasting methods typically outperformed more complex ones in time-series forecasting [29,30]. Nevertheless the impressive results of deep learning in computer vision and language-related tasks probably triggered a shift in time series modeling from classical statistical approaches to Deep Learning (DL) based methods. During the 2018 M4 forecasting competition numerous participants relied partly on ML [31] and it was the first time in its competition history [32] that it was won by a model using ML techniques [33]. Besides the M4 forecasting competition there have been various other forecasting accuracy improvements via the usage of DL models (e.g., [34–38]) during recent years.

Whereas most classical time-series forecast methods such as ETS and ARIMA are local models for which the free parameters are estimated individually per time-series, ML models typically learn the free parameters jointly over a collection of time-series. Therefore ML models are able to extract patterns from irregular collections of time series which cannot be distinguished at individual level. Recent studies [34,36–39] have demonstrated that Neural Networks (NNs) are particularly well suited to take advantage of a dataset which consists of large amounts of time-series. Moreover, whereas classical models are typically limited to univariate inputs, NNs are able to use multiple inputs, and therefore these types of models can be enhanced by (additional) explanatory variables. Finally, cross-series learning enables cold-start forecasting [37] which can be used for transects with a limited amount of observations.

1.3. Research Objective

The main aim of this study is to find out which time-series forecast algorithms are most suitable for long-term shoreline evolution predictions. More specifically:

- To compare eight time-series forecast algorithms, for forecasting multi-annual shoreline evolution using SDS, by means of commonly used error measures;
- To assess the reliability of these algorithms over increasing forecast horizons;
- To provide insight into learning across time-series of SDS using ML forecast algorithms;

- To illustrate the potential of conditional probabilistic time-series ML algorithms for long-term shoreline evolution predictions.

2. Materials and Methods

2.1. Data

The ShorelineMonitor data [4] consists of SDS, at roughly 2.2-million globally defined 500-m alongshore spaced transects, from 1984 until 2016. This dataset also contains metadata for each transect, including geospatial coordinates, categorical features, soil type (sandy vs. non sandy), and a proxy for the change rate uncertainty of the time-series [4]. After excluding approximately 400-thousand noisy transects in the polar latitudes [40] we selected the transects flagged [4] to be located at sandy beaches, with low shoreline detection errors and low errors in change rate or timespan. Subsequently, all detected outliers [4,40] were removed. Finally, we selected the transects without missing values. Thus, we obtained a dataset consisting of 37,111 time-series, with one observation per year, based on composite satellite imagery from Luijendijk et al. [4], for the period 1984–2016. Figure 1a shows all included time-series as a density line plot. The time-series were annually averaged and are therefore equally spaced along the time axis. We excluded all transects with one or more missing annually averaged observations since classical models ETS and ARIMA are not able to deal with missing values and interpolating these missing values requires several arbitrary decisions beyond the scope of this study. Thus, all time-series are equally spaced and aligned, i.e., the starting point $t = 1$ equivalents for all series to 1 January 1984 and the length is for all series 33 years with annual observation frequency (Figure 1).

The ShorelineMonitor data includes metadata, such as the change-rate uncertainty, that most ML methods can use directly, as a static real covariates, to condition their predictions. In addition, we created two more static categorical covariates. With hierarchical clustering algorithms `DBSCAN` [41] and `OPTICS` [42] the transects were clustered, by their geospatial coordinates, into respectively 13,331 and 8938 groups of density-connected components with a maximum distance (1-km) between the clusters. Figure 1b shows a cluster of time-series in the area of Duck, North Carolina, USA. The figure shows that, at least for some clusters, SDS series have a similar structure over time.

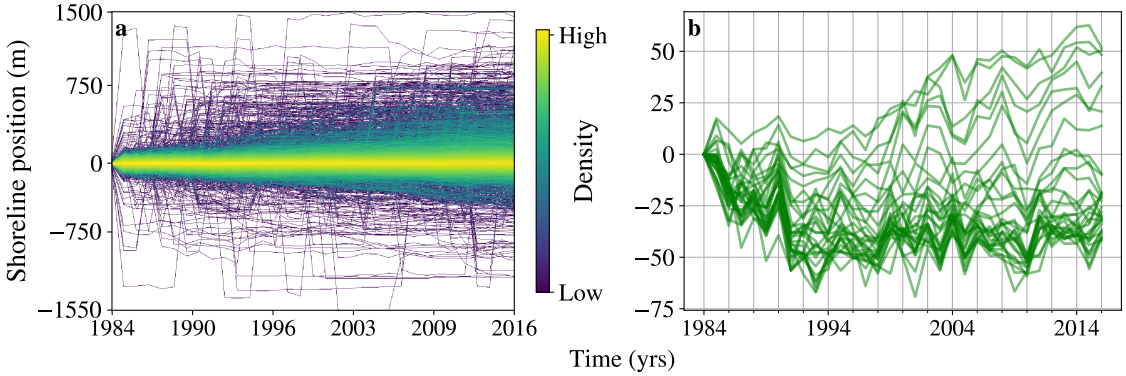

Figure 1. (**a**) Density line plot of all time-series ($N = 37,111$) included in the forecast evaluation. (**b**) The transects are clustered into density-connected groups with a maximum dinstance between the groups using hierarchical cluster algorithms. Here, all SDS time-series present in a group of transects (clustered by `DBSCAN` algorithm), in the area of Duck, USA, are shown. The transects in this cluster are density-connected within 1-km distance.

Figure 2 shows the spatial spread of the transects included in the study. Although sites across the whole world are included, certain areas, such as Europe, are under-represented. This spatially unequal cover reflects the frequency of earth-observations by the Landsat program [43], the relatively high cloud cover in certain areas [44] and the unequal distribution of sandy coasts across the world [4].

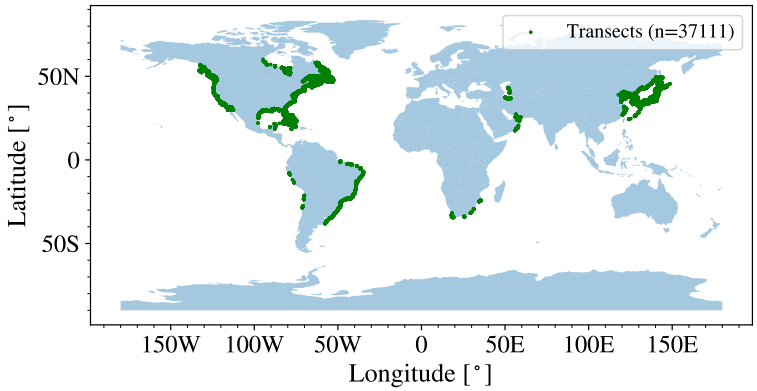

Figure 2. Transect sites with SDS time-series included in the forecast evaluation analysis.

2.2. Forecasting Objective

In total we consider eight forecasting algorithms and use a dataset consisting of 37,111 time-series. Let N be the set of univariate time-series $\{z_{i,1:T}\}_{i=1}^{N}$, where $z_{i,1:T} = (z_{i,1}, z_{i,2}, \ldots, z_{i,T})$ and $z_{i,t} \in \mathbb{R}$ denotes the value of the i-th transect at year t, counting from 1984. Each algorithm f is provided with the same task: forecast future shoreline position $z \in \mathbb{R}^H$ given a set of historical shoreline positions $z \in \mathbb{R}^P$ (Equation (1)), where P refers to past observations and H represents the forecast horizon.

$$f : \mathbb{R}^P \rightarrow \mathbb{R}^H \tag{1}$$

In each time-series, the latter H years of observations are reserved to assess forecast skill, by means of various Key Performance Indicators (KPIs). The time-series are split into a train $[z_{i,t_1}, z_{i,t_2}, \ldots, z_{i,t_{0-1}}]$ and test partition $[z_{i,t_0}, z_{i,t_{0+1}}, \ldots, z_{i,t_{0+H}}]$, where t_0 denotes the time point from which we assume $z_{i,t}$ to be unknown (Figure 3). In our evaluation experiments, the models models consume up to 26 years of observations to generate a seven year forecast. Forecast accuracy is then assessed by comparing this forecast to the observed values in the test partition, which have remained hidden to the model.

Most DL algorithms are able to consider multivariate inputs. In our experiments each transect has an associated covariate vector consisting of three static features: the change-rate proxy, and the two cluster IDs (Section 2.1). Let $X = \{\mathbf{x}_{i,1:T+H}\}_{i=1}^{N}$ be this set of covariate vectors associated to the time-series, with $\mathbf{x}_{i,t} \in \mathbb{R}^D$. Please note that the covariates are also provided to the model over the forecast horizon.

Finally, each type of model is conditioned by its own parameters Φ. For state space models these are for instance the latent states, while for NNs they represent the network weights. Thus, the point forecast objective can be stated as:

$$f(\mathbf{x}_{i,t}; \Phi) : \mathbb{R}^{T_{0-1}} \rightarrow \mathbb{R}^{T_{0+H}} \tag{2}$$

Most of the considered time-series models generate probabilistic forecasts instead of point forecasts. Whereas a point forecast compromises one expected value per future time step, a probabilistic forecast describes the expected values in terms of a certain distribution. The probabilistic forecast objective is therefore to predict the probability distribution in the prediction range $[z_{i,t_0}, z_{i,t_{0+1}}, \ldots, z_{t_{0+H}}]$ given observations from the conditioning range $[z_{i,t_1}, z_{i,t_2}, \ldots, z_{i,t_{0-1}}]$, their associated covariates $[x_{i,1}, x_{i,2}, \ldots, x_{i,t_{0+H}}]$ and model parameters Φ:

$$p(z_{i,t_0:t_{0+H}} | z_{i,1:t_{0-1}}, \mathbf{x}_{i,1:t_{0+H}}; \Phi) \tag{3}$$

2.3. Forecasting Models

Our intention was to use a wide range of forecast algorithms. We categorize forecasting approaches into two subclasses: classical statistical forecast methods and data-driven ML algorithms. Although such categorization is in theory problematic and inaccurate, as it does not respect many of the subtle characteristics of existing forecast methods, we will use this intuitive distinction for practical purposes. From the eight algorithms evaluated in this study three, OLS, ETS, ARIMA, can be considered statistical whereas the remaining five, SimpleFFN, SimpleLSTM, DeepAR, MQCNN, DeepSSM belong to the ML domain. The collection of used forecast approaches does not include all available methods; for example, Season Trend Loess (STL) could be added to the statistical approaches whereas DeepFactor would be another interesting ML candidate. Nevertheless we are convinced that the methods used in this study provide a broad overview for possible pathways in data-driven time-series-based shoreline modeling.

The ML models used in this study are all based on (combinations of) NNs. There exists a wide variety of NNs, such as Multi-layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTM) NNs. RNNs especially suggest to be a good fit for time-series forecasting. These are networks with loops in them, allowing information to persist. However, RNNs often fail to model long-term dependencies as they suffer from the vanishing gradients problem [45]. LSTMs are specifically designed to surpass this problem [46]. The hidden units of these networks also learn some kind of feature representations of the raw input, but here dynamic memory is provided by feeding hidden units back to themselves in each time-step [47]. As a detailed description of NNs is beyond the scope of this study, we refer the interested reader to [48] for a more extensive discussion on the topic.

We approach the forecasting problem as a supervised learning problem by setting a model structure in advance and learning the model parameters using a statistical optimization method. For local models ETS and ARIMA the parameters were optimized per time-series. OLS does not require any parameters and only involves least-squares optimization. With the ML approaches at least some parameters are learned jointly over the collection of time-series. Some ML models are designed to generate one-step ahead forecasts, while others directly produce multi-step predictions. Nevertheless one-step forecasting models can be used to generate multi-step forecasts by recursively feeding the model with its own one-step ahead estimate. However, feeding the network with its own predictions results in error propagation through the network. In other architectures, such as in sequence-to-sequence models, error propagation is avoided by setting the decoder to output all future target values at once so that there is no need to unroll over the forecast horizon. Future shorelines can be estimated directly as point forecasts or modeled as probability distributions. Apart from OLS and SimpleLSTM all models generate probabilistic forecasts, i.e., they predict a conditional distribution or density of future shorelines. In a well calibrated probabilistic model the observation is indistinguishable from a random draw from the predictive distribution.

2.3.1. Ordinary Least Squares

OLS was fitted, per time-series, on training observations by optimizing least squares error. To generate future predictions the regression line is extrapolated over the forecast horizon. Thus, future expected values $\hat{z}_{i,t_{0+1}:T_{0+H}}$ are equivalent to $\beta_0 + \beta_1 z_{i,t_{0+1}:t_{0+H}}$, where coefficients β_0 and β_1 refer respectively to the intercept and slope.

2.3.2. Error Trend Seasonality

Exponential Smoothing (ES) methods describe time-series with weighted averages of trend and seasonality. The set of ES models is constituted by 5 different input types for the trend component and 3 input types for the seasonal components, thus, in total there exist 15 ES methods [49]. By also considering an error component the underlying state space models are able to generate forecasts including prediction intervals [50]. In this study, we

implement ETS probabilistic forecasts through the `RForecast` package [26]. The models are fit individually per time-series, i.e., model parameters are learned for each time-series independently. The error, trend and seasonality component are left default so that the package will automatically select the optimal model from the set of model candidates by Aikake Information Criterion (AIC). Forecasts are then generated using the best model, with optimized parameters, for the forecast horizon. The medians of these ETS probabilistic forecast density are equivalent to the ES point forecast.

2.3.3. Autoregressive Integrated Moving Average

Whereas ES-models are describing the trend and seasonality in data, ARIMA models focus on autocorrelations in the series [51]. The model requires three parameters: p, the order of the autoregressive part; d, the degree of first differencing; and, q, the order of the moving average part. Similar to ETS, ARIMA models are estimated automatically per time-series with the `RForecast` package [26]. The forecast package is estimating the ARIMA model, i.e., the parameters p, d, q, by a combination of maximum likelihood estimation, unit root tests and minimization of the AIC criterion [51]. The technique finds values for the parameters which maximize the probability of obtaining the data that we have observed. The model is able to generate forecasts by replacing future observations with their forecasts, in iterative fashion over the forecast horizon. The prediction intervals for ARIMA models are based on assumptions that the residuals are uncorrelated and normally distributed [51].

2.3.4. Simple Long-Short-Term-Memory

The sequence-to-sequence architecture of the SimpleLSTM model consists of a fully connected NN with 1 layer of LSTM cells. The model contains an encoder-decoder framework, which allows the LSTM to map an input sequence to an output sequence of different length. The network receives the past vector (26) of a series as input, which equivalents to the maximum context length (26), i.e., the length of the series (33)—the forecast horizon (7). The LSTM cell transforms this vector to its hidden size (64). Subsequently, the LSTM output (64) is mapped using a linear sequential layer to a future vector (7). The SimpleLSTM model is fed with pure time-series data, i.e., no other explanatory variables are provided. The weights are updated using the `Torch Adam optimizer` [52] with MSE loss criterion calculated per batch of time-series. The models weights were initialized by setting them to 0. The learning rate was 0.01. The model was trained in batches of 50 time-series and run for 10 epochs. The SimpleLSTM model was trained and evaluated in slightly different fashion as the other forecasting models. Instead of separating train, validation and test observations within each series (Figure 3), this model works with train, validation and test series, i.e., the dataset (n = 37,111) was split into train, validation and test partitions at time-series level. Thus, the model was trained, validated and tested respectively on 23,750, 5938 and 7423 time-series.

2.3.5. Simple Feedforward Network

The SimpleFFN [53] is a MLP NN that predicts the next target time-steps given the previous ones. It takes as input a window of context length and produces an output dimension prediction length. The model has 40 hidden nodes per layer. Weights are optimized by calculating the mean of the L1 loss.

2.3.6. Deep Autoregression

DeepAR [34] is a auto-regressive RNN time-series model that learns a global model from historical data of all time-series. Importantly DeepAR not only takes he last target value as input, but also several lagged values that are relevant for the given frequency. Thus, the model is autoregressive, in so far that it considers the observation at the last time step $z_{i,t-1}$; and, recurrent because the hidden units are feeding their output back to themselves in each time-step, which can be both LSTM or Gated Recurrent Unit (GRU)

cells. It estimates parameters of a parametric distribution or uses a parameterization of the quantile function [53].

2.3.7. Multi-Quantile Convolutional Neural Network

MQCNNis based on a sequence-to-sequence network that is capable of directly generate multi-horizon quantile forecasts [37]. In this model the sequence-to-sequence MLP decoder is set to output multiple quantile values per time step in the forecast horizon and trained using the quantile loss function [37]. These quantile decoder models were combined with RNN and Dilated causal convolution encoders (CNNs) to create RNN-QR and CNN-QR models [53]. MQCNN is a type of CNN-QR model. More, specifically in our experiments we used a hierarchical causal convolution 1-D encoder and a dense layer to compute the projection weights into the quantile space $[0.1, 0.2, \ldots, 0.9]$ [53].

2.3.8. Deep State Space Models

DeepSSM [36] is a probabilistic time-series model that combines state space models (Section 2.3.2) with DL. The model parameterizes a linear state-space model, implemented via a Kalman filter, separately per time-series, but using a RNN whose weights are learned globally across all time-series. The model is therefore able to combine some desired properties of state space models such as data efficiency and interpretability, with other advantages of global models, i.e., learn complex cross-series mappings [36]. The model learns a globally shared mapping from the covariate vector $x_{i,1:T_i}$ associated with each target time-series $z_{i,1:T}$ to the parameters of a linear state space model for the i-th time-series.

2.4. Software

All data and models were processed and built with `Python 3.8.5` and `R 4.0.3` using several additional packages for both languages. OLS fits were calculated with `SciPy 1.5.2` [54] because of its attractive computational speed. The SimpleLSTM model was built on top of `Torch 1.4.0` [52]. All other models were built using `GluonTS 0.5.2` [53], which relies on `MXNet 1.6.0` for weight optimization. However, in fact both State Space Models (SSMs) (ETS and ARIMA) were built via the `GluonTS RForecastPredictor`, a thin wrapper calling the `RForecast` package [26]. All models were trained and evaluated on an Intel(R) Core(TM) i5-4200M CPU @ 2.50 GHz processor. Processing time was measured using `Python`'s built-in module time.

2.5. Hyper-Parameters

Most hyper-parameters of models developed in `GluonTS` are the same. For instance, all were initialized with Xavier weights, 0.001 learning rate, 0.5 learning rate decay and 0.1 dropout rate. Additionally, they were all run for 100 epochs, each epoch consisting of 50 batches with 32 time-series per batch. Furthermore, the conditional probability was modeled using the student's t-distribution. Finally, the required frequency field in `GluonTS` was set to annual frequency, matching the annually averaged observations. Model specific (hyper-)parameters are mentioned at their description.

2.6. Model Validation and Evaluation Measures

Forecast accuracy was assessed by comparing model predictions with observed ground truth values by means of several KPIs. For each time-series the last seven years of SDS were reserved for accuracy assessment. Figure 3 shows how the time-series are divided into train and test observations. The classical models optimize their free parameters per time-series over the training observations while performance is subsequently assessed using the test observations (Figure 3). The DL algorithms first sample sequences of observations from the training part of each series (Figure 3). These sequences consist of a adjacent training and validation window. The training window corresponds to the conditioning range, i.e., observations provided to the model to generate its predictions, with a maximum length of 19 years during training. The validation window is the last

output window that is used for network validation. The parameters with DL models are learned jointly over the time-series by evaluating their performance per batch of time-series over a validation window. When the model has trained, i.e., the network weights are optimized, its performance can be assessed over the test observations. To update weights of probabilistic forecast models the median quantile loss was used; for point forecast models MSE loss was used. Probabilistic forecast predictors are set to produce 300 sample-forecasts from the trained model. Together these trajectories yield a non-parametric representation of the predictive distribution.

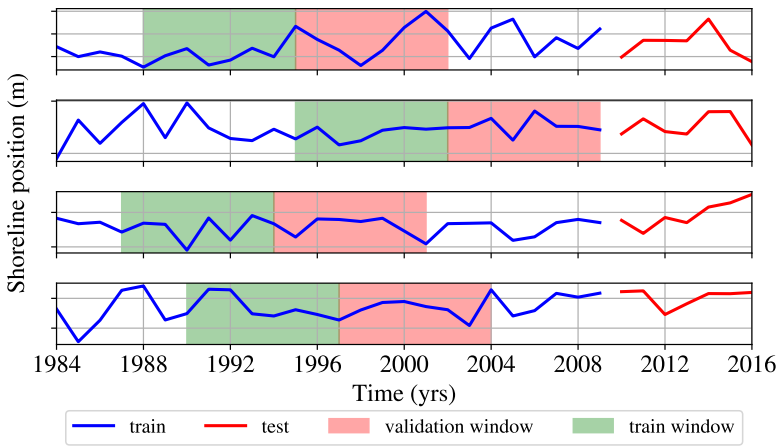

Figure 3. Schematic overview of train, validation and test partitions. The latter seven years of observations (test partition) remain hidden to the models when optimizing their parameters and are reserved for forecast accuracy assessment. Considering the ML models, train and validation windows are sampled within the train sequence. These models update their weights per batch of time-series and optimize their hyperparameters using the validation window.

To compare forecast performance of the different models we used several error measures commonly found in forecasting literature, including the MSE, Mean Absolute Error (MAE), Mean Absolute Scaled Error (MASE), Mean Absolute Percentage Error (MAPE) and Continuous Ranked Probability Score (CRPS). It is commonly noted that every error measure has its own (dis)advantages and that there exists no perfect error measure. Essentially these KPIs are trying to summarize model performance over all time-series in one number which is an immense simplification of reality.

$$\mathrm{MAE}_i = \frac{1}{N}\sum_{t=1}^{N}|z_{i,t} - \hat{z}_{i,t}| \tag{4}$$

$$\mathrm{MSE}_i = \frac{1}{N}\sum_{t=1}^{N}(z_{i,t} - \hat{z}_{i,t})^2 \tag{5}$$

Both MAE (Equation (4)) and MSE (Equation (5)) are scale dependent, i.e., their scale depends on the scale of the data. The measures are often used to compare different methods on the same set of data. Importantly MSE are more sensitive to outliers than MAE because it gives more weight to most significant errors. On the other hand, MAE can be biased when the distribution of SDS is skewed as it aims to split the dataset in two equal parts. Squared Error (SE) and Absolute Error (AE) are further used to derive various other performance indicators such as Root Mean Squared Error (RMSE) ($\sqrt{\mathrm{MSE}}$), Normalized Root Mean Squared Error (nRMSE) ($\frac{\sqrt{MSE}}{|target|}$) and Normalized Difference (ND) ($\frac{\mathrm{AE}}{|target|}$).

$$\mathrm{MAPE}_i = \frac{100}{N}\sum_{t=1}^{N}\frac{|z_{i,t} - \hat{z}_{i,t}|}{z_{i,t}} \tag{6}$$

$$\text{sMAPE}_i = \frac{100}{N} \sum_{t=1}^{N} \frac{|z_{i,t} - \hat{z}_{i,t}|}{(|z_{i,t} + \hat{z}_{i,t}|) * 2} \tag{7}$$

MAPE (Equation (6)) and Symmetric Mean Absolute Percentage Error (sMAPE) (Equation (7)) are measures based on percentage errors. These metrics are often used to assess performance across different datasets as they have the advantage of being scale-independent. Although sMAPE specifically addresses the issue of irregular penalty [55] for negative errors in MAPE, both methods are sensitive to values close to 0. When $z_t = 0$ for any t in the considered series these measures become infinite or undefined. Therefore, these error measures can easily be dominated by a single forecasting outlier. This issue is circumvented by adding 1 to both nominator and denominator for all observations with $z_{i,t} < 1e^{-8}$. Finally, percentage errors assume meaningful zero [56], i.e., the zero point is real and actually means there is nothing. The SDS time-series are measurements of length (m) with respect to a reference year (1984) and therefore do not violate this assumption.

$$\text{MASE}_i = \frac{1}{N} \frac{\sum_{t=1}^{N} |z_{i,t} - \hat{z}_{i,t}|}{\frac{1}{M-1} \sum_{t=2}^{M} |z_{i,t} - z_{i,t-1}|} \tag{8}$$

Scaled errors can be used to compare forecast accuracy across series with different units. Hyndman and Koehler [56] introduce MASE, which is based on scaled errors by using naïeve (seasonal) forecasts to calculate the relative error. With MASE the absolute forecast errors are normalized by the average in-sample one-step naïve (seasonal) forecast error. Thus, MASE scores larger than 1 indicate that forecast accuracy of the model under consideration is worse than its naïve benchmark, and vic versa.

Probabilistic forecast algorithms generate a probability distribution instead of a point-estimate value. For evaluation, various sample forecasts are drawn from the trained model, which is described by an estimated parametric distribution. This sample of forecasts (or sample paths) constitute the empirical distribution of forecasts. Evaluation is performed by verifying for each sample in the set whether an observation occurred or not, i.e., if the sample observation is observed below the prediction interval or not.

The CRPS (Equation (9)) [57] can be used to measure dissimilarity between the Cumulative Distribution Function (CDF) and the observed values. The CRPS assigns a numerical score between the CDF $\hat{F}_{\{T_0:T_{0+H}\}|T}$ and the observed values $z_{\{T_0:T_{0+H}\}}$, which when assessed over quantile forecasts, translates to:

$$\text{CRPS}(\hat{Q}, z) = 2 \int_0^1 [\hat{Q}(p) - z][1_{\{z \leq \hat{Q}(p)\}} - p] dp \tag{9}$$

where $\hat{Q}$ is the estimated CDF, p a probability level, and z the observed value. In this study we estimate the CDF by drawing 300 forecast samples from the trained model. Then the CRPS is approximated as follows. For a given quantile level p the sum of quantile losses of all observations in the prediction horizon are calculated as $2 \sum_{t=T_0}^{H} [\hat{Q}_{t,p} - z_t][1_{\{z_t \leq \hat{Q}_{(t,p)}\}} - p]$. This so-called pinball loss is weighted by dividing it by the sum of absolute target values over the prediction horizon for all time-series, i.e., $\frac{\sum_{i=1}^{N} \text{pinball loss}_i}{\sum_{i=1}^{N} \sum_{t=T_0}^{H} |z_{i,t}|}$, where N are the number of transects and H is the prediction horizon. These weighted quantile losses were calculated for quantile levels $p \in [0.1, 0.2, \ldots, 0.9]$. The CRPS is approximated by mean averaging the weighted quantile losses.

Another scoring method included for assessing probabilistic forecast performance is the Mean Scaled Interval Score (MSIS) [58]. The metric scores at the same time upper and lower prediction intervals.

3. Results

To compare forecast skill among used models, we first present a comparison of various error measures. We further present reliability and loss curves for the probabilistic

algorithms and report processing time for all algorithms. We also introduce figures describing forecast error with respect to the prediction horizon. To conclude we present a probabilistic forecast example, with shoreline predictions up to 2030.

3.1. Assessing Model Performance

Table 1 presents mean-aggregated error metrics per evaluated forecast method. All metrics are derived from the same set of 37,111 time series, over a forecast horizon of seven years. The SimpleLSTM model has best accuracy scores for error measures based on squared error loss. Yet this method is also characterized by weak scores in terms of other error measures, including both MAPE and MASE. The data further indicate that ETS models have good and consistent forecast performance. They are outperforming all other algorithms by MAE and MASE, while doing relatively well in terms of the other error measures. With this data we can see that OLS fits have, besides sMAPE, low accuracy for all error measures with respect to other methods. The results show that ETS, ARIMA, SimpleFFN, DeepAR and MQCNN consistently have relatively good accuracy scores, while differences within this group are notably small, especially for the probabilistic error measure CRPS. The MQCNN model directly outputs quantile forecasts, while other probabilistic methods derive these by sampling their CDF multiple times. Since the MQCNN model architecture does not allow deriving MSE-based models in similar fashion as for the other probabilistic models these accuracy metrics are not included in the table.

Table 1. Mean-aggregated error per forecast method. Reported numbers are measured over a forecast horizon $H = 7$ years. Probabilistic forecast indicators CRPS and MSIS could only be calculated for probabilistic forecasting algorithms. Further, SE based measures for MQCNN are not included because the network architecture does not allow deriving SE-based performance indicators in similar fashion as the other probabilistic forecast algorithms. Lower scores indicate better forecasts.

	MSE	MAE *	MASE	MAPE	sMAPE	RMSE *	NRMSE	ND	MSIS	CRPS
Linear regr	2510	20.0	1.54	14.0	0.83	50.1	1.38	0.55	-	-
ETS	1998	**16.6**	**1.35**	11.9	0.84	44.7	1.23	**0.46**	16.6	0.39
ARIMA	2032	17.1	1.38	13.4	0.88	45.1	1.24	0.47	17.0	0.40
SimpleLSTM	**1686**	19.0	1.66	20.4	0.92	**41.1**	**1.11**	0.51	-	-
SimpleFFN	1817	17.0	1.36	11.4	**0.82**	42.6	1.17	0.47	17.1	0.39
DeepAR	1930	17.6	1.39	**10.0**	0.83	43.9	1.21	0.48	**16.2**	0.40
MQCNN	-	17.2	1.38	11.1	0.82	-	-	0.47	-	0.39
DeepSSM	2060	18.4	1.44	12.8	0.82	45.4	1.25	0.51	18.6	0.43

* Unit in meters.

Table 2 presents forecast rank scores per error measure, which are derived by ranking all generated forecasts per error measure, from best to weakest performance, and computing the sum of the ranks per method. The data shows that the SimpleFFN model achieves best forecasting results in terms of ranking. The method outperforms all other algorithms by MAE, MASE and sMAPE, while obtaining the second-best rank MSE and MAPE, after ES and DeepAR respectively. Although the SimpleLSTM model is able to outperform other algorithms by MSE-based statistics, its MSE-ranking is by far the weakest. Moreover, the forecasts generated by this algorithm overall score the worst ranking for all ranked error measures. In accordance with results in Table 1, OLS forecasts have a low ranking compared to other algorithms. Overall, besides weak performance of the SimpleLSTM model, forecast ranking scores appear to be generally in accordance with mean error measures as presented in Table 1.

Figure 4 presents the log-normalized error distributions, of MSE, MAE, MAPE and MASE. Error measures are log-normalized because all distributions of error are heavily tailed. These tails exist in both directions, although the positive domain is characterized by higher skew than the negative domain. Forecasts generated by the SimpleLSTM model are characterized by higher MSE kurtosis than other algorithms, while its MAPE distribution has highest overall errors. All models except SimpleLSTM are characterized by a distinctive spike around zero-MAPE. Overall, these results suggest that underlying distributions of

error in the forecasts are unevenly distributed and characterized by heavy tails, especially in positive domain.

Table 2. Forecast rank per error measure and evaluated method. The score is derived by ranking all generated forecasts, from best to worst performance, and computing the sum of scored ranks per method. Only error measures which were calculated at individual time-series level could be included in the ranking. Note, lower scores indicate better forecasts.

	Rank MSE	Rank MAE	Rank MASE	Rank MAPE	Rank sMAPE
Linear regr	131,145	151,161	152,599	154,082	146,465
ETS	**123,708**	142,775	141,349	144,806	146,710
ARIMA	125,312	144,471	143,431	143,954	151,876
SimpleLSTM	149,200	172,123	175,640	161,818	160,722
SimpleFFN	125,084	**141,962**	**140,551**	143,918	**144,548**
DeepAR	125,253	143,065	142,409	**143,292**	146,913
MQCNN	-	142,702	140,940	143,920	145,099
DeepSSM	129,489	149,260	150,600	151,725	145,183

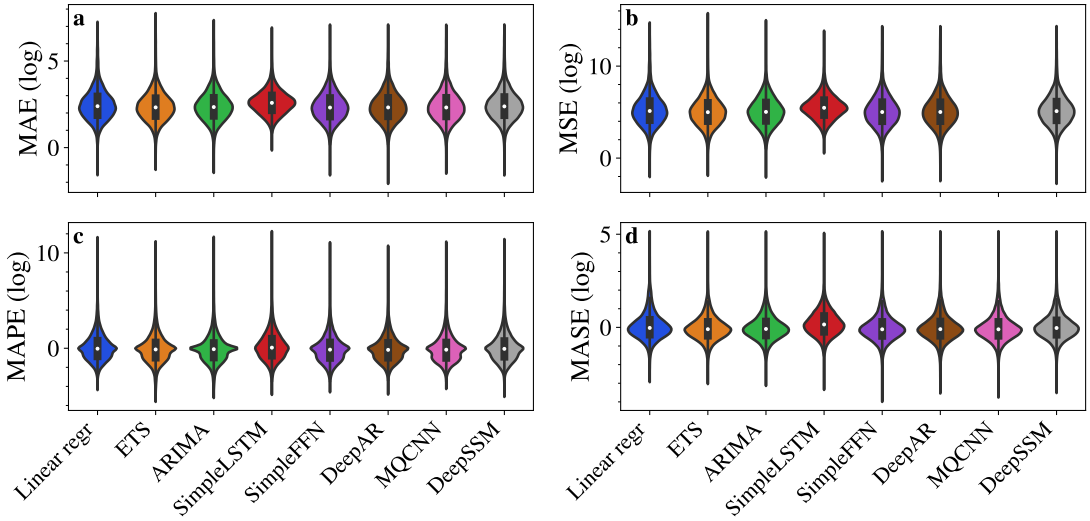

Figure 4. Log-normalized density-kernel estimations for four types of error measures. The error metrics are computed over a forecast horizon $H = 7$ years. (**a**) MAE; (**b**) MSE; (**c**) MAPE; (**d**) MASE.

Figure 5 shows quantile coverage and loss for the probabilistic forecasting approaches, obtained by evaluating the algorithms over the 7 year forecast horizon. Figure 5a is a so-called reliability plot, where the x-axis represents nominal probabilistic levels and the y-axis describes observed levels. For a quantile forecast $\hat{z}^{(\alpha)}_{t_0+H}$ with nominal level $\alpha = 0.5$, one expects that the observations z_{t_0+H} (y-axis) are to be less than the nominal level $\hat{z}^{\alpha}_{t_0+H}$ (x-axis) 50% of the times. Thus, coverage in a reliability plot indicates how much of the data is below the predicted quantile for the given level. The optimal reliability curve is described by $y = x$, i.e., the nominal probabilistic levels are equivalent to the observed values. Figure 5a shows that quantile coverage is for all probabilistic algorithms very close to the optimal diagonal (dashed line) over the full range of nominal quantile levels. Deviation of this the optimal reliability curve captures conditional bias: when empirical coverage is lower than the corresponding nominal level this indicates overshooting, and vice versa. Thus, the data suggest that all probabilistic models are reliable.

Figure 5b shows quantile loss over the different quantile levels. Quantile loss is computed as the pinball loss (Section 2.6) multiplied by 2, so that the 50th-percentile is equivalent to the total AE over the forecast horizon. Thus, when mean averaging quantile loss over the prediction horizon (Here, $H = 7$), the 50th-percentile is equivalent to MAE values reported in Table 1. The figure can be used to deduce which models are preferable

for certain parts of the probability distribution. Note, lower quantile loss indicates more accurate predictions. The data show that quantile loss is centered approximately between the 50-th and 60-th quantile and that loss is higher in the upper quantiles. Overall all probabilistic algorithms describe a similar curve, although loss is considerably higher for the DeepSSM model, especially in the upper quantile regions.

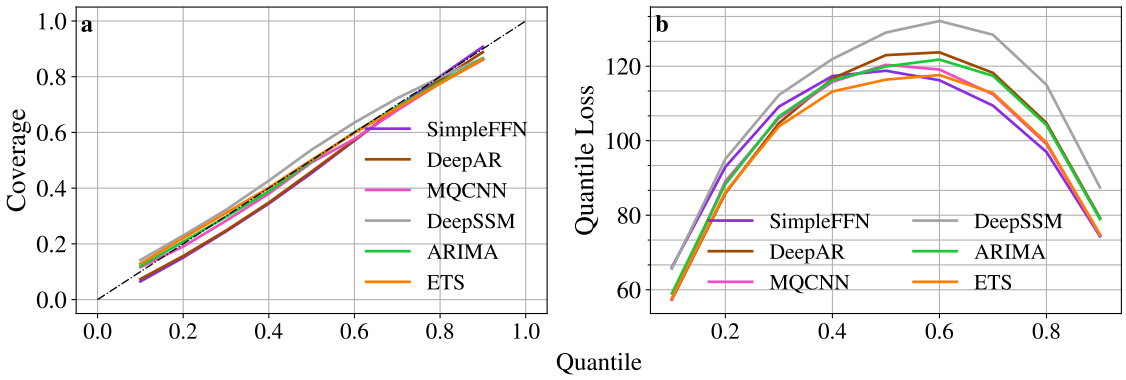

Figure 5. Quantile coverage and weighted quantile loss for probabilistic forecast algorithms. (**a**) Coverage indicates how much of the data is below the predicted quantile for the given level. The dashed blue diagonal represents the ideal scenario where observed quantile coverage exactly matches the nominal quantile levels. (**b**) Quantile loss is the pinball loss multiplied by 2, so that the 50th-percentile equivalents the total AE over the forecast horizon. Mean-averaging total AE over the forecast horizon ($H = 7$) are equal to MAE error measures presented in Table 1. In this panel lower quantile loss indicates more accurate predictions. It shows which models are preferable for certain parts of the the probability distribution.

Table 3 reports processing time per algorithm over one complete evaluation iteration. The data show that evaluating OLS, SimpleFFN and MQCNN is considerably faster than other algorithms.

Table 3. Processing time (minutes) of the performance evaluation, reported per algorithm, measured over one complete evaluation iteration. For all algorithms, besides SimpleLSTM, evaluation windows were distinguished within each time-series (Figure 3) so that the evaluation set compromises all time-series ($N = 37,111$). For SimpleLSTM runtime was measured over a 5-times cross-validation of the test set ($N = 7400$); in total 37,000 series. Considering ML algorithms, reported processing times do not include model training.

	OLS	**ETS**	**ARIMA**	**SimpleLSTM**	**SimpleFFN**	**DeepAR**	**MQCNN**	**DeepSSM**
t (min)	0.3	29.4	126.1	57.1	1.6	33.0	1.45	25.1

3.2. Long-Term Predictions

To determine the effect of prediction length on the forecast skill we repeated the analysis over a gradually increasing prediction range from 1 until 11 years. For each forecast horizon, a separate model was trained and evaluated. Figure 6 shows the forecast performance in terms of MSE and CRPS over increasing prediction lengths. A maximum forecast horizon of 11 years was chosen so that the algorithms could at least be provided with data spanning twice the forecast horizon, i.e., 22 years of SDS. Figure 6a shows MSE for the probabilistic algorithms and OLS over increasing prediction length. In this figure we can observe that probabilistic forecast algorithms consistently outperform OLS over all prediction lengths by MSE. The data further shows that MSE increases with prediction length, approximately with a linear trend, although DeepAR and DeepSSM are characterized by somewhat more irregular trends. Overall the graph indicates that, considering all prediction lengths, SimpleFFN, ETS and ARIMA provide best forecasts, when evaluated by MSE. Figure 6b presents mean weighted quantile loss, an approximation of the CRPS (Section 2.6), over prediction length. Since CRPS can only be computed for probabilistic forecasts OLS and SimpleLSTM are not included in the graph. Overall, and

similar to results described in panel a of Figure 6, CRPS increments approximately with a linear trend over increasing prediction lengths, although this trend is more regular for ETS, ARIMA, SimpleFFN, MQCNN than for DeepAR and DeepSSM. The data show that the DeepSSM model is outperformed consistently over the full prediction length range by all other probabilistic algorithms. Considering all predictions lengths the graph indicates that ETS, ARIMA, SimpleFFN and MQCNN are able to produce relatively tight and accurate results with respect to DeepSSM and DeepAR.

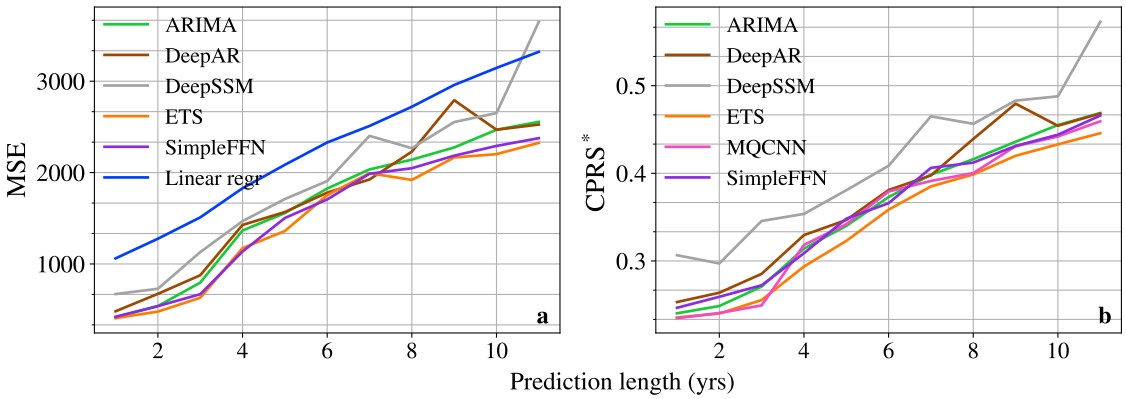

Figure 6. Relation between respectively prediction length and some accuracy measure. Models were trained and evaluated separately per prediction length. Naturally, forecast error can be expected to increase when models are set to predict longer horizons. (**a**) Forecast accuracy measured by MSE; and, (**b**) weighted quantile loss (*), an approximation of the CRPS. The CRPS integrates quantile loss over all quantile levels and therefore generalizes MAE measures for probabilistic forecasts. OLS and SimpleLSTM are not present in this panel because only probabilistic forecasts can be evaluated by CRPS.

To generate some forecast examples, all models were retrained, but now without reserving the latter 7 years of observations for accuracy assessment. The models were set to produce shoreline evolution predictions of 14 years, until 2030. Figure 7 shows the 14-year probabilistic forecasts of DeepAR and MQCNN, for a transect at Carova beach (36.5253°N; 75.86103°W), a sandy stretch of coast in North Carolina, USA. These examples illustrate how (probabilistic) data-driven time series models can be used to project multi-decadal shoreline evolution beyond linear realm. The graphs also show that although overall error statistics are similar for both models, they generate fairly different shoreline predictions.

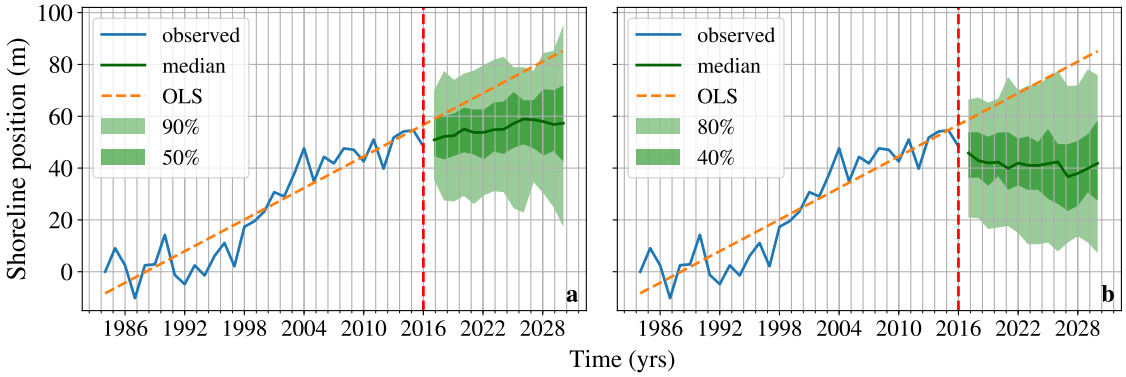

Figure 7. Probabilistic forecasts for a transect at Carova beach (36.5253°N; 75.86103°W), North Carolina, USA. (**a**) Forecast generated by DeepAR; and, (**b**) MQCNN until 2030. Please note that prediction intervals for both models are slightly different. MQCNN's configuration only provides forecasts at the point-decimal quantiles so that the 0.05, 0.25, 0.5, 0.95 quantiles, required to calculate the 50% and 90% prediction intervals, are missing.

4. Discussion

In science, simple models are typically preferred over complex ones [59]. When future shorelines can be predicted successfully with an OLS fit, such model should be preferred over models of higher complexity. Although future shorelines can for many sites be approximated with an OLS regression, some shorelines evolve with patterns that cannot be captured with straight line (Figure 1b). Processes that cause seasonal and non-linear dynamics can be both of natural origin, such as a changing seasonality of storms [60], or human-induced, such as coastal interventions such as sand nourishments [61]. The relatively weak (ranked) accuracy metrics of simple OLS (Tables 1 and 2) might be related to the fact that an OLS regression cannot model these types of coastal dynamics. When shorelines are subject to forces causing non-linear shoreline dynamics, or when shoreline oscillations are to be included in the prediction, other forecast algorithms than OLS could be better suitable to predict future trajectories.

The classical methods ETS and ARIMA model shoreline trajectories by estimating a set of parameters describing structural aspects of the time-series. ARIMA models are able to capture a flexible range of different shoreline evolution patterns using its autoregressive component. Moreover, with a differencing operator (I-component) directly integrated into the model it is able to deal with linear trends present in shoreline evolution, e.g., steady erosion due to depleted sediment fluxes in a nearby river. ETS models do not assume stationary series because trend is explicitly set as one of its parameters. While ETS are able to deal with multiplicative trend and seasonality, ARIMA models can only account for additive trend and seasonality. Multiplicative trends in shoreline trajectories might occur in areas with sand extraction [62] or coastal development [63] because both presumably grow with population and economy. The moving average component in ARIMA is derived from errors that the model made in the past. In practice, ARIMA models can therefore deal with shocks in shoreline evolution, such as a beach nourishment which gradually weakens over time. Combining autocorrelations with differencing and a moving average error dependence, or error with trend and seasonality, appears to be a powerful tool for univariate shoreline evolution forecasting as much of the variability observed in shoreline evolution can be captured by these components. Nevertheless, model simplicity should be balanced against goodness-of-fit, with a model selection method such as AIC(c), to avoid overfitting. The AIC cannot be used to compare different families of models, such as ETS and ARIMA, because AIC has to be computed over the same set of values, while the differencing component in ARIMA shortens the signal per applied order. For model comparison between different types of models we therefore rely on error measures. In line with [12], the results show that methods based on weighted averages and autocorrelations (here, ETS and ARIMA) are a favorable choice over OLS extrapolation for modeling future shoreline trajectories, respectively for 55% and 53% of the time-series, when evaluated by MSE. In addition to previous studies, the data indicate that this advantage is also valid for globally SDS and holds over longer forecast horizons; at least up to 11 years.

Contrary to statistical approaches, ML methods learn the free model parameters globally over a collection of time-series. Therefore these models should be able to extract patterns from irregular collections of shoreline evolution series which cannot be distinguished at individual level. Assuming that shoreline evolution at transects in close proximity evolve in similar patterns, or are at least correlated in their evolution [64], it was hypothesized that making the ML models spatially aware (Figure 1b) would enhance forecast performance and result in a clear advantage of ML methods over classical approaches. Yet, the (ranked) performance indicators (Tables 1 and 2) indicate that the level of spatial awareness we provided to the ML models was not sufficient to make them outperform classical forecast methods (Figures 5 and 6). In addition, many of the considered time-series exhibit a trend, which is easily captured by the classical approaches [26], but typically remains a challenge for ML models. Other reasons for the considerably weak ML performance can, most likely, be attributed to the absence of any forcing data of shoreline dynamics and the quality of SDS data. No additional explanatory variables

were available at the same spatial- and temporal resolution as the SDS data. Although used Earth-observing satellite data has global coverage, it has low temporal resolution (annual composites) and high granularity (15/30-m), with respect to the scale of future shoreline changes (often tens-of-meters). Performance of our ML algorithms is, most likely, hampered by absence of any shoreline-dynamics forcing data and the quality of the SDS data, as ML algorithms solely rely on data and its quality [65]. As a consequence, our probabilistic ML methods, evaluated by MSE, do not outperform OLS regression for 46% of the transects.

The results show that ETS, ARIMA, SimpleFFN, DeepAR and MQCNN consistently outperform OLS, SimpleLSTM and DeepSSM. However, whereas OLS and DeepSSM show weaker performance over all (ranked) error measures (Tables 1 and 2), each part of the probability distribution (Figure 5b) and over all prediction lengths (Figure 6), SimpleLSTM generates best forecasts by squared-error metrics (MSE, RMSE and nRMSE), but worst in terms of other KPIs. These data particularly highlight how forecasts are optimized to a certain learning objective. SimpleLSTM is trained to minimize MSE, which considers every error with equal weight, across all time-series, hence, learns one global model, which is accurate on average, but does not generate meaningful forecasts at site-specific level. The weak performance of this model is in accordance with our assumption that one global forecasting model for shoreline evolution, without site-specific bias, is unreasonable. To generate meaningful forecasts with SimpleLSTM, it should either be provided with site-specific (forcing) data or be applied to a cluster of transects, which are subject to similar shoreline dynamics (e.g., [21,22]). The other ML algorithms are probabilistic models, which are optimized by minimizing quantile loss. Although these models produce worse results than SimpleLSTM over all equally weighted errors, they generate better forecasts for the larger proportion of the dataset, because they target median forecasts. Considering the unequal distribution (Figure 1a) of SDS and the amount of shocks present therein favoring median forecasts appears to be a more reasonable objective.

The relatively high MSE (Table 1) reflects the sensitivity of this metric to outliers. Whereas variability is for most time-series in the order of tens-of-meters, some are characterized by sudden shocks, up to several hundreds of meters (Figure 1a). When these shocks occur unpredictably during the test window (latter seven years) they have an immense impact on MSE, as it is directly proportional to the square. The effect of outliers is further illustrated by the approximately log-normal distributions of all error measures (Figure 4) and indicates that overall forecast performance is hampered by a considerably small amount of transects. Figure 8c shows an ARIMA forecast for a site located in Mar del Plata, Argentina, next to a groyne (37.9658°S; 57.5391°W). This transects is a typical example for one of the sites where the algorithms, unsurprisingly, fail to predict future shoreline trajectories because the shoreline dynamics drastically change during the evaluation window. Figure 8d shows a count-density plot of time-series which compromise the 5% worst forecasts per algorithm. The figure shows that for the majority of transects where algorithms arguably fail to predict future shorelines, the shoreline trajectories drastically change during the evaluation window. Similar to the transect example of Mar del Plata (Figure 8c), these sudden changes can often be traced back to coastal interventions, such as construction of coastal defenses or beach nourishments. Within this context, it is particularly important to note that ML algorithms would be capable of adequately modeling these trajectories as their predictions can be conditioned by covariates (Equation (3)).

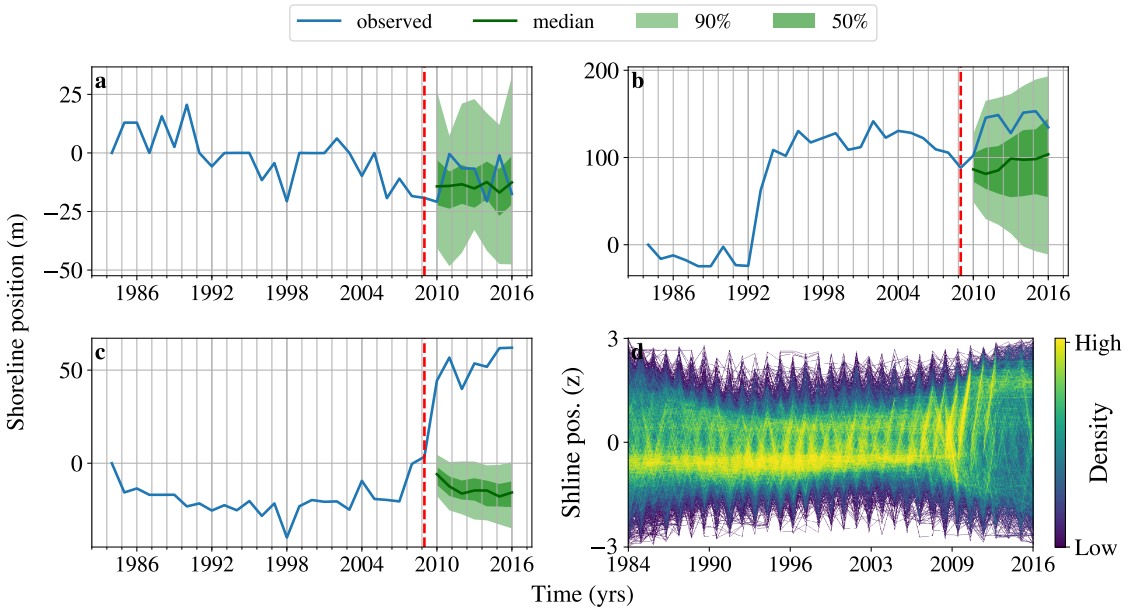

Figure 8. Forecast examples to illustrate typical behavior of the forecast algorithms. (**a**) SimpleFFN forecast for a site located on Cape Lookout (34.9572°N; 76.1767°W), North Carolina, USA. This example shows how probabilistic forecasts are typically reliable and predict the right trend, but lack (some) sharpness. (**b**) ETS forecast for a transect site on Peck's beach (39.2672°N; −74.5840°W), New Jersey, USA. The graph shows how classical models are able to deal with shocks in shoreline trajectories, especially when observations lie more distantly in the past. (**c**) ARIMA forecast for a site located, next to a groyne, on the coast of Mar del Plata, Argentina (37.9658°S; 57.5391°W). (**d**) Count-density plot of standarized time-series which compromise per algorithm the 5% worst forecasts in terms of MSE.

Apart from OLS regression and the SimpleLSTM all algorithms generate probabilistic forecasts. Describing shoreline evolution predictions in terms of probabilities respects the dynamic and fluid behavior of these natural systems while ensuring a richer characterization of the data. Although all probabilistic algorithm generated reliable predictions (Figure 5a) forecast skill appeared often to be hampered by relatively low sharpness (wide prediction intervals) (Figures 7 and 8a,c). In general, the prediction intervals appear to be too wide to provide basis for coastal management or derive other conclusions thereof. Although the results indicate similar probabilistic forecast performance for SimpleFFN, DeepAR and MQCNN (Table 1; Figures 5 and 6b), we noted that MQCNN generates forecasts with sharper prediction intervals, which is, most likely, related to the fact that MQCNN directly outputs quantile forecasts, while the other models generate quantile forecasts by sampling the CDF.

Although our ML models did not significantly outperform classical approaches they showed other advantages. Whereas most ML models were evaluated in several minutes, ARIMA took more than two hours (Table 3). Considering applicability at global level, where for a study conducted at the same spatial scale, approximately 2 million transects should be considered, this will become an important factor to be taken into account. Whereas training time of a shoreline forecasting ML algorithm will increase when additional explanatory variables are fed to the model, inference times will remain almost instantaneous, making these models capable of on-demand forecasting at global level. Nevertheless, the main advantage of ML methods over classical methods is undoubtedly their capability to consider additional covariates capturing processes that drive shoreline dynamics. Although no forcing data was available for this study, we expect such data will be introduced in near-future, bringing about major advances in learning across time-series of SDS and their associated covariates, and hence, to better long-term predictions of future shorelines.

The algorithms and data used have several limitations directly affecting our results, which should also be taken into account when using this approach to predict shorelines

and draw conclusions thereof. Natural phenomena, such as foam from induced breaking waves in areas with consistent swell, can cause poorly derived shorelines SDS, manifested as shocks in the time series. Furthermore, the 33-year window may not fully resolve some rare cases of coastal change, such as extreme events and human interventions [11]. Finally, whereas this signal can be considered long enough to express decadal-scale variability [11], observation frequency (annual composites) and signal length (33-years) are relatively small with respect to algorithm requirements and forecast horizon [53].

5. Conclusions and Future Directions

This study can be considered as one of the first steps towards a more thorough understanding of shoreline modeling from a data-driven and time-series modeling perspective that is of interest to coastal engineers, managers and scientists. Evaluating eight forecast algorithms over 37-thousand SDS time-series, we find that classical forecast approaches ETS and ARIMA altogether with ML models SimpleFFN, DeepAR, MQCNN are more skilled than OLS regression for the majority of study sites. When predicting shorelines seven years ahead, these algorithms are preferable over OLS regression for 54% of the transect-sites, with, on average, 29% smaller MSEs. This study has also shown that the forecast advantage of these algorithms over OLS regression exists for all considered forecast horizons (1 to 11 years); an important indicator for that these algorithms can also be successfully applied on multi-decadal forecast horizons. The experiments have shown that ML models did not generate significantly better predictions than classic methods. The considerably weak performance of the ML algorithms is attributed the absence of shoreline-dynamics forcing data and to the granularity of the SDS data, as ML algorithms solely depend on data and its quality. The results further show that ML approaches have significant computational advantages, which will be of importance for global studies, including many more transect-sites, and for on-demand predictions in a production environment. We also demonstrate that ML techniques have two important advantages over classic time-series forecast methods: they are able to consider additional explanatory variables, and that they can do so, while learning across time-series of shorelines. We expect that these advantages will be extensively exploited in the near-future, when shoreline-dynamics forcing data will come available at same spatial and temporal resolution as SDS, enhancing conditional probabilistic shoreline evolution forecasting. Although the prediction intervals of most forecasts generated in this study are too wide to sustain reliable coastal management or to derive other conclusions thereof, the evaluated algorithms have the potential to become valuable tools for coastal engineers to prioritise coastal maintenance tasks; scientists who study shoreline dynamics or wish to include future shoreline trajectories into their studies; and, coastal managers who face important decisions about this vulnerable zone.

Author Contributions: F.C. conceived methodology, developed all code, analyzed results and wrote original draft—first as MSc student Data Science at the University of Amsterdam, later as graduate intern at Deltares. A.L. conceived the study, supervised the whole project, curated original SDS data, and contributed to reviewing, writing and editing. C.R.R. contributed to forecasting study design, supervised MSc-thesis part on behalf of University of Amsterdam and contributed to reviewing, writing and editing. E.K. contributed to methodology, supervised MSc-thesis part on behalf of Deltares, curated updated SDS data and contributed to reviewing, writing and editing. F.B. contributed to interpretation of results, curated original SDS data and contributed to reviewing, writing and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Deltares Strategic Research Programme 'Coastal and Offshore Engineering' and 'Seas and Coastal Zones'.

Data Availability Statement: Source code and data presented in this study are available upon reasonable request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SDS	Satellite-derived Shorelines
NDWI	Normalized Difference Water Index
SLR	Sea-level Rise
ARIMA	Autoregressive Integrated Moving Average
ETS	Error Trend Seasonality
STL	Season Trend Loess
OLS	Ordinary Least Squares
ES	Exponential Smoothing
SSM	State Space Model
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
MLP	Multi-layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory
AIC	Akaike Information Criterion
AE	Absolute Error
SE	Squared Error
CDF	Cumulative Distribution Function
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
sMAPE	symmetric Mean Absolute Percentage Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
nRMSE	Normalized Root Mean
MASE	Mean Absolute Scaled Error
CRPS	Continuous Ranked Probability Score
MSIS	Mean Scaled Interval Score
KPI	Key Performance Indicator

References

1. Nicholls, R.J.; Cazenave, A. Sea-Level Rise and Its Impact on Coastal Zones. *Science* **2010**, *328*, 1517–1520. [CrossRef]
2. McLachlan, A.; Brown, A. 2—The Physical Environment. In *The Ecology of Sandy Shores*, 2nd ed.; McLachlan, A., Brown, A., Eds.; Academic Press: Burlington, NJ, USA, 2006; pp. 5–30. [CrossRef]
3. Long, J.W.; Plant, N.G. Extended Kalman Filter framework for forecasting shoreline evolution. *Geophys. Res. Lett.* **2012**, *39*. [CrossRef]
4. Luijendijk, A.; Hagenaars, G.; Ranasinghe, R.; Baart, F.; Donchyts, G.; Aarninkhof, S. The State of the World's Beaches. *Sci. Rep.* **2018**, *8*. [CrossRef] [PubMed]
5. Murray, N.J.; Phinn, S.R.; DeWitt, M.; Ferrari, R.; Johnston, R.; Lyons, M.B.; Clinton, N.; Thau, D.; Fuller, R.A. The global distribution and trajectory of tidal flats. *Nature* **2018**, *565*, 222–225. [CrossRef] [PubMed]
6. Vos, K.; Harley, M.D.; Splinter, K.D.; Simmons, J.A.; Turner, I.L. Sub-annual to multi-decadal shoreline variability from publicly available satellite imagery. *Coast. Eng.* **2019**, *150*, 160–174. [CrossRef]
7. Harley, M.D.; Kinsela, M.A.; Sánchez-García, E.; Vos, K. Shoreline change mapping using crowd-sourced smartphone images. *Coast. Eng.* **2019**, *150*, 175–189. [CrossRef]
8. Davidson, M.; Lewis, R.; Turner, I. Forecasting seasonal to multi-year shoreline change. *Coast. Eng.* **2010**, *57*, 620–629. [CrossRef]
9. Davidson, M.; Turner, I.; Guza, R. The effect of temporal wave averaging on the performance of an empirical shoreline evolution model. *Coast. Eng.* **2011**, *58*, 802–805. [CrossRef]
10. Montaño, J.; Coco, G.; Antolínez, J.A.A.; Beuzen, T.; Bryan, K.R.; Cagigal, L.; Castelle, B.; Davidson, M.A.; Goldstein, E.B.; Ibaceta, R.; et al. Blind testing of shoreline evolution models. *Sci. Rep.* **2020**, *10*. [CrossRef] [PubMed]
11. Vousdoukas, M.I.; Ranasinghe, R.; Mentaschi, L.; Plomaritis, T.A.; Athanasiou, P.; Luijendijk, A.; Feyen, L. Sandy coastlines under threat of erosion. *Nat. Clim. Chang.* **2020**, *10*, 260–263. [CrossRef]
12. Southgate, H.N. Data-based yearly forecasting of beach volumes along the Dutch North Sea coast. *Coast. Eng.* **2011**, *58*, 749–760. [CrossRef]

13. Boak, E.H.; Turner, I.L. Shoreline Definition and Detection: A Review. *J. Coast. Res.* **2005**, *214*, 688–703. [CrossRef]
14. Wulder, M.A.; Loveland, T.R.; Roy, D.P.; Crawford, C.J.; Masek, J.G.; Woodcock, C.E.; Allen, R.G.; Anderson, M.C.; Belward, A.S.; Cohen, W.B.; et al. Current status of Landsat program, science, and applications. *Remote Sens. Environ.* **2019**, *225*, 127–147. [CrossRef]
15. Zhu, Z.; Wulder, M.A.; Roy, D.P.; Woodcock, C.E.; Hansen, M.C.; Radeloff, V.C.; Healey, S.P.; Schaaf, C.; Hostert, P.; Strobl, P.; et al. Benefits of the free and open Landsat data policy. *Remote Sens. Environ.* **2019**, *224*, 382–385. [CrossRef]
16. Hagenaars, G.; de Vries, S.; Luijendijk, A.P.; de Boer, W.P.; Reniers, A.J. On the accuracy of automated shoreline detection derived from satellite imagery: A case study of the sand motor mega-scale nourishment. *Coast. Eng.* **2018**, *133*, 113–125. [CrossRef]
17. Dolan, R.; Fenster, M.S.; Holme, S.J. Temporal Analysis of Shoreline Recession and Accretion. *J. Coast. Res.* **1991**, *7*, 723–744.
18. Crowell, M.; Douglas, B.C.; Leatherman, S.P. On Forecasting Future U.S. Shoreline Positions: A Test of Algorithms. *J. Coast. Res.* **1997**, *13*, 1245–1255.
19. Douglas, B.C.; Crowell, M.; Leatherman, S.P. Considerations for Shoreline Position Prediction. *J. Coast. Res.* **1998**, *14*, 1025–1033.
20. Genz, A.S.; Fletcher, C.H.; Dunn, R.A.; Frazer, L.N.; Rooney, J.J. The Predictive Accuracy of Shoreline Change Rate Methods and Alongshore Beach Variation on Maui, Hawaii. *J. Coast. Res.* **2007**, *2007*, 87–105. [CrossRef]
21. Genz, A.S.; Frazer, L.N.; Fletcher, C.H. Toward Parsimony in Shoreline Change Prediction (II): Applying Basis Function Methods to Real and Synthetic Data. *J. Coast. Res.* **2009**, *2009*, 380–392. [CrossRef]
22. Frazer, L.N.; Genz, A.S.; Fletcher, C.H. Toward Parsimony in Shoreline Change Prediction (I): Basis Function Methods. *J. Coast. Res.* **2009**, *2009*, 366–379. [CrossRef]
23. Baart, F. Confidence in Coastal Forecasts. Ph.D Thesis, Delft University of Technology, Delft, The Netherlands, 2013.
24. Hapke, C.J.; Reid, D.; Richmond, B.M.; Ruggiero, P.; List, J. National assessment of shoreline change Part 3: Historical shoreline change and associated coastal land loss along sandy shorelines of the California Coast. *US Geol. Surv. Open File Rep.* **2006**, *1219*, 79.
25. Douglas, B.C.; Crowell, M. Long-Term Shoreline Position Prediction and Error Propagation. *J. Coast. Res.* **2000**, *16*, 145–152.
26. Hyndman, R.J.; Khandakar, Y. Automatic Time Series Forecasting: The Forecast Package for R. *J. Stat. Softw.* **2008**, *27*. [CrossRef]
27. Brownlee, J. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*; Machine Learning Mastery: New York, NY, USA, 2018.
28. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *Int. J. Forecast.* **2021**, *37*, 388–427. [CrossRef]
29. Clements, M.P.; Hendry, D.F. Explaining the Results of the M3-Competition. *Int. J. Forecast.* **2001**, *17*, 537–584. [CrossRef]
30. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: Results, findings, conclusion and way forward. *Int. J. Forecast.* **2018**, *34*, 802–808. [CrossRef]
31. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods. *Int. J. Forecast.* **2020**, *36*, 54–74. [CrossRef]
32. Hyndman, R.J. A brief history of forecasting competitions. *Int. J. Forecast.* **2020**, *36*, 7–14. [CrossRef]
33. Smyl, S.; Ranganathan, J.; Pasqua, A. M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model. 2018. Available online: https://eng.uber.com/m4-forecasting-competition (accessed on 1 April 2020).
34. Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **2020**, *36*, 1181–1191. [CrossRef]
35. Oreshkin, B.N.; Carpov, D.; Chapados, N.; Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv* **2020**, arXiv:cs.LG/1905.10437.
36. Rangapuram, S.S.; Seeger, M.W.; Gasthaus, J.; Stella, L.; Wang, Y.; Januschowski, T. Deep State Space Models for Time Series Forecasting. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31, pp. 7785–7794.
37. Wen, R.; Torkkola, K.; Narayanaswamy, B.; Madeka, D. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv* **2018**, arXiv:stat.ML/1711.11053.
38. Laptev, N.; Yosinski, J.; Li, L.E.; Smyl, S. Time-series extreme event forecasting with neural networks at uber. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 34, pp. 1–5.
39. Bandara, K.; Bergmeir, C.; Hewamalage, H. LSTM-MSNet: Leveraging Forecasts on Sets of Related Time Series with Multiple Seasonal Patterns. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, 1–14. [CrossRef]
40. Kras, E. Planetary-scale Classification of Natural and Human-Induced Sandy Shoreline Evolution. Master's Thesis, TU Delft, Delft, The Netherlands, 2019.
41. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD'96, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996*; AAAI Press: Portland, OR, USA, 1996; pp. 226–231. [CrossRef]
42. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. OPTICS: Ordering Points to Identify the Clustering Structure. *ACM SIGMOD Rec.* **1999**, *28*, 49–60. [CrossRef]
43. Wulder, M.A.; White, J.C.; Loveland, T.R.; Woodcock, C.E.; Belward, A.S.; Cohen, W.B.; Fosnight, E.A.; Shaw, J.; Masek, J.G.; Roy, D.P. The global Landsat archive: Status, consolidation, and direction. *Remote Sens. Environ.* **2016**, *185*, 271–283. [CrossRef]
44. Garcia, P.; Benarroch, A.; Riera, J.M. Spatial distribution of cloud cover. *Int. J. Satell. Commun. Netw.* **2008**, *26*, 141–155. [CrossRef]

45. Hochreiter, S. The Vanishing Gradient Problem during Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **1998**, *6*, 107–116. [CrossRef]
46. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
47. Benidis, K.; Rangapuram, S.S.; Flunkert, V.; Wang, B.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; et al. Neural forecasting: Introduction and literature overview. *arXiv* **2020**, arXiv:cs.LG/2004.10240.
48. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press: Cambridge, MA, USA, 2016.
49. Taylor, J.W. Exponential smoothing with a damped multiplicative trend. *Int. J. Forecast.* **2003**, *19*, 715–725. [CrossRef]
50. Hyndman, R.; Koehler, A.; Ord, K.; Snyder, R. *Forecasting with Exponential Smoothing*; Springer: Berlin/Heidelberg, Germany, 2008. [CrossRef]
51. Hyndman, R.; Athanasopoulos, G. *Forecasting: Principles and Practice*, 2nd ed.; OTexts: Melbourne Australia, 2018.
52. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
53. Alexandrov, A.; Benidis, K.; Bohlke-Schneider, M.; Flunkert, V.; Gasthaus, J.; Januschowski, T.; Maddix, D.C.; Rangapuram, S.; Salinas, D.; Schulz, J.; et al. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *J. Mach. Learn. Res.* **2020**, *21*, 1–6.
54. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
55. Meade, N.; Armstrong, J.S. Long Range Forecasting: From Crystal Ball to Computer (2nd Edition). *J. Oper. Res. Soc.* **1986**, *37*, 533. [CrossRef]
56. Hyndman, R.J.; Koehler, A.B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688. [CrossRef]
57. Matheson, J.E.; Winkler, R.L. Scoring Rules for Continuous Probability Distributions. *Manag. Sci.* **1976**, *22*, 1087–1096. [CrossRef]
58. Gneiting, T.; Raftery, A.E. Strictly Proper Scoring Rules, Prediction, and Estimation. *J. Am. Stat. Assoc.* **2007**, *102*, 359–378. [CrossRef]
59. Green, K.C.; Armstrong, J.S. Simple versus complex forecasting: The evidence. *J. Bus. Res.* **2015**, *68*, 1678–1685.
60. Splinter, K.D.; Turner, I.L.; Reinhardt, M.; Ruessink, G. Rapid adjustment of shoreline behavior to changing seasonality of storms: Observations and modelling at an open-coast beach. *Earth Surf. Process. Landforms* **2017**, *42*, 1186–1194. [CrossRef]
61. de Schipper, M.A.; Ludka, B.C.; Raubenheimer, B.; Luijendijk, A.P.; Schlacher, T.A. Beach nourishment has complex implications for the future of sandy shores. *Nat. Rev. Earth Environ.* **2020**. [CrossRef]
62. Bendixen, M.; Best, J.; Hackney, C.; Iversen, L.L. Time is running out for sand. *Nature* **2019**, *571*, 29–31. [CrossRef]
63. Neumann, B.; Vafeidis, A.T.; Zimmermann, J.; Nicholls, R.J. Future Coastal Population Growth and Exposure to Sea-Level Rise and Coastal Flooding—A Global Assessment. *PLoS ONE* **2015**, *10*, e0118571. [CrossRef] [PubMed]
64. Short, A.D.; Trembanis, A.C. Decadal Scale Patterns in Beach Oscillation and Rotation Narrabeen Beach, Australia: Time Series, PCA and Wavelet Analysis. *J. Coast. Res.* **2004**, *20*, 523–532. [CrossRef]
65. Najafabadi, M.M.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*. [CrossRef]