DEEP LEARNING FOR TIME SERIES FORECASTING: TUTORIAL AND LITERATURE SURVEY

Konstantinos Benidis[*]
Amazon Research
Berlin, Germany
kbenidis@amazon.com

Syama Sundar Rangapuram
Amazon Research
Berlin, Germany
rangapur@amazon.com

Valentin Flunkert
Amazon Research
Berlin, Germany
flunkert@amazon.com

Yuyang Wang
Amazon Research
East Palo Alto, CA, USA
yuyawang@amazon.com

Danielle Maddix
Amazon Research
East Palo Alto, CA, USA
dmmaddix@amazon.com

Caner Turkmen
Amazon Research
Berlin, Germany
atturkm@amazon.com

Jan Gasthaus
Amazon Research
Berlin, Germany
gasthaus@amazon.com

Michael Bohlke-Schneider
Amazon Research
Berlin, Germany
bohlkem@amazon.com

David Salinas
Amazon Research
Berlin, Germany
dsalina@amazon.com

Lorenzo Stella
Amazon Research
Berlin, Germany
stellalo@amazon.com

François-Xavier Aubet
Amazon Research
Berlin, Germany
aubetf@amazon.com

Laurent Callot
Amazon Research
Berlin, Germany
lcallot@amazon.com

Tim Januschowski[*†]
Zalando SE
Berlin, Germany
tim.januschowski@zalando.de

ABSTRACT

Deep learning based forecasting methods have become the methods of choice in many applications of time series prediction or *forecasting* often outperforming other approaches. Consequently, over the last years, these methods are now ubiquitous in large-scale industrial forecasting applications and have consistently ranked among the best entries in forecasting competitions (e.g., M4 and M5). This practical success has further increased the academic interest to understand and improve deep forecasting methods. In this article we provide an introduction and overview of the field: We present important building blocks for deep forecasting in some depth; using these building blocks, we then survey the breadth of the recent deep forecasting literature.

***K*eywords** Time series · Forecasting · Deep learning

1 Introduction

Forecasting is the task of extrapolating time series into the future. It has many important applications [54] such as forecasting the demand for items sold by retailers [41, 190, 156, 136, 14, 25], the flow of traffic [111, 126, 118], the demand and supply of energy [45, 170, 117, 157], or the covariance matrix, volatility and long-tail distributions in finance [30, 29, 124, 12, 197]. As such, it is a well-studied area (e.g., see [84] for an introduction) with its own dedicated research community. The machine learning, data science, systems, and operations research communities as well as application-specific research communities have also studied the problem intensively (e.g., see a series of recent tutorials [56, 55, 57, 58]). In contrast to traditional forecasting applications, modern incarnations often exhibit

[*]Equal contribution.
[†]Work done while at AWS.

large panels of related time series, all of which need to be forecasted simultaneously [89]. Although these problem characteristics make them amenable to deep learning or neural networks (NNs), as in many other domains over the course of history, NNs were not always a standard tool to tackle such problems. Indeed, their effectiveness has historically been regarded as mixed (e.g., [202]).

The history of NNs starts in 1957 [152] and in 1964 for NNs in forecasting [83]. Since then, interest in NNs has oscillated, with upsurges in attention attributable to breakthroughs. The application of NNs in time series forecasting has followed the general popularity, typically with a lag of a few years. Examples of such breakthroughs include Rumelhart et al. [153, 154] that popularized the training of multilayer perceptrons (MLPs) using back-propagation. Significant advances were made subsequently such as the use of convolutional NNs (CNNs) [113], and Long Short Term Memory (LSTM) [81] cells that address the issue of recurrent NNs' (RNNs) training, just to name a few. Despite these advances, NNs remained hard to train and difficult to work with. Methods such as Support Vector Machines (SVMs) [26] and Random Forests [79] that were developed in the 1990s proved to be highly effective (LeCun et al. [114] found that SVMs were as good as the best designed NNs available at the time) and were supported by attractive theory. This shifted the interest of researchers away from NNs. Forecasting was no exception and results obtained with NNs were mostly mixed as reflected in a highly cited review [202]. The breakthrough that marked the dawn of the deep learning era came in 2006 when Hinton et al. [78] showed that it was possible to train NNs with a large number of layers (deep) if the weights are initialized appropriately. Accordingly, deep learning has had a sizable impact on forecasting [110] and NNs have long entered the canon of standard techniques for forecasting [84]. New models specifically designed for forecasting tasks have been proposed, taking advantage of deep learning to supercharge classical forecasting models or to develop entirely novel approaches. This recent burst of attention on *deep forecasting* models is the latest twist in a long and rich history.

Driven by the availability of (closed-source) large time series panels, the potential of deep forecasting models, i.e., forecasting models based on NNs, has been exploited primarily in applied industrial research divisions over the last years [111, 64, 156, 190].[3] With the overwhelming success of deep forecasting methods in the M4 competition [169], this has convinced also formerly skeptical academics [128, 129]. In the most recent M5 competition, deep forecasting methods were the second and third placed solutions [130] although the competition was otherwise dominated by tree-based forecasting methods such as LightGBM [99] and XGBoost [33], see e.g., [92]. Modern software frameworks [1, 143, 34] have sped up the development of NN models and dedicated forecasting packages available [4].

While the history of NNs for forecasting is rich, the focus of this article is on more recent developments in NN for forecasting, roughly since the time that the term "deep learning" was coined. As such, we do not attempt to give a complete historical overview and sacrifice comprehensiveness for recency. The main objectives of this article are to educate on, review and popularize the recent developments in forecasting driven by NNs for a general audience. Therefore, we place emphasis on an educational aspect via a tutorial of deep forecasting in the first part (Section 2). In the second part, Section 3, we provide an overview of the state-of-the-art of modern deep forecasting models. Our exposition is driven by an attempt to identify the main building blocks of modern deep forecasting models which hopefully enables the reader to digest the rapidly increasing literature more easily. We do not attempt a taxonomy of all existing methods and our selection of the building blocks is opinionated, motivated by our experience of innovating in this area with a strong focus on practical applicability. Compared with other surveys [119, 76, 202], we provide a more comprehensive overview with a particular focus on recent, advanced topics. Finally, in Section 4, we conclude and speculate on potentially fruitful areas for future research.

2 Deep Forecasting: A Tutorial

In the following, we formalize the forecasting problem, summarize those advances in deep learning that we deem as the most relevant for forecasting, expose important building blocks for NNs and discuss archetypal models in detail. For general improvements that fueled the deep learning renaissance, like weight initialization, optimization algorithms or general-purpose components such as activation functions, we refer to standard textbooks like [71]. We are aware to be opinionated in both the selection of topics as well as the style of exposition. We attempt to take a perspective akin to a deep forecasting model builder who would compose a forecasting model out of several building blocks such as NN architectures, input transformations and output representations. Although not all models will fit perfectly into this exposition, it is our hope that this downside is outweighed by the benefit of allowing the inclined reader to invent new models more easily.

[3] Forecasting is an example of a sub-discipline in the machine learning community where the comparatively modest attention it receives in published research is in stark contrast to a tremendous business impact.

2.1 Notation and Formalization of the Forecasting Problem

Matrices, vectors and scalars are denoted by uppercase bold, lowercase bold and lowercase normal letters, i.e., $\mathbf{X}$, $\mathbf{x}$ and x, respectively. Let $\mathcal{Z} = \{\mathbf{z}_{i,1:T_i}\}_{i=1}^{N}$ be a set of N univariate time series, where $\mathbf{z}_{i,1:T_i} = (z_{i,1}, \ldots, z_{i,T_i})$, $z_{i,t}$ is the value of the i-th time series at time t and $\mathbf{Z}_{t_1:t_2}$ the values of all N time series at the time slice $[t_1, t_2]$. Typical examples for the domain of the time series values include $\mathbb{R}, \mathbb{N}, \mathbb{Z}, [0, 1]$. The set of time series is associated with a set of covariate vectors denoted by $\mathcal{X} = \{\mathbf{X}_{i,1:T_i}\}_{i=1}^{N}$, with $\mathbf{x}_{i,t} \in \mathbb{R}^{d_x}$. Note that each vector $\mathbf{x}_{i,t}$ can include both time-varying or static features. We denote by α a general input in a model (that can be any combination of covariates and lagged values of the target) and by β a general output. Since α and β refer to a general case, we always represent them with lowercase normal letters. We denote by θ the parameters of a model (e.g., parameters of a distribution) and by Φ the learnable free parameters of the underlying NN (e.g., the weights and biases).

In the most general form, the object of interest in forecasting is the conditional distribution

$$p(\mathbf{Z}_{t+1:t+h}|\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}; \theta), \tag{1}$$

where θ are the parameters of a (probabilistic) model. Eq. (1) is general in the sense that each $\mathbf{z}_i \in \mathcal{Z}$ is multidimensional (the length of the time series), $\mathcal{Z}$ is multivariate (the number of time series $|\mathcal{Z}| = N > 1$) and the forecast is multi-step (h steps). Varying degrees of simplification of Eq. (1) are considered in the literature, for example by assuming factorizations of p and different ways of estimating θ. In the following, we present the three archetypical models for addressing Eq. (1).

Local univariate model: A separate (*local*) model is trained independently for each of the N time series, modelling the predictive distribution

$$p(\mathbf{z}_{i,t+1:t+h}|\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}; \theta_i), \quad \theta_i = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}), \tag{2}$$

where Ψ is a generic function mapping input features to the parameters θ_i of the probabilistic model that are local to the i-th time series. Note that one may use multidimensional covariates $\mathbf{x}_{i,t}$ for each of the N models, but they are still solving a univariate problem, i.e., forecasting only one time series. The use of covariates common to all N models is possible but any pattern that is learned in one model is not used in another (unless provided explicitly which prohibits parallel training). Many classical approaches fall into this category and traditionally NNs were employed in this local fashion (e.g., [202]). Note that this approach is not suitable for cold start problems: i.e., forecasting a time series without historical values.

Global univariate model: A single, *global* model [91, 135] is trained using available data from all N time series. However, the model is still used to predict a univariate target. It does not produce joint forecasts of all time series but forecasts of any single time series at a time. This is also sometimes referred to as a cross-learning approach, e.g., [161]. In a more general form, global univariate models specialize Eq. (1) to

$$p(\mathbf{z}_{i,t+1:t+h}|\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}; \theta_i), \quad \theta_i = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}, \Phi), \tag{3}$$

where Φ are shared parameters among all N time series.

In this article, Ψ in global models is usually a NN and $\mathbf{X}_i$ include item-specific features to allow the model to distinguish between the time series. Although the parameters θ_i of the probabilistic model for each time series are different, they are still predicted using shared parameters (or weights) Φ in Ψ. This allows for efficient learning since the model pools information from all time series and in particular improves inference for shorter time series compared to local univariate models. Such a model is expected to learn some advanced features ("embeddings") exploiting information across time series. Once these advanced features are learned via Ψ, the global model is then used to forecast each time series *independently*. That is, although during training the model sees all the related time series together, the prediction is done by looking at each time series individually. Note that the embeddings learned in the global model are useful beyond the N time series used in the training. This addresses the cold start problem in the sense that the global model can be used to provide forecasts for time series without historical values. Global models are also referred to as cross-learning or panel models in econometrics and statistics and have been the subject of considerable study, e.g., via dynamic factor models [66].

Multivariate model: Here, a single model is learned for all N time series using all available data, directly predicting the *multivariate* target:

$$p(\mathbf{Z}_{t+1:t+h}|\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}; \theta), \quad \theta = \Psi(\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}, \Phi). \tag{4}$$

Note that the model also learns the dependency structure among the time series. Technically speaking, Eq. (4) is a global multivariate model and a further distinction from local multivariate models, such as VARMA [125], is possible.

Table 1: Summary of deep forecasting models based on forecast and model type. For one-step and multi-step forecasting models $h = 1$ and $h > 1$, respectively.

Forecast type	Model type	Formulation
Point	Local univariate	$\hat{\mathbf{z}}_{i,t+1:t+h} = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h})$
	Global univariate	$\hat{\mathbf{z}}_{i,t+1:t+h} = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}, \Phi)$
	Multivariate	$\hat{\mathbf{Z}}_{t+1:t+h} = \Psi(\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}, \Phi)$
Probabilistic	Local univariate	$P(\mathbf{z}_{i,t+1:t+h}\|\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}; \theta_i), \quad \theta_i = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h})$
	Global univariate	$P(\mathbf{z}_{i,t+1:t+h}\|\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}; \theta_i), \quad \theta_i = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}, \Phi)$
	Multivariate	$P(\mathbf{Z}_{t+1:t+h}\|\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}; \theta), \quad \theta = \Psi(\mathbf{Z}_{1:t}, \mathbf{X}_{1:t+h}, \Phi)$

Remarks Note that in Eq. (1) and in the following model-specific cases we have chosen the multi-step ahead predictive distribution. We can always obtain a multi-step predictive distribution via a rolling one-step predictive distribution. In our discussion so far, we presented *probabilistic forecast* models that learn the entire distribution of the future values. However, it may be desirable to model specific values such as the mean, median or some other quantile, instead of the whole probability distribution. These are called *point-forecast* models and the optimal choice of the summary statistics to turn a probabilistic forecast into a point forecast depends on the metric used to judge the quality of the point forecast [104]. More concretely, a point-forecast global univariate model learns a quantity $\hat{\mathbf{z}}_{i,t+1:t+h} = \Psi(\mathbf{z}_{i,1:t}, \mathbf{X}_{i,1:t+h}, \Phi)$, where $\hat{\mathbf{z}}_{i,t+1:t+h}$ is some point estimate of the future values of the time series. Table 1 summarizes the various modelling option based on the forecast and model types.

2.2 Neural Network Architectures

NNs are compositions of *differentiable* functions formed from simple building blocks to learn an approximation of some unknown function from data. An NN is commonly represented as a directed acyclic graph consisting of nodes and edges. The edges between the nodes contain weights (also called parameters) that are learned from the data. The basic unit of every NN is a neuron (illustrated in Fig. 1a), consisting of an input, an affine transformation with learnable weights and (optionally) a nonlinear activation function. Different types of NNs arrange these components in different ways. We refer to other reviews [119] for more details on the main architectures. Here, we only offer a high-level summary for completeness, focusing instead on forecasting specific ingredients for NNs such as input processing and loss functions.

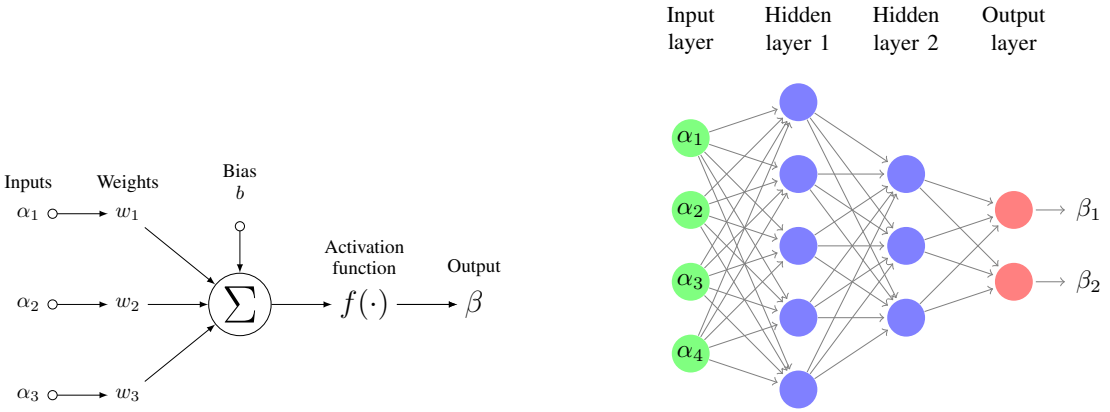
2.2.1 Multilayer perceptron

In multilayer perceptrons (MLPs) or synonymously feedforward NNs, layers of neurons are stacked on top of each other to learn more complex nonlinear representations of the data. An MLP consists of an input and an output layer, while the intermediate layers are called *hidden*. The nodes in each layer of the network are fully connected to all the nodes in the previous layer. The output of the last hidden layer can be seen as some nonlinear feature representation (also called an *embedding*) obtained from the inputs of the network. The output layer then learns a mapping from these nonlinear features to the actual target. Learning with MLPs, and more generally with NNs, can be thought of as the process of learning a nonlinear feature map of the inputs and the relationship between this feature map and the actual target. Figure 1b illustrates the structure of an MLP with two hidden layers. Modern incarnations of the MLP have added important details to alleviate problems like vanishing gradients [80]. For example, ResNet [75], contains direct connections between hidden layers $\ell - 1$ and $\ell + 1$, skipping over the hidden layer ℓ.

One of the main limitations of MLPs is that they do not exploit the structure often present in the data in applications such as computer vision, natural language processing and forecasting. Moreover, the number of inputs and outputs is fixed making them inapplicable to problems with varying input and output sizes as in forecasting. Next, we discuss more complex architectures that overcome these limitations, for which MLPs are often used as the basic building blocks.

2.2.2 Convolutional neural networks

Convolutional neural networks (CNNs) [112] are a special class of NNs that are designed for applications where inputs have a known ordinal structure such as images and time series [71]. CNNs are *locally* connected NNs that use *convolutional* layers to exploit the structure present in the input data by applying a convolution function to smaller neighborhoods of the input data. Convolution here refers to the process of computing moving weighted sums by



(a) Single node (b) MLP

Figure 1: (a) Structure of a single node or neuron. An affine transformation is applied to the input followed by an activation function, i.e., $\beta = f\left(\sum \alpha_i w_i + b\right)$. The weights and bias parameters are learned during training. (b) Illustration of the MLP structure. Each circle in the hidden and output layers is a node, i.e., it applies an affine transformation followed by a nonlinear activation to the set of its inputs.

sliding the so-called *filter* or *kernel* over different parts of the input data. The size of the filter as well as how the filter is slid across the input are part of the hyperparameters of the model. A nonlinear activation, typically ReLU [68], is then applied to the output of the convolution operation.

In addition to convolutional layers, CNNs also typically use a *pooling* layer to reduce the size of the feature representation as well as to make the features extracted from the convolutional layer more robust. For example, a commonly used max-pooling layer, which is applied to the output of convolutional layers, extracts the maximum value of the features in a given neighborhood. Similarly to the convolution operation, the pooling operation is applied to smaller neighborhoods by sliding the corresponding filter over the input. A pooling layer, however, does not have any learnable weights and hence both the convolution and the pooling layer are counted as one layer in CNNs.

Of particular importance for forecasting are the so-called *causal* convolutions, defined as

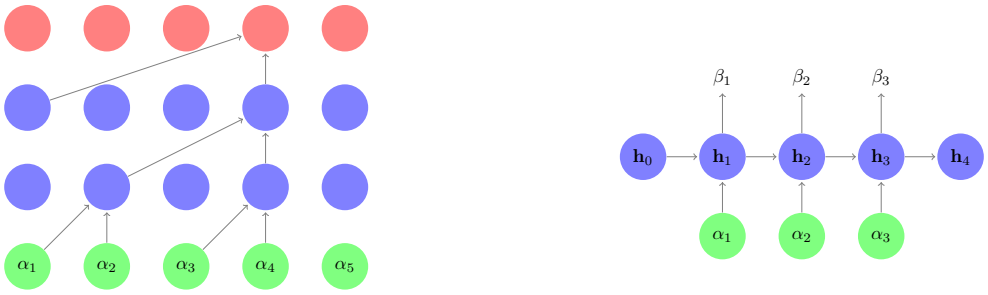
$$h_j = \sum_{d \in \mathcal{D}} w_d \alpha_{j-d} \,,$$

where h_j is the output of a hidden node, α denotes the input, $\mathcal{D} = \{1, \ldots, n\}$ for some n, $|\mathcal{D}|$ is the *width* of the causal convolution (or also called the receptive field) and $\mathbf{w}$ are the learnable parameters. In other words, causal convolutions are weighted moving averages which only take inputs into account which are before j hence the reference to causality in its name. A variation are *dilated* causal convolutions where we vary the index set $\mathcal{D}$, e.g., such that it does not necessarily contain consecutive values, but only every k-th value. Typically, these dilated causal convolutions are stacked on top of each other where the output of one layer of dilated causal convolutions is the input of another layer of causal convolution and the dilation grows by the depth of the NN. Figure 2a illustrates the general structure of a CNN with dilated causal convolutions.

2.2.3 Recurrent neural networks

Recurrent neural networks (RNNs) are NNs specifically designed to handle sequential data that arise in applications such as time series, natural language processing and speech recognition. The core idea consists of connecting *recurrently* the NNs' hidden units back to themselves with a time delay [94, 95]. Since hidden units learn some kind of feature representations of the raw input, feeding them back to themselves can be interpreted as providing the network with a dynamic memory. One crucial detail here is that the same network is used for all timesteps, i.e., the weights of the network are shared across timesteps. This weight-sharing idea is similar to that in CNNs where the same filter is used across different parts of the input. This allows the RNNs to handle sequences of varying length during training and, more importantly, generalize to sequence lengths not seen during training. Figure 2b illustrates the general structure of an (unrolled) RNN.

Although RNNs have been widely used in practice, training them is difficult given that they are typically applied to long sequences of data. A common issue while training very deep NNs by gradient-based methods using back-propagation is that of vanishing or exploding gradients [142] which renders learning challenging. Hochreiter and



(a) CNN (b) RNN

Figure 2: (a) Structure of a CNN consisting of a stack of three causal convolution layer. The input layer (green) is non-dilated and the other two are dilated. (b) Structure of an unrolled RNN. At each timestep t the network receives an external input α_t and the output of the hidden units from the previous time step $\mathbf{h}_{t-1}$. The hidden units all share the same weights. The internal state of the network is updated to $\mathbf{h}_t$ that is going to play the role of the previous state in the next timestep $t+1$. Finally, the network outputs β_t which is a function of α_t and $\mathbf{h}_t$.

Schmidhuber [81] proposed Long short-term memory networks (LSTM) to address this problem. Similar to Resnet, via the skip-connections, LSTMs (and a simplified version Gated recurrent units (GRU) [36]) always offer a path where the gradient does not vanish or explode.

2.2.4 Transformer

A more recent architecture is based on the *attention* mechanism which has received increased interest in other sequence learning tasks [37, 38, 184, 116] for its ability to improve on long sequence prediction tasks over RNNs. One natural way to address this issue is to learn more than one feature representation (contrary to RNNs), e.g., one for each time step of the input sequence and decide which of these representations are useful to predict the current element of the target sequence. Bahdanau et al. [10] suggest using a weighted sum of the representations where the weights are jointly learned along with the feature representation learning and the prediction. Note that at each time step in the prediction, one needs to learn a separate set of weights for the representations. This is essentially training the predictor to learn to which parts of the input sequence it should pay *attention* to produce a prediction. This attention mechanism has been shown to be instrumental for the state of the art in speech recognition and machine translations tasks [37, 38]. Inspired by the success of attention models, Vaswani et al. [184] developed the so-called *Transformer* model and showed that attention alone is sufficient, thus making the training amenable for parallelization and large number of parameters [28, 44]. In the literature, the term Transformer can refer to both the specific model and to the overall architecture as well.

2.3 Input Transformations

The careful handling of the input (parameters α_t in Fig. 1 and 2) is a practically important ingredient for deep learning models in general and deep forecasting models in particular. Deep forecasting models are most commonly deployed as so-called global models (see Section 2.1), which means that the weights of the NN are trained across the panel of time series. Hence, it is important that the scale of the input is comparable. Standard techniques such as mean-variance scaling carry over to the forecasting setting. In practice, it is important to avoid leakage of future values in normalization schemes, so that mean and variance are taken over past windows (similar to causal convolutions).

Traditionally, the forecasting literature has used transformations such as the Box-Cox, i.e.,

$$h = \frac{z^{\lambda} - 1}{\lambda}, \tag{5}$$

where z is the input of the transformation, h is the output and λ is a free parameter. Box-Cox is a popular heuristic to have the input data more closely resemble the Gaussian distribution. A Box-Cox transformation can be readily integrated into an NN, with the free parameter λ optimized as part of the training process jointly with the other parameters of the network. More sophisticated approaches based on probability integral transformation (PIT) or Copulas are similarly possible, see e.g., [88] (and references therein) for a recent example.

A further standard technique is the discretization of input into categorical values or *bins*, for example by choosing the number and borders of bins such that each bins contains equal mass, see e.g., [145] for an example in forecasting.

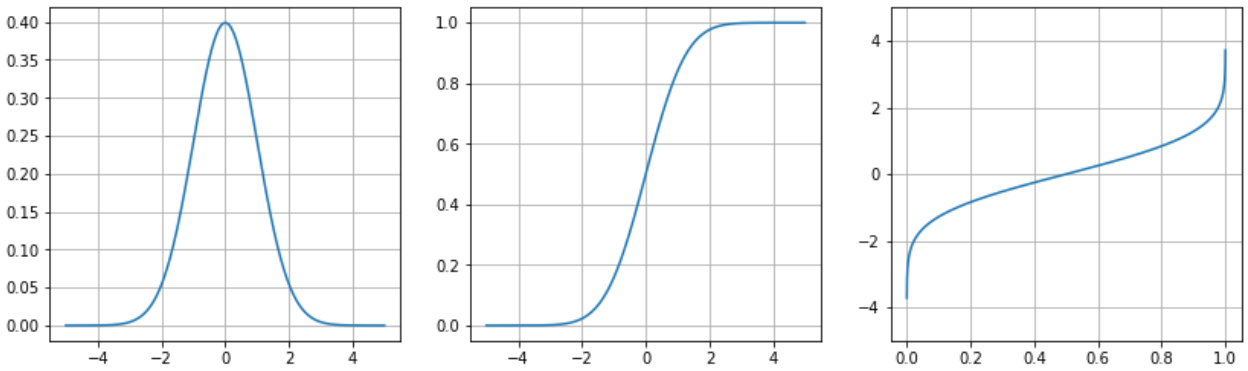


Figure 3: For a Gaussian distribution, its density function f is on the left-hand panel, the corresponding cumulative density function F (the primitive integral of f) in the central panel and the quantile function F^{-1} on the right-hand panel.

We note that any input transformation must be reversed also to obtain values in the actual domain of interest. It is a choice for the modeller where/when to apply this reversal. Two extreme choices are to have transformation of the input and output fully outside the NN or have the input transformations as part of the NN and hence be subjected to learning.

2.4 Output Models and Loss Functions

Similar to the input, the output (β_t in Fig. 1 and 2) deserve a special discussion. Closely related is the question on the choice of loss function which we use to train a NN. The simplest form of an output is a single value, also referred to as a *point* forecast. For this case, the output $\hat{z}_{i,t}$ is the best (w.r.t. the chosen loss function) estimate for the true value $z_{i,t}$. Standard regression loss functions (like ℓ_p losses with their regularized modifications) can be used or more sophistication accuracy metrics specifically geared towards forecasting such as the MASE, sMAPE or others [85].

As remarked in Section 2.1, a point estimate $\hat{z}_{i,t}$ can be seen as a particular realization from a probabilistic estimate of $p(z_{i,t})$. Depending on the accuracy metric used in forecasting, a different realization may be appropriate [104]. So, even for obtaining point forecasts, *probabilistic* forecasts are important. More importantly, forecasts are often used in downstream optimization problem where some form of *expected* cost is to be minimized and for this, an estimate of the entire probability distribution is required. The probability distribution can be represented equivalently by its probability density function (PDF), the cumulative density function (CDF) or its inverse, the quantile function. Fig. 3 contains a visualization of the different representations for the Gaussian distribution. Across the deep forecasting landscape, most approaches (e.g., [156, 64, 150, 147, 146]), have chosen the PDF and quantile function to represent $p(z_{i,t})$ and we will discuss general recipes next. Since the CDF has typically not been chosen to represent $p(z_{i,t})$, we do not discuss it further.

2.4.1 PDF

Arguably the most common way to represent a probability distribution in forecasting is via its PDF. The literature contains examples of using the standard parametric distribution families to represent probabilistic forecasts. For example, the output layer of an NN may produce the mean and variance parameter of a Gaussian distribution. So, the parameter β_t in Fig. 1 and 2 is a two-dimensional vector corresponding to μ_t and σ_t of a Gaussian distribution. We typically achieve $\sigma_t \geq 0$ by mapping the corresponding parameter through a softplus function. For the loss function, a natural choice is the negative log-likelihood (NLL) since a PDF allows to readily compute the likelihood of a point under it.

Beyond Gaussian likelihood, a number of differentiable parametric distributions have been used in the literature depending on the nature of the forecasting problem, e.g., the student-t distribution or the Tweedie distribution for continuous data, the negative binomial distribution for count data and more flexible approaches via mixtures of Gaussian. Although forecasting is most commonly done for domains of numerical values (i.e., we assume $z_{i,t}$ to be in $\mathbb{R}$ or $\mathbb{N}$), other distributions such as the multinomial have also been employed successfully in forecasting even though they have no notion of the order on the domain [145]. The deployment of a multinomial distribution requires a binning of the input values (see Section 2.3). An alternative approach is to cut the output space in bins and treat each of them as a uniform distribution, while modelling the tails with a parametric distribution [50], this results in a piecewise linear CDF.

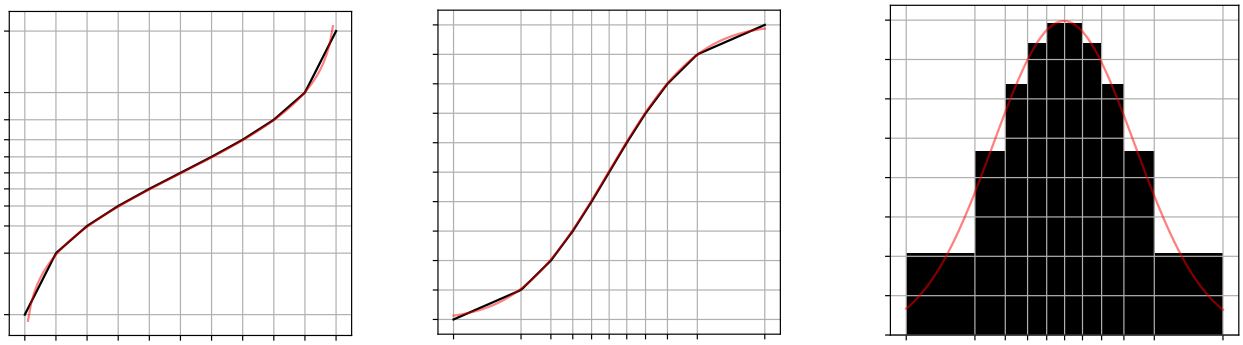


Figure 4: An illustration how a quantile function parametrized by linear splines (left panel) corresponds to a piece-wise linear CDF (middle) which in turn corresponds to a piece-wise constant PDF as assumed in an adaptive binning strategy (right panel).

2.4.2 Quantile function

Another representation of $p(z_{i,t})$ is via the quantile function which has a particular importance for forecasting. Often, a particular quantile is of practical interest. For example, in a simplified supply chain scenario for inventory control, there is a direct correspondence between the chosen quantile and a safety stock level in the newsvendor problem [54].

So naturally, estimating the quantiles directly via quantile regression approaches [103] is a common choice in forecasting either via choosing a single quantile (in a point-forecasting approach) or multiple quantiles simultaneously [51, 190]. Essentially, this discretizes the quantile function and estimates specific points only. A common choice for the loss function is the quantile loss or pinball loss. For the q-th quantile and F^{-1} the quantile function, the quantile loss is defined as

$$\mathrm{QS}_q\left(\hat{F}^{-1}_{i,t}(q), z_{i,t}\right) := 2\left(\mathbb{1}_{\{z_{i,t}\leq \hat{F}^{-1}_{i,t}(q)\}} - q\right)\left(\hat{F}^{-1}_{i,t}(q) - z_{i,t}\right), \tag{6}$$

where $\mathbb{1}_{\{cond\}}$ is the indicator function that is equal to 1 if cond is true and 0 otherwise. The output of the NN is $\hat{F}^{-1}_{i,t}(q)$, i.e., the estimated value of the q-th quantile. For $q = 0.5$ this reduces to the median of the forecast distribution and is a common choice of point forecasts.

As an alternative to a quantile regression approach, we can make a parametric assumption on the quantile function and estimate it directly. The main requirements for modelling a quantile function are that its domain should be constrained to $[0, 1]$ and the function should be monotonically increasing. This can be achieved easily via linear splines for example, so the output of the NN's last layers are the corresponding free parameters. For the loss function, a rich theory around the continuous ranked probability score (CRPS) exists [132, 69] and CRPS can be used as a loss function directly. CRPS can be defined [108] to summarize all possible quantile losses as

$$\mathrm{CRPS}(\hat{F}_{i,t}, z_{i,t}) := \int_0^1 \mathrm{QS}_q\left(\hat{F}^{-1}_{i,t}(q), z_{i,t}\right)\, dq. \tag{7}$$

Multivariate extensions such as the energy score [69] exist.

Interestingly, a popular discretization strategy, adaptive binning, used with multinomial distributions corresponds to quantile functions parametrized by piece-wise linear splines, see Fig. 4.

2.4.3 Further approaches

The recent deep learning literature contains more advanced examples for density estimation, most prominently via Generalized Adversarial Networks (GANs). We discuss them in Section 3.8 and discuss *normalizing flows* here which have arguably resonated more strongly in forecasting. Normalizing flows are invertible NNs that transform a simple distribution to a more complex output distribution. Invertibility guarantees the conservation of probability mass and allows the evaluation of the associated density function everywhere. The key observation is that the probability density of an observation $z_{i,t}$ can be computed using the change of variables formula:

$$p(z_{i,t}) = p_{y_{i,t}}(f^{-1}(z_{i,t}))|\det[\mathrm{Jac} f^{-1}_{i,t} z_{i,t}]|, \tag{8}$$

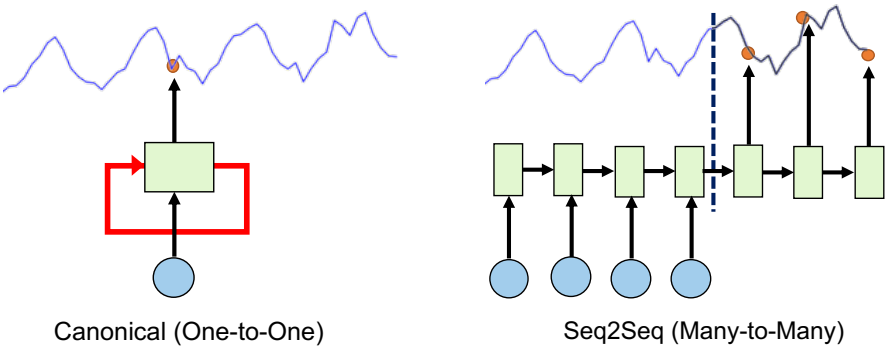


Figure 5: Canonical versus sequence-to-sequence models.

where the first term $p_{y_{i,t}}(f^{-1}(z_{i,t}))$ is the (in general simple) density of a variable $y_{i,t}$, and the second is the absolute value of the determinant of the Jacobian of $f_{i,t}^{-1}$, evaluated at $z_{i,t}$.

The invertible function f is typically parametrized by an NN. A particular instantiation is the Box-Cox transformation, Eq. (5). The field of normalizing flows (e.g., [46, 101, 137]) studies invertible NNs that typically transform isotropic Gaussians to more complex data distributions. The choice of a particular instantiation of f can facilitate the computation of the likelihood of a given point when the NLL is amenable as a loss function. Alternatively, generating samples may be computationally more viable for other instantiations (this is typically the cases with generative adversarial networks as well). In this case, the NLL can be replaced by other loss functions such as CRPS.

A number of extensions are possible. For example, more complex models for $p(z_{i,t})$ are possible such as Hidden Markov Models or Linear Dynamical Systems. NNs can output the free parameters of these models but then need to be combined with the learning and inference schemes associated with these models, such as Kalman Filtering/Smoothing in the case of Linear Dynamical System [146, 42] or the Forward/Backward Algorithm in the case of Hidden Markov Models [5]. Another avenue is to relax constraints on the representation of $p(z_{i,t})$ to obtain closely related objects with more favorable computational properties. For example, *energy based models* (EBMs) approximate the unnormalized log-probability [115, 77]. EBMs perform well in learning high dimensional distributions at the cost of being difficult to train [174] and have been employed in forecasting [151].

2.5 Archetypical Architectures

With all key components in place, in this section we present in more details popular forecasting architectures. In particular we focus on the widely-used RNN-based architecture that takes as input its previous hidden state, the currently available information and produces an one-step ahead estimate of the target time series. There are subtle details on how to handle a multi-step unrolled model during training (e.g., [109]), which we will skip over. We further examine the sequence-to-sequence (seq2seq) modelling approach where the model takes an *encoding sequence* as input and maps it to a *decoding* sequence (of predetermined length) on which the loss is computed against the actual values $\mathbf{z}$ during training. A typical instance in the training set in this approach consists of the target and covariate values up to a certain point in time t as the encoding sequence and the outputs of the NN are a predetermined number of target values after time t. Figure 5 contrasts both approaches. In the following we present two popular deep forecasting models, DeepAR and MQRNN/MQCNN, in some details to illustrate the core concepts. They represent the one-step-ahead RNN-based and seq2seq approach, respectively.

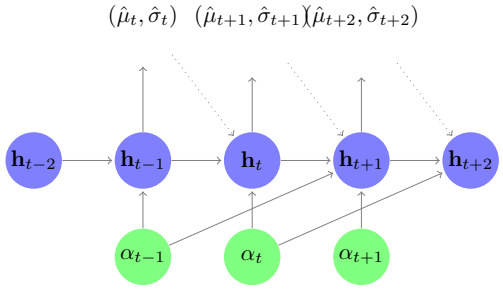


Figure 6: DeepAR: The model outputs parameters of a previously chosen family of distributions. Samples from this distribution can be fed back into the model during prediction (dotted lines) or in case of α_t being missing.

2.5.1 DeepAR

Among the first of the modern deep forecasting models is DeepAR [156], a global univariate model (see Table 1)

that consists of an RNN backbone (typically an LSTM).[4] The input of the model is a combination of lagged target values and relevant covariates. The output is either a point forecast with a standard loss function or, in the basic variant, a probabilistic forecast via the parameters of a PDF (e.g., μ and σ of a Gaussian distribution), where the loss function is then the NLL. The output modelling of DeepAR has been the subject of follow-up work, e.g., Jeon and Seong [93] propose a Tweedie loss, Mukherjee et al. [136] propose a mixture of Gaussians as the distribution and domain specific feature processing blocks. Figure 6 summarizes the architecture. The dotted arrows in the picture correspond to drawing a sample that can be used as alternative input (as a lagged target) during training (even though α_t may be available or in the case where an α_t is missing) and during prediction to obtain multi-step ahead forecasts.

It is also possible to change the output of DeepAR to model the quantile function and use CRPS, Eq. (7), as the loss function [64, 96, 72]. While this in general computationally challenging, special cases are amendable for practical computation. For example, we can assume a parametrization of the quantile function by linear isotonic regression splines:

$$s(q; \gamma, b, d) = \gamma + \sum_{\ell=0}^{L} b_\ell (q - d_\ell)_+ \tag{9}$$

where $q \in [0, 1]$ is the quantile level, $\gamma \in \mathbb{R}$ is the intercept term, $b \in \mathbb{R}^{L+1}$ are weights describing the slopes of the function pieces, $d \in \mathbb{R}^{L+1}$ is a vector of knot positions, L the number of pieces of the spline and $(x)_+ = \max(x, 0)$ is the ReLU function. In order for $s(\cdot)$ to represent a quantile function we need to guarantee its monotonicity and restrict its domain to $[0, 1]$. Both of these constraints can readily be achieved using standard NN tooling using a reparametrization of Eq. (9), while CRPS can be solved in closed form for linear splines (see [64]).

Bag of tricks. While the general setup of DeepAR is straightforward, a number of algorithmic optimizations turn it into a robust, general-purpose forecasting model. The handling of missing values via sample replacement from the probability distribution is one such example. Another one is oversampling of "important" training examples during training, where importance typically corresponds to time series with larger absolute values. Adding lagged values further help improve predictive accuracy. Lags can be chosen heuristically based on the frequency of the time series. For example, in a time series with daily frequency, a lag of 7 days often helps. Similarly, covariates corresponding to calendar events (e.g., indicator variables for weekends or holidays) can help further.

2.5.2 MQRNN/MQCNN

As an example for another type of deep forecasting model, we discuss the multi-horizon quantile recurrent forecaster (MQRNN) [190] next which was conceived concurrently to DeepAR. Contrary to DeepAR, it is most naturally deployed as a discriminative, seq2seq model in a quantile regression setting. For each time point t in the forecast horizon, MQRNN outputs a chosen number of estimates for corresponding quantiles and the loss function in MQRNN is Eq. (6), i.e., the pinball loss summed over all quantiles and time points.

While MQRNN can use multiple configurations, often a CNN-based architecture is chosen in practice in the encoder (MQCNN) for computational efficiency reasons over RNN-based methods and two MLPs in the decoder. The first MLP captures all inputs during the forecast horizon and the context provided by the encoder. A second, local MLP applies only to specific horizons for which it uses the corresponding available input and the output of the MLP. A further innovation provided by MQCNN is the training scheme via the so-called *forking sequences* where the model forecasts by placing a series of decoders with shared parameters at each timestep in the encoder. Thus, the model can structurally forecast at each timestep, while the optimization process is stabilized by updating the gradients from the sequences together. An additional component of MQRNN is a local MLP component that aims to model spikes and events specifically.

3 Literature review

In the prior section, we provided an in-depth introduction to selected, basic topics. Building on these topics, we survey the literature on modern deep forecasting models more broadly in this section. Given the breadth of the literature available, our selection is necessarily subjective.

We proceed as follows. In Section 3.1 we present probabilistic forecasting models, both one-step and multi-step. Similarly, in Section 3.2 we summarize point forecast models. We remark that, after Section 2, we have recipes at hand to turn an one-step ahead forecasting model into a multi-step forecasting model and a point forecasting model into a

[4] Hewamalage et al. [76] provide an overview specifically targeted at RNNs for forecasting.

probabilistic model. We discuss hybrids of deep learning with state space models in Section 3.3, multivariate forecasting in Section 3.4, physics-based model in Section 3.5, global-local models in Section 3.6, models for intermittent time series in Section 3.7 and generative adversarial networks for forecasting in Section 3.8. We close this section with an overview of the large number of available models in Section 3.9 where we also provide guidelines on where to start the journey with deep forecasting models.

3.1 Probabilistic Forecast Models

3.1.1 One-step forecast

The DeepAR model presented in Sec. 2.5.1, is an example of one-step canonical forecasting model. In its base variant, DeepAR is a global univariate model which learns a univariate distribution; we discuss multivariate extensions in Sec. 3.4. DeepAR can be equipped with outputs representing a parametrized PDF including Gaussian Mixture Distributions Mukherjee et al. [136] or quantile functions Gasthaus et al. [64].

Rasul et al. [151] propose TimeGrad which, like DeepAR, is an RNN model using LSTM or GRU cells for which samples are drawn from the data distribution at each time step, with the difference that in TimeGrad the RNN conditions a diffusion probabilistic model [172] which allows the model to easily scale to multivariate time series and accurately use the dependencies between dimensions. Replacing the RNN-backbone of DeepAR with dilated causal convolutions has been proposed as both point and probabilistic forecasting models [20, 4, 183].

3.1.2 Multi-step forecast

Contrary to the some of the models in Section 3.1.1 which produce one-step ahead forecasts, multi-step forecasts can be obtained directly with a seq2seq architecture. In Section 2.5.2, we reviewed the MQRNN/MQCNN architecture [190] as a seq2seq architecture for probabilistic forecasting. The main advantage of seq2seq over one-step ahead forecast models is that the decoder architecture can be chosen to output all future target values at once. This removes the need to unroll over the forecast horizon which can lead to *error accumulation* since early forecast errors propagate through the forecast horizon. Thus, the decoder of seq2seq forecasting models is typically an MLP while other architectures are also used for the encoder [190, 138].

Wen and Torkkola [189] extended the MQCNN model with a generative quantile copula. This model learns the conditional quantile function that maps the quantile index, which is a uniform random variable conditioned on the covariates, to the target. During training, the model draws the quantile index from a uniform distribution. This turns MQCNN into a generative, marginal quantile model. The authors combine this approach with a Gaussian copula to draw correlated marginal quantile index random values. They show that the Gaussian copula component improves the forecast at the distribution tails. Chen et al. [35] proposed DeepTCN, another seq2seq model where the encoder is the dilated causal convolution with residual blocks, and the decoder is simply an MLP with residual connections. Structure-wise, DeepTCN is almost the same as the basic structure of MQCNN [190], i.e., without the local MLP component that aims to model spikes and events.

Park et al. [141] propose the incremental quantile functions (IQF), a flexible and efficient distribution-free quantile estimation framework that resolves quantile crossing with a simple NN layer. A seq2seq encoder-decoder structure is used although the method can be readily applied to recurrent models with one-step ahead forecasts [156]. IQF is trained using the CRPS loss (Eq. (7)) similar to [64].

A combination of recurrent and encoder-decoder structures has also been explored. In [205], the authors use an LSTM with Monte Carlo dropout as both the encoder and decoder. However, unlike other models that directly use RNNs to generate forecasts, the learned embedding at the end of the decoding step is fed into an MLP prediction network and is combined with other external features to generate the forecast. Along a similar line, Laptev et al. [111] employ an LSTM as a feature extractor (LSTM autoencoder), and use the extracted features, combined with external inputs to generate the forecasts with another LSTM.

Van Den Oord et al. [183] introduced the WaveNet architecture, a generative model for speech synthesis, which uses dilated causal convolutions to learn the long range dependencies important for audio signals. Since this architecture is based on convolutions, training is very efficient on GPUs – prediction is still sequential and further changes are necessary for fast inference. Adaptations of WaveNet for forecasting are available [4].

3.2 Point Forecast Models

Point forecast models do not model the probability distribution of the future values of a time series but rather output directly a point forecast that typically corresponds to a summary statistic of the predictive distribution. We have

discussed generic recipes on how to turn a point forecasting model into a probabilistic forecasting model in Section 2.4 and the literature contains further examples (see e.g., [74, 175] for recent complementary approaches).

3.2.1 One-step forecast

A considerable amount of attention of the community is dedicated to one-step forecasting. LSTNet [107] is a model using a combination of a CNN and an RNN. Targeting multivariate time series, LSTNet uses a convolution network (without pooling) to extract short-term temporal patterns as well as correlations among variables. The output of the convolution network is fed into a recurrent layer and a temporal attention layer which, combined with the autoregressive component, generates the final point forecast. While LSTNet uses a standard point forecast loss function, it can readily be turned into a probabilistic forecast model using the components described in Sec. 2, e.g., by modifying LSTNet to output the parameters of a probability distribution and using NLL as a loss function. Qiu et al. [144] proposed an ensemble of deep belief networks for forecasting. The outputs of all the networks is concatenated and fed into a support vector regression model (SVR) that gives the final prediction. The NNs and the SVR are not trained jointly though. Hsu [82] proposed an augmented LSTM model which combines autoencoders with LSTM cells. The input observations are first encoded to latent variables, which is equivalent to feature extraction, and are fed into the LSTM cells. The decoder is an MLP which maps the LSTM output into the predicted values. For point forecast multivariate forecasting, Yoo and Kang [198] proposed time-invariant attention to learn the dependencies between the dimensions of the time series and use them with a convolution architecture to model the time series.

Building upon the success of CNNs in other application domains, Borovykh et al. [24] proposed an adjustment to WaveNet [183] that makes it applicable to conditional forecasting. They evaluated their model on various datasets with mixed results, concluding that it can serve as a strong baseline and that various improvements could be made. In a similar vein, inspired by the Transformer architecture [184] Song et al. [173] proposed an adjustment that makes the architecture applicable to time series. Their method is applied to both regression and classification tasks.

3.2.2 Multi-step forecast

N-BEATS [138] is an NN architecture purpose-built for the forecasting task that relies on a deep, residual stack of MLP layers to obtain point forecasts. The basic building block in this architecture is a forked MLP stack that takes the block input and feeds the intermediate representation into separate MLPs to learn the parameters of the context (the authors call it backcast) and forecast time series models. The residual architecture removes the part of the context signal it can explain well before passing to the next block and adds up the forecasts. The learned time series model can have free parameters or be constrained to follow a particular, functional form. Constraining the model to trend and seasonality functional forms does not have a big impact on the error and generates models whose stacks are interpretable since the trend and seasonality components of the model can be separated and analyzed. N-BEATS has also been interpreted as a meta-learning model [139], where the repeated application of residual blocks can be seen as an inner optimization loop. N-BEATS generalizes better than other architectures when trained on a source dataset (e.g., M4-monthly) and applied to a different target datasets (e.g., M3-monthly).

Lv et al. [126] propose a *stacked autoencoder* (SAE) architecture to learn features from spatio-temporal traffic flow data. On top of the autoencoder, a logistic regression layer is used to output predictions of the traffic flow at all locations in a future time window. The resulting architecture is trained layer-wise in a greedy manner. The experimental results show that the method significantly improves over other shallow architectures, suggesting that the SAE is capable of extracting latent features regarding the spatio-temporal correlations of the data. In the same context of spatio-temporal forecasting and under the seq2seq framework, Li et al. [118] proposed the Diffusion Convolutional Recurrent NN (DCRNN). Diffusion convolution is employed to capture the dependencies on the spatial domain, while an RNN is utilized to model the temporal dependencies. Finally, Asadi and Regan [6] proposed a framework where the time series are decomposed in an initial preprocessing step to separately feed short-term, long-term, and spatial patterns into different components of a NN. Neighbouring time series are clustered based on their similarity of the residuals as there can be meaningful short-term patterns for spatial time series. Then, in a CNN based architecture, each kernel of a multi-kernel convolution layer is applied to a cluster of time series to extract short-term features in neighbouring areas. The output of the convolution layer is concatenated by trends and is followed by a convolution-LSTM layer to capture long-term patterns in larger regional areas.

Bandara et al. [13] addressed the problem of predicting a set of disparate time series, which may not be well captured by a single global model. For this reason, the authors propose to cluster the time series according to a vector of features extracted using the technique from [87] and the Snob clustering algorithm [186]. Only then, an RNN is trained per cluster, after having decomposed the series into trend, seasonality and residual components. The RNN is followed by an affine neural layer to project the cell outputs to the dimension of the intended forecast horizon. This approach is applied to publicly available datasets from time series competitions, and appears to consistently improve against

learning a single global model. In subsequent work, Bandara et al. [15] continued to mix heuristics, in this instance seasonality decomposition techniques, known from classical forecasting methods with standard NN techniques. Their aim is to improve on scenarios with multiple seasonalities such as inter and intra daily. The findings are that for panels of somewhat unrelated time series, such decomposition techniques help global models whereas for panels of related or homogeneous time series this may be harmful. The authors do not attempt to integrate these steps into the NN architecture itself, which would allow for end-to-end learning.

Cinar et al. [39] proposed a content attention mechanism that seats on top of any seq2seq RNN. The idea is to select a combination of the hidden states from the history and combine them using a pseudo-period vector of weights to the predicted output step.

Li et al. [116] introduce two modifications to the Transformer architecture to improve its performance for forecasting. First, they include causal convolutions in the attention to make the key and query context dependent, which makes the model more sensitive to local contexts. Second, they introduce a sparse attention, meaning the model cannot attend to all points in the history, but only to selected points. Through exponentially increasing distances between these points, the memory complexity can be reduced from quadratic to $O(T(\log T)^2)$, where T is the sequence length, which is important for long sequences that occur frequently in forecasting. Other architectural improvements to the Transformer model have also been used more recently to improve accuracy and computational complexity in forecasting applications. For example, Lim et al. [120] introduce the Temporal Fusion Transformer (TFT), which incorporates novel model components for embedding static covariates, performing "variable selection", and gating components that skip over irrelevant parts of the context. The TFT is trained to predict forecast quantiles, and promotes forecast interpretability by modifying self-attention and learning input variable importance. Eisenach et al. [51] propose MQ-Transformer, a Transformer architecture that employs novel attention mechanisms in the encoder and decoder separately, and consider learning positional embeddings from event indicators. The authors discuss the improvements not only on forecast accuracy, but also on excess forecast volatility where their model improves over the state of the art. Finally, Zhou et al. [204] recently proposed the Informer, a computationally efficient Transformer architecture, that specifically targets applications with long forecast horizons. The Informer introduces a *ProbSparse* attention layer and a distilling mechanism to reduce both the time complexity and memory usage of learning to $O(T \log T)$, while improving forecast performance over deep forecasting benchmark.

3.3 Deep State Space Models

In contrast to pure deep learning methods for time series forecasting introduced in Section 2, Rangapuram et al. [146] propose to combine classical state space models (SSM) [49, 86] with deep learning. The main motivation is to bridge the gap between SSMs that provide a principled framework for incorporating structural assumptions but fail to learn patterns across a collection of time series, and NNs that are capable of extracting higher order features but results in models that are hard to interpret. Their method parametrizes a linear Gaussian SSM using an RNN. The parameters of the RNN are learned jointly from a dataset of raw time series and associated covariates. Instead of learning the SSM parameters $\theta_{i,1:T_i}$ for the i-th time series individually or locally in the terminology of Section 2.1), the model is global and learns a shared mapping from the covariates associated with each target time series to the parameters of a linear SSM. This mapping $\theta_{i,t} = f(\mathbf{x}_{i,1:t}; \Phi)$, for $i = 1, \ldots, N$ and $t = 1, \ldots, T_i + h$, is implemented by an RNN with weights Φ which are shared across different time series as well as different time steps. Note that f depends on the entire covariate time series up to time t as well as the set of shared parameters Φ. Since each individual time series i is modelled using an SSM with parameters Θ_i, assumptions such as temporal smoothness in the forecasts are easily enforced. The shared model parameters Φ are learned by maximizing the likelihood given the observations $\mathcal{Z} = \{\mathbf{z}_{i,1:T_i}\}_{i=1}^N$. The likelihood terms for each time series reduce to the standard likelihood computation under the linear-Gaussian SSM, which can be carried out efficiently via Kalman filtering [16]. Once the parameters Φ are learned, it is straightforward to obtain the forecast distribution via the SSM parameters $\theta_{i,T_i+1:T_i+h}$.

There are two major limitations of the method proposed in Rangapuram et al. [146]: first, the observations are assumed to follow a Gaussian distribution and second, the underlying latent process that generates observations is assumed to evolve linearly. de Bézenac et al. [42] address the first limitation via *Normalizing Kalman Filters* (NKF) by augmenting SSMs with normalizing flows [46, 101, 137] thereby giving them the flexibility to model non-Gaussian, multimodal data. Their main idea is to map the non-Gaussian observations $\{\mathbf{z}_{i,1:T_i}\}$ to more Gaussian-like observations via a sequence of *learnable*, nonlinear transformations (e.g., a normalizing flow) so that the method in [146] can then be applied on the transformed observations. While being more flexible, their method still retains attractive properties of linear Gaussian SSMs, namely, tractability of exact inference and likelihood computation, efficient sampling, and robustness to noise.

In a concurrent work to [42], Kurle et al. [106] improve the method in [146] by addressing both limitations. In particular, to model nonlinear latent dynamics, they propose a recurrent switching Gaussian SSM, which uses additional latent

variables to switch between different linear dynamics. Moreover, to handle non-Gaussian observations, they propose a nonlinear emission model via a decoder-type NN [61]. Although the exact inference is no longer tractable with these improvements, they show that the approximate inference and likelihood estimation can be Rao-Blackwellised; i.e., the inference for the Gaussian latent states can be done exactly while the inference for the switch variables needs to be approximated.

Finally, Ansari et al. [5] propose to extend [146] via incorporating switching dynamics. The recurrent explicit duration switching dynamical system (RED-SDS) is a flexible model that is capable of identifying both state- and time-dependent switching dynamics of a time series. State-dependent switching is enabled by a recurrent state-to-switch connection and an explicit duration count variable is used to improve the time-dependent switching behavior. A hybrid algorithm that approximates the posterior of the continuous states via an inference network and performs exact inference for the discrete switches and counts provides efficient inference. The method is able to infer meaningful switching patterns from the data and extrapolate the learned patterns into the forecast horizon.

3.4 Multivariate Forecasting

The models presented up to this point are mainly global univariate models, i.e., they are trained on all time series but they are still used to predict a univariate target. When dealing with multivariate time series, one should be able to exploit the dependency structure between the different time series in the panel in a generalization of Eq. (3) to Eq. (4).

Toubeau et al. [179] and Salinas et al. [155] combined RNN-based models with copulas to model multivariate distributions. The model in [179] uses a nonparametric copula to capture the multivariate dependence structure. In contrast, the work in [155] uses a Gaussian copula process approach. Salinas et al. [155] use a low-rank covariance matrix approximation to scale to thousands of dimensions. Additionally, the model implements a non-parametric transformation of the marginals to deal with varying scales in the dimensions and non-Gaussian data. More recently, Rasul et al. [150] proposed to represent the data distribution with a type of normalizing flows called Masked Autoregressive Flows [140] while using either an RNN or a Transformer [184] to model the multivariate temporal dynamics of time series. Normalizing flows were also used to bring deep SSMs [146] to a flexible, multivariate scenario [42]. Rasul et al. [151] propose TimeGrad which, like DeepAR, is an RNN model for which samples are drawn from the data distribution at each time step, with the difference that in TimeGrad the RNN conditions a diffusion probabilistic model [172] which allows the model to easily scale to multivariate time series and accurately use the dependencies between dimensions.

A recent application of global multivariate models is for hierarchical forecasting problems [17, 191, 176, 7]. Typically, in such problems an aggregation structure is defined (e.g., via a product hierarchy) and a trade-off between forecast accuracy and forecast coherency with respect to the aggregation structure must be managed. Here, forecast coherency or consistency means that the forecasts conforms to the aggregation structure, so that aggregated forecasts are the same as forecasts of aggregated time series. This aggregation structure is typically encoded via linear constraints where the aggregation structure is captured in a matrix S. Rangapuram et al. [147] propose to use a multivariate model such as [155] and enforce consistency of forecast samples via incorporation of a projection of the samples with S into the learning problem. Dedicated work exists for aggregation along the time dimension [178, 8, 148].

In some multivariate forecasting settings the different dimensions are tied together by some interpretable connections other than a hierarchy and this can be modelled as part of the input layer rather than the output as discussed so far. One can for example think of forecasting the traffic network of a city where the traffic at each of the location in the city is mostly influenced by the traffic at the neighboring locations, like in PEMS-BAY and METR-LA [118]. Graph Neural Networks (GNN) have been used in this forecasting setting [163, 43, 193, 102, 63, 206] where, in addition to the forecasting task, the challenge is to best use the graph information that is provided or even learn the graph if none is available. The methods that propose to learn the graph do so by looking for the graph that allows to produce the most accurate forecasts. An embedding is learned for each dimension, and similarity scores are computed between every two dimension using these embeddings from which the adjacency matrix is obtained, either by taking the K-top edges [43, 193] or sampling from them [163, 102]. As of now, two main strategies have been proposed to learn the node embeddings, either simply by gradient descent [43, 193] or by taking representation from the time series [163, 102], with the latter approach to seemingly yielding better results. While these methods were all presented as point forecasting method, one could obtain probabilistic forecasts by training these models to parametrize a predictive distribution as explained in Section 2.4.

3.5 Physics-based Models

In *physics-based* models, deep forecasting methods have been proposed that model the underlying dynamics in sophisticated ways. Chen et al. [32] proposed the Neural ODE (NODE) model, where an ordinary differential equation (ODE) is solved forward in time, and the adjoint equation is solved backwards in time using backpropagation. One

limitation of the Neural ODE model is that the unknown parameters θ are assumed to be constant in time. Other limitations such as computational complexity have been addressed in follow-up work, e.g., [18]. Vialard et al. [185] extends the NODE model to allow the parameters $\theta(t)$ to be time-varying by introducing a shooting formulation. In the shooting formulation, the optimal θ is determined by minimizing a regularized loss function. Vialard et al. [185] also shows that a residual network (ResNet) can be expressed as the Forward Euler discretization of an ODE with time step $\Delta t = 1$. Wang et al. [187] compares successful time series deep sequence models, such as [156, 146] to NODE and other hybrid deep learning models to model COVID-19 dynamics, as well as the population dynamics using the Lotka-Volterra equations. Through their benchmarking study, the authors show that distribution shifts can pose problems for deep sequence models on these tasks, and propose a hybrid model AutoODE to model the underlying dynamics.

3.6 Global-local

With *local models*, the free parameters of the model are learned individually for each series in a collection, see Section 2.1. Classical local time series models such as SSMs, ARIMA, and exponential smoothing (ETS) [84] excel at modelling the complex dynamics of individual time series given a sufficiently long history. Other local models include Gaussian SSMs, which are computationally efficient, e.g., via a Kalman filter, and Gaussian Processes (GPs) [149, 159, 67, 27]. These methods provide uncertainty estimates, which are critical for optimal downstream decision making. Since these methods are local, they learn one model per time series and cannot effectively extract information across multiple time series. These methods are unable to address cold-start problems where there is a need to generate predictions for a time series with little or no observed history.

Conversely, recall that in *global models*, their free parameters are learned jointly on every series in a collection of time series. NNs have proven particularly well suited as global models [156, 64, 146, 190, 111]. Global methods can extract patterns from collections of irregular time series even when these patterns would not be distinguishable using a single series.

Global-local models have been proposed to combine the advantages of both global and local models. Examples include mixed effect models [40], which consist of two kinds of effects: fixed (global) effects that describe the whole population, and random (local) effects that capture the idiosyncratic of individuals or subgroups. A similar mixed approach is used in hierarchical Bayesian [65] methods, which combine global and local models to jointly model a population of related statistical problems. In an early example of hierarchical Bayesian models, [31] combined global and local features for intermittent demand forecasting in retail planning. In [3, 123], other combined global and local models are detailed.

A recent global-local family of models, Deep Factors [188] provide an alternative way to combine the expressive power of NNs with the data efficiency and uncertainty estimation abilities of classical probabilistic local models. Each time series, or its latent function for non-Gaussian data, is represented as the weighted sum of a global time series and a local model. The global part is given by a linear combination of a set of deep dynamic factors, where the loading is temporally determined by attentions. The local model is stochastic. Typical choices include white noise processes, linear dynamical systems, GPs [127] or RNNs. The stochastic local component allows for the uncertainty to propagate forward in time, while the global NN model is capable of extracting complex nonlinear patterns across multiple time series. The global-local structure extracts complex nonlinear patterns globally while capturing individual random effects for each time series locally.

The Deep Global Local Forecaster (DeepGLO) [162] is a method that "thinks globally and acts locally" to forecast collections of up to millions of time series. It crucially relies on a type of temporal convolution (a so-called leveled network), that can be trained across a large amount time series with different scales without the need for normalization or rescaling. DeepGLO is a hybrid model that uses a global matrix factorization model [200] regularized by a temporal deep leveled network and a local temporal deep level network to capture patterns specific to each time series. Each time series is represented by a linear combination of k basis time series, where $k \ll N$, with N the total number of time series. The global and local models are combined through data-driven attention for each time series.

A further example in the global-local model class is the ES-RNN model proposed by Smyl [169] that has recently attracted attention by winning the M4 competition [129] by a large margin on both evaluation settings. In the ES-RNN model, locally estimated level and trend components are multiplicatively combined with an RNN model. Apart from its global-local nature, it also integrates aspects of different model classes into a a single model similar to Deep State Space models (Section 3.3). In particular, the h-step ahead prediction $\hat{\mathbf{z}}_{i,t+1:t+h} = l_{i,t} \cdot \mathbf{s}_{i,t+1:t+h} \cdot \exp(\text{RNN}(\mathbf{x}_{i,t}))$ consists of a level $l_{i,t}$ and a seasonal component $s_{i,t}$ obtained through local exponential smoothing, and the output of a global RNN model $\text{RNN}(\mathbf{x}_{i,t})$, where $\mathbf{x}_{i,t}$ is a vector of preprocessed data extracted from deseasonalized and normalized time series $\mathbf{x}_{i,t} = \log(\mathbf{z}_{i,t-K:t}/(\mathbf{s}_{i,t-K:t} l_{i,t}))$ cut in a window of length $K + 1$. The RNN models are

composed of dilated LSTM layers with additional residual connections. The M4-winning entry used slightly different architectures for the different type of time series in the competition.

3.7 Intermittent Time Series

We noted in the introduction that deep forecasting models had a major impact on operational forecasting problems. In these large-scale problem, intermittent time series occur regularly [25]. Accordingly, research on NNs for intermittent time series forecasting has been an active area. Salinas et al. [156] propose a standard RNN architecture with a negative binomial likelihood to handle intermittent demand similar to [171] in classical methods. To the best of our knowledge, other likelihoods that have been proposed for intermittent time series in classical models, e.g., by [160], have not yet been carried over to NNs. However, some initial work is available via more standard likelihoods [156, 93].

In the seminal paper on intermittent demand forecasting [41], Croston separates the data in a sequence of observed non-zero demands and a sequence of time intervals between positive demand observations, and runs exponential smoothing separately on both series. A comparison of NNs to classical models for intermittent demand first appeared in Gutierrez et al. [73], where the authors compare the performance of a shallow and narrow MLP with Croston's method. They find NNs to outperform classical methods by a significant margin.

Kourentzes [105] proposes two MLP architectures for intermittent demand, taking demand sizes and intervals as inputs. As in Gutierrez et al. [73], the networks are shallow and narrow by modern standards, with only a single hidden layer and three hidden units. The difference between the two architectures is in the output. In one case interval times and non-zero occurrences are output separately, while in the other a ratio of the two is computed. The approach proposed by Kourentzes [105] outperforms other approaches primarily with respect to inventory metrics, but not forecasting accuracy metrics, challenging previous results in [73]. It is unclear whether the models are used as global or local. However, given the concern around overfitting and regularization, we assume that these models were primarily used as local models in the experiments.

Both approaches of [73, 105] only offer point forecasts. This shortcoming is addressed by [180, 182], where the authors propose renewal processes as natural models for intermittent demand forecasting. Specifically, they use RNNs to modulate both discrete time and continuous time renewal processes, using the simple analogy that RNNs can replace exponential smoothing in [41].

Finally, a recent trend in sequence modelling employs NNs in modelling discrete event sequences observed in continuous time [48, 133, 194, 164, 181, 165] and [166] for an overview. Notably, Xiao et al. [195] use two RNNs to parametrize a probabilistic "point process" model. These networks consume data from asynchronous event sequences and uniformly sampled time series observations respectively. Their model can be used in forecasting tasks where time series data can be enriched with discrete event observations in continuous time.

3.8 Generalized Adversarial Networks

Additionally to the approaches mentioned in Sections 2.4 and 2.4.3, the recent literature contains further examples for density estimation, most prominently via Generalized Adversarial Networks (GANs) [70]. While GANs have received much attention in the overall deep learning literature [98, 199, 52, 121, 53], this has not been reflected in forecasting. We speculate that this is because a discriminator network can be replaced by metrics such as CRPS which measure the quality of generated samples. We therefore only provide a brief overview here and mention that, while they rely on the buildings blocks discussed in Section 2, they typically require architectures that are more complex than then ones discussed here and lead to involved optimization problems.

Despite the comparably less attention that GANs have received in forecasting, they have been recently applied to the time series domain [53, 199] to synthesize data [177, 53] or to employ an adversarial loss in forecasting tasks [192]. Many time series GAN architectures use recurrent networks to model temporal dynamics [134, 53, 199]. Modelling long-range dependencies and scaling recurrent networks to higher lengths is inherently difficult and limits the application of time series GANs to short sequence lengths [199, 53]. One way to achieve longer realistic synthetic time series is by employing convolutional [183, 11, 62] and self-attention architectures [184].

Convolutional architectures are able to learn relevant features from the raw time series data [183, 11, 62], but are ultimately limited to local receptive fields and can only capture long-range dependencies via many stacks of convolutional layers. Self-attention can bridge this gap and allow for modelling long-range dependencies from convolutional feature maps, which has been a successful approach in the image [203] and time series forecasting domain [116]. Another technique to achieve long sample sizes is progressive growing, which successively increases the resolution by adding layers to a GAN generator and discriminator during training [97]. A recent proposal [22] synthesizes progressive growing with convolutions and self-attention into a novel architecture particularly geared towards time series.

3.9 Summary and Practical Guidelines

In Section 2 and this section, we introduced a large number of deep forecasting models. We summarize the main approaches in Table 2. The list below provide keys to reading the table.

- *Forecast* distinguishes between probabilistic (*Prob*) and *Point* forecasts.
- *Horizon* indicates whether the model does one-step predictions (noted 1) in which case multi-step forecasts are obtained recursively, or if it directly predicts a whole sequence (≥ 1).
- *Loss* and *Metrics* specifies the loss used for training and metrics used for evaluation. Here, we only provide an explanation of the acronyms and not the definition of each metric which can be easily found in the corresponding papers: negative log-likelihood (NLL), quantile loss (QL), continuous ranked probability score (CRPS), (normalized) (root) mean squared error (NRMSE, RMSE, MSE), root relative squared error (RRSE), relative geometric RMSE (RGRMSE), weighted absolute percentage error (WAPE), normalized deviation (ND), mean absolute deviation (MAD), mean absolute error (MAE), mean relative error (MRE), (weighted) mean absolute percentage error (wMAPE, MAPE), mean absolute scaled error (MASE), overall weighted average (OWA), mean scaled interval score (MSIS), Kullback-Leibler divergence (KL), Value-at-Risk (VaR), expected shortfall (ES), empirical correlation coefficient (CORR), area under the receiver operating characteristic (AUROC), percentage best (PB).

While Table 2 serves to illustrate the wealth of deep forecasting methods now available, their sheer number may be slightly overwhelming. Furthermore, empirical evidence on the effectiveness of the different architectures has so far not revealed a clearly superior approach [4]. In this, forecasting differs from other domains, e.g., natural language processing where Transformer-based models [184] dominate overall. Also, deep forecasting methods seem to differ from other model families, such as tree-based methods where LightGBM [99] or XGBoost [33] dominate (as in the recent M5 forecasting competition [92]). We speculate that this diffuse picture is in part due to the practical reasons, the relative immaturity of the field and the corresponding software implementations and in part due to fundamental reason as a natural consequence of the breadth and diversity of forecasting problems.

So, choosing the appropriate architecture for a problem at hand can be a daunting task. In the following, we therefore attempt to provide guidelines for a more informed deep forecasting model selection. These are largely based on our own experience in working with practical forecasting problem and they should primarily be taken as a non-exhaustive guidance on where to start model exploration.

3.9.1 Baseline methods and standard mode of deployment

At the start of any in-depth model exploration, considering a baseline model is commonly accepted best practice. To the best of our knowledge, the most mature deep forecasting models are DeepAR [156] and MQCNN [190] which exist in a number of open-source and commercial implementations.[5] As a practical guideline, we recommend to start model exploration using at least these methods as baselines. Other candidates we would consider are N-BEATS [138], WaveNet [183] and a Transformer-based model. The relative performance of these methods compared with other methods should give reasonable, directional evidence whether the problem at hand is amenable to deep forecasting methods. We note that AutoML approaches for forecasting are available[6] but while promising are in their infancy. At least in the M5 competition, they are still outperformed by the aforementioned more specialized deep forecasting models.

Our typical suggestion is to employ NNs as global models since, given enough data, global methods outperform classical local methods when dealing with groups of similar time series.[7] Interestingly, recent empirical evidence have shown that global models can achieve a state-of-the-art performance even in heterogeneous groups of time series. This is supported by the M4 [129] and M5 competitions where the top performing models had some form of globality. This suggests a more general applicability of global methods with a high impact on practical application where a general automated forecasting mechanism is required.

[5] `https://aws.amazon.com/blogs/machine-learning/now-available-in-amazon-sagemaker-deepar-algorithm-for-more-accurate-time-series-forecasting/`

[6] `http://ai.googleblog.com/2020/12/using-automl-for-time-series-forecasting.html`

[7] This is a more generally applicable fact beyond NN. Montero-Manso and Hyndman [135] show favorable theoretical and empirical properties for global over local models.

Study	Structure	Forecast	Horizon	Loss	Metrics	Data Types	Comments
DeepAR [156]	RNN	Prob	1	NLL	Coverage, QL, ND, NRMSE	demand, traffic, electricity	Learns parametric distributions
Toubeau et al. [179]	RNN/CNN	Prob	1	NLL/QL	RMSE, price	electricity	Nonparametric copula to capture multivariate dependence
Salinas et al. [155]	RNN	Prob	1	NLL	QL, MSE	electricity, traffic, exchange rate, solar, taxi, wiki	Learns multivariate model via low-rank Gaussian copula processes
ARMDN [136]	RNN	Prob	1	NLL	wMAPE	demand	Like [156], but using mixture of Gaussian's and domain specific feature processing
QARNN [196]	MLP	Prob	1	QL	VaR, ES	finance	Conditional quantile function estimation
SQF-RNN [64]	RNN	Prob	1	CRPS	QL, MSIS, NRMSE, OWA	demand, traffic, count data, finance, M4	Models non-parametric distributions with splines
LSTNet [107]	CNN + RNN + MLP	Point	1	ℓ_1	RRSE, CORR	traffic, solar, electricity, exchange rate	Extracts short and long temporal patterns with a CNN and RNN, respectively
Zhu and Laptev [205]	RNN + MLP	Prob	1	-	sMAPE, calibration	daily trips	Fits an encoder (RNN) that constructs an embedding state, which is fed to a prediction network (MLP)
Laptev et al. [111]	RNN	Prob	1	MSE	sMAPE	traffic, M3	LSTM as feature extractor
Qiu et al. [144]	MLP + SVR	Point	1	ℓ_2 for MLP, SVR objective	RMSE, MAPE	energy, housing	Ensemble of DBNs where their output is fed to an SVR
A-LSTM [82]	RNN + MLP	Point	1	ℓ_2, ℓ_2 regularizer	RMSE	electricity consumption	Combination of LSTM with autoencoders
Borovykh et al. [24]	CNN	Point	1	ℓ_1, ℓ_2 regularizer	RMSE, MASE, HITS	index forecasting, exchange rate	WaveNet [183] based model adjusted for time series forecasting
SAnD [173]	MLP + Attention	Point	1	ℓ_2, cross-entropy, multi-label classification loss	AUROC, MASE, MSE	clinical	Transformer [116] based model adjusted for time series forecasting

Zhang [201]	MLP	Point	1	MSE	MSE, MAD	sunspot, lynx, exchange rate	Hybrid local model that uses ARIMA to capture the linear component and a NN for the nonlinear residuals
Khashei and Bijari [100]	MLP	Point	1	MSE	MAE, MSE	sunspot, lynx, exchange rate	Hybrid local model that uses ARIMA and a NN for trend correction
Deep State Space [146]	RNN + State Space	Prob	≥ 1	NLL	P50, P90 quantile loss	traffic, electricity, tourism, M4	RNN parametrized linear Gaussian SSM
NKF [42]	RNN + State Space + Normalizing Flow (NF)	Prob	≥ 1	NLL	QL	traffic, electricity, exchange rate, solar, wiki	RNN parametrized linear Gaussian SSM combined with normalizing flow, which acts as an emission model to handle non-Gaussian data
ARSGLS [106]	Recurrent Switching State Space + NN	Prob	≥ 1	NLL	QL	traffic, electricity, exchange rate, solar, wiki	Recurrent Switching State Space combined with decoder-type NN, which acts as an emission model to handle non-Gaussian data
MQ-RNN/CNN [190]	RNN/CNN + MLP	Prob	$\geq$1	QL	QL, calibration, sharpness	demand	Learns pre-specified grid of quantiles
Wen and Torkkola [189]	CNN + MLP	Prob	$\geq$1	QL, inverse reconstruction loss, NLL	QL, quantile crossing, QL over sum of future intervals	demand	Combines model in [190] with Gaussian copula
DeepTCN [35]	CNN + MLP	Prob	$\geq$1	QL	QL	retail demand	Learns pre-specified grid of quantiles
N-BEATS [138]	MLP	Point	≥ 1	sMAPE, MASE, MAPE	sMAPE, MASE, OWA	M4	Deep, residual MLP that learns interpretable trend and seasonality function
Lv et al. [126]	Stacked autoencoder	Point	≥ 1	MSE, KL sparsity constraint	MAE, MRE, RMSE	traffic	Stacked autoencoders with logistic regression output layer
DCRNN [118]	RNN	Point	≥ 1	NLL	MAE, MAPE, RMSE	traffic	Diffusion convolution for spatial and RNN for temporal dependencies
Asadi and Regan [6]	CNN + RNN	Point	≥ 1	ℓ_2	MAE, RMSE	traffic	Decomposition-based model for spatio-temporal forecasting

Bandara et al. [13]	RNN + Classical Decomposition	Point	≥ 1	-	sMAPE	CIF2016, NN5	Clusters time series based on set of features and train one model per cluster
LSTM-MSNet [15]	RNN + Classical Decomposition	Point	≥ 1	ℓ_1	sMAPE, MASE	M4, energy	Decomposition based model with multiple seasonal patterns
Cinar et al. [39]	RNN + Attention	Point	≥ 1	ℓ_2, ℓ_2 regularizer	MSE, sMAPE	energy, max temperature, CPU usage, air quality	Attention mechanism on top of RNN
Deep Factors [188]	RNN + GP	Prob	≥ 1	NLL	QL, MAPE	electricity, traffic, taxi, uber	Global RNN and a local GP
DeepGLO [162]	CNN	Point	≥ 1	ℓ_2	WAPE, MAPE, sMAPE	electricity, traffic, wiki	Global matrix factorization regularized by a deep leveled network
ES-RNN [169]	RNN	Point	≥ 1	QL	MASE, sMAPE, MSIS	M4	Locally estimated seasonality and trend and global RNN
Kourentzes [105]	MLP	Point	1	ℓ_2	ME, MAE, service level	intermittent demand	MLP-based intermittent demand model
Attentional Twin RNN [195]	RNN	Prob	1	NLL	MAE	point process data	Event sequence prediction
Gutierrez et al. [73]	MLP	Point	1	ℓ_2	MAPE, RGRMSE, PB	intermittent demand	MLP-based intermittent demand model
Deep Renewal Process [180]	RNN	Prob	≥ 1	NLL	P50, P90 quantile loss	intermittent demand	RNN-based intermittent demand model inspired by point processes
WaveNet [183, 4]	CNN	Prob	≥ 1	NLL	mean opinion score	traffic, electricity, M4	Diluted causal convolutions
Transformer [116]	MLP	Point	1	NLL	QL	electricity, traffic, wind, M4, solar	Transformer with causal convolutions and sparse attention
AttnAR [198]	CNN	Point	1	ℓ_2	RMSE	electricity, traffic, solar, exchange rate	Multivariate forecasting
LSTM MAF [150]	RNN	Prob	≥ 1	NLL	RMSE	electricity, traffic, solar, exchange rate	Multivariate forecasting using normalizing flows

TimeGrad [151]	RNN	Prob	≥ 1	NLL	RMSE	electricity, traffic, solar, exchange rate, taxi, wikipedia	Multivariate forecasting using diffusion models.
TFT [120]	LSTM, MLP	Prob	≥ 1	QL	P50, P90 quantile loss	electricity, traffic, retail, volatility	Modified transformer architecture for improved interpretability
MQ-Transformer [51]	CNN, MLP	Prob	≥ 1	QL	P50, P90, LT-SP	electricity, traffic, retail, volatility, retail demand (proprietary)	Architectural improvements on MQ-RNN/CNN for multi-step forecasting
Informer [204]	CNN, MLP	Point	≥ 1	MSE	MSE, MAE	electricity, weather, sensor data	Sparse and computationally efficient transformer architecture

Table 2: Summary of modern deep forecasting models.

3.9.2 Data characteristics

The amount of data available is among the easiest dimensions in choosing a deep forecasting model. First, NNs require a minimum amount of data to be effective in comparison to other, more parsimoniously parametrized models. This is perhaps the most important factor in successful applications of NNs in forecasting. How much data does one need for a given application? Several important points should be discussed on this question. First, the amount of data is often misunderstood as the *number of time series* but in reality the amount of data typically relates to the *number of observations*. For instance, one may have only one time series but many thousands of observations, as in the case of a time series from a real-time sensor where measurements happen every second for a year, allowing to fit a complex NN [2]. Second, it is probably better to see the amount of data in terms of *information quantity*. For instance, in finance the amount of information of many millions of hourly transactions is limited given the very low signal-to-noise ratio in contrast to a retailer whose products follow clear seasonality and patterns, making it easier to apply deep learning methods. The more structured the data is (e.g., via strong seasonality or knowledge about the underlying process) the better deep forecasting models that incorporate these structures will fare. On the contrary, if the time series are more irregular or short, a more data-driven approach (e.g., via Transformer-based models) will often be preferable. The importance of covariate information for the forecasting problem at hand can further help determine the correct method. Some NN architectures need extensions to include such information while others readily accept them.

From a practical perspective, NNs have been reported to outperform demand forecasting baselines starting from 50000 observations in [156] and from a few hundred observations in load-forecasting [146, 188]. Understanding better these limitation, both theoretically and empirically, is an area of current research and is not yet well understood. See [131] for some current theoretical work on sample complexity of global-local approaches for instance and [23] for empiricial work.

3.9.3 Problem characteristics

The characteristics of the forecasting problem to be solved are natural important decision points. We list a few dimensions to consider here.

One important aspect of a model is its forecast nature, i.e., if it produces probabilistic or point forecasts. The choice of this is highly dependent on the underlying application. To illustrate this we can examine two different forecasting use cases: product demand and CPU utilization. In the former use case one wishes to forecast the future demand of a product in order to take a more informed decision about the stock that is required to have in a warehouse or to optimize the labour planning based on the traffic that is expected. In the latter, the forecast of CPU utilization could be used to identify in a timely manner if a process will fail in order to proactively resolve associated issues, or to detect possible anomalous behaviours that could trigger some root cause analysis and system improvements. Although in both applications a forecast is required, the end goal is different, which changes the requirements of the chosen forecasting model. For example, for product demand the whole distribution of the future demand might be important: one cannot rely on a single forecast value since the variance in the forecast plays an important role to avoid out of stock issues or under/over planning the expected required labour. Therefore, in this application it is important to use a model that focuses on predicting accurately the whole distribution. On the other hand, for CPU utilization one might be interested in the 99-th percentile, since everything below that threshold might not be of particular interest or does not produce any actionable alarm. In this case, a model that focuses on a particular quantile of importance is of higher interest than a model that predicts the whole distribution with possibly worse accuracy on the selected quantile.

It is observed [156, 146, 155, 42, 106] empirically that autoregressive models are superior in performance (in terms of forecast accuracy) compared to state space models, especially when the data is less noisy and the forecast horizon is not too long. This is not surprising given that the autoregressive models directly use past observations as input features and treat own predictions as lag inputs in the multi-step forecast setting. A general rule of thumb is that if one knows details such as the forecast horizon, the quantile to query or the exact goals of the forecasting problem in advance and these are unlikely to change, then a discriminative model is often a good default choice. Conversely, state space models proved to be robust when there are missing and/or noisy observations [42]. Moreover, if the application-specific constraints can be incorporated in the latent state, then state space models usually perform better even in the low-data regimes [146].

The length of the forecast horizon relative to the history or, more generally speaking, the importance of the historic values for future values must further be taken into account. For example, very long forecast horizons may require to control (e.g., via differential equations) the exponential growth in the target. A canonical example for this is forecasting of a pandemic. This example further clarifies the importance of being able to produce counterfactuals for what-if analysis (e.g., the incorporation of intervention). Not all deep forecasting models allow for this.

3.9.4 Other Aspects

A number of other aspects can further help to narrow the model exploration space. For example, computational constraints (how much time/money for training is available, are there constraints on the latency during inference) can favor "simpler" NNs, see e.g., [23] for a discussion on multi-objective forecasting model selection. Another aspect to consider could be CNN over RNN-based architectures. The skill set of the research team available is an important factor. For example, probabilistic models often are more sensitive towards parametrization and identifying reasonable parameter ranges requires in-depth knowledge. On the other extreme, troubleshooting Transformer-based models requires deep learning experience that not every research team may possess. The time budget available for model development and the willingness to extend existing models are further factors.

4 Conclusions and Avenues for Future Work

This article has attempted to provide an introduction to and an overview of NNs for forecasting or deep forecasting. We began by providing a panorama of some of the core concepts in the modern literature on NNs chosen by their degree of relevance for forecasting. We then reviewed the literature on recent advances in deep forecasting models.

Deep forecasting methods have received considerable attention in the literature because they excel at addressing forecasting problems with many related time series and at extracting weak signals and complex patterns from large amounts of data. From a practical perspective, the availability of efficient programming frameworks helps to alleviate many of the pain points that practitioners experience with other forecasting methods such as manual feature engineering or the need to derive gradients. However, NNs are not a silver bullet. For many important classes of forecasting problems such as long-range macro-economic forecasts or other problems requiring external domain knowledge not learnable from the data, deep forecasting methods are not the most appropriate choice and will likely never be. Still, it is our firm belief that NNs belong to the toolbox of every forecaster, in industry and academia.

Building onto the existing promising work in NNs for forecasting, many challenges remain to be solved. We expect that the current trends of hybridizing existing time series techniques with NNs [146, 169, 64, 180] and bringing innovations from other related areas or general purpose techniques to forecasting [70, 183, 184] will continue organically. Typical general challenges for NNs, such as data effectiveness, are important in forecasting and likely need a special treatment (see [59] for an approach in time series classification with transfer learning). Other topics of general ML interest such as interpretability, explainability and causality (e.g., [19, 122, 158]) are of particular practical importance in the forecasting setting. It is our hope that original methods such as new NN architectures will be pioneered in the time series prediction sector (e.g., [138]) and that those will then feed back into the general NN literature to help solve problems in other disciplines.

Beyond such organic improvements, we speculate that another area in which NNs have had tremendous impact [167, 168] may become important for forecasting, namely deep reinforcement learning. In contrast to current practice, where forecasting merely serves as input to downstream decision problems (often mixed-integer nonlinear stochastic optimization problems), for example to address problems such as restocking decisions, reinforcement learning allows to directly learn optimal decisions in business context [90]. It will be interesting to see whether reinforcement based approaches can improve decision making – and how good forecasting models could help improve reinforcement approaches.

As methodology advances, so will the applicability. Many potential applications of forecasting methods are underexplored. To pick areas that are close to the authors' interests, in database management, cloud computing, and system operations a host of applications would greatly benefit from the use of principled forecasting methods (see e.g., [21, 9, 60]). Forecasting can also be used to improve core ML tasks such as hyperparameter optimization (e.g., [47]) and we expect more applications to open up in this area.

References

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[3] Amr Ahmed, Moahmed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J Smola. Scalable inference in latent variable models. In *Proceedings of the fifth ACM International Conference on Web Search and Data Mining*, pages 123–132. ACM, 2012.

[4] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Sundar Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[5] Abdul Fatir Ansari, Konstantinos Benidis, Richard Kurle, Ali Caner Turkmen, Harold Soh, Alexander J Smola, Bernie Wang, and Tim Januschowski. Deep explicit duration switching models for time series. *Advances in Neural Information Processing Systems*, 34, 2021.

[6] Reza Asadi and Amelia C Regan. A spatial-temporal decomposition based deep neural network for time series forecasting. *Applied Soft Computing*, 87:105963, 2020.

[7] George Athanasopoulos, Roman A Ahmed, and Rob J Hyndman. Hierarchical forecasts for australian domestic tourism. *International Journal of Forecasting*, 25(1):146–166, 2009.

[8] George Athanasopoulos, Rob J Hyndman, Nikolaos Kourentzes, and Fotios Petropoulos. Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1):60–74, 2017.

[9] Fadhel Ayed, Lorenzo Stella, Tim Januschowski, and Jan Gasthaus. Anomaly detection at scale: The case for deep distributional time series models, 2020.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[11] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[12] Luca Vincenzo Ballestra, Andrea Guizzardi, and Fabio Palladini. Forecasting and trading on the vix futures market: A neural network approach based on open to close returns and coincident indicators. *International Journal of Forecasting*, 35(4):1250 – 1262, 2019. ISSN 0169-2070.

[13] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using long short-term memory networks on groups of similar series. *arXiv preprint arXiv:1710.03222*, 8:805–815, 2017.

[14] Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *International Conference on Neural Information Processing*, pages 462–474. Springer, 2019.

[15] Kasun Bandara, Christoph Bergmeir, and Hansika Hewamalage. Lstm-msnet: Leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[16] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[17] Souhaib Ben Taieb, James W Taylor, and Rob J Hyndman. Coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 3348–3357, 2017.

[18] Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural odes. *Advances in Neural Information Processing Systems*, 34, 2021.

[19] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In *Information Science and Applications (ICISA)*, pages 913–922. Springer, 2016.

[20] Toby Bischoff and Austin Gross. Wavenet & dropout: An efficient setup for competitive forecasts at scale. In *Proceedings of the International Symposium on Forecasting*, 2019.

[21] Michael Bohlke-Schneider, Shubham Kapoor, and Tim Januschowski. Resilient neural forecasting systems. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, DEEM'20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380232.

[22] Michael Bohlke-Schneider, Paul Jeha, Pedro Mercado, Shubham Kapoor, Jan Gasthaus, and Tim Januschowski. PSA-GAN: Progressive self attention gans for synthetic time series. *International Conference on Learning Representations (ICLR)*, 2022.

[23] Oliver Borchert, David Salinas, Valentin Flunkert, Tim Januschowski, and Stephan Günnemann. Multi-objective model selection for time series forecasting. *arXiv preprint arXiv:2202.08485*, 2022.

[24] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[25] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, 10(12):1694–1705, 2017.

[26] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[27] Sofiane Brahim-Belhouari and Amine Bermak. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, 2004.

[28] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[29] Laurent Callot, Mehmet Caner, A Özlem Önder, and Esra Ulaşan. A nodewise regression approach to estimating large portfolios. *Journal of Business & Economic Statistics*, pages 1–12, 2019.

[30] Laurent AF Callot, Anders B Kock, and Marcelo C Medeiros. Modeling and forecasting large realized covariance matrices and portfolio choice. *Journal of Applied Econometrics*, 32(1):140–158, 2017.

[31] Nicolas Chapados. Effective bayesian modeling of groups of related count time series. In *International Conference on Machine Learning*, pages 1395–1403. PMLR, 2014.

[32] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[33] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794. ACM, 2016.

[34] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *NeurIPS Workshop on Machine Learning Systems*, 2015.

[35] Yitian Chen, Yanfei Kang, Yixiong Chen, and Zizhuo Wang. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, 399:491–501, 2020.

[36] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[37] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602*, 2014.

[38] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pages 577–585, 2015.

[39] Yagmur Gizem Cinar, Hamid Mirisaee, Parantapa Goswami, Eric Gaussier, Ali Aït-Bachir, and Vadim Strijov. Position-based content attention for time series forecasting with sequence-to-sequence RNNs. In *International Conference on Neural Information Processing*, pages 533–544, 2017.

[40] Michael J Crawley. Mixed-effects models. *The R Book, Second Edition*, pages 681–714, 2012.

[41] J. D. Croston. Forecasting and stock control for intermittent demands. *Journal of the Operational Research Society*, 23(3):289–303, Sep 1972. ISSN 1476-9360.

[42] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33, 2020.

[43] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada*, pages 2–9, 2021.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[45] I. Dimoulkas, P. Mazidi, and L. Herre. Neural networks for GEFCom2017 probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1409 – 1423, 2019.

[46] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[47] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3460–3468. AAAI Press, 2015. ISBN 978-1-57735-738-4.

[48] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, 2016.

[49] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. Oxford University Press, 2012.

[50] Elena Ehrlich, Laurent Callot, and François-Xavier Aubet. Spliced binned-pareto distribution for robust modeling of heavy-tailed time series. *arXiv preprint arXiv:2106.10952*, 2021.

[51] Carson Eisenach, Yagna Patel, and Dhruv Madeka. Mqtransformer: Multi-horizon forecasts with context dependent and feedback-aware attention. *arXiv preprint arXiv:2009.14799*, 2020.

[52] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. In *International Conference on Learning Representations*, 2018.

[53] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[54] Fotios Petropoulos et al. Forecasting: theory and practice. *International Journal of Forecasting*, 2020.

[55] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Forecasting big time series: old and new. *Proceedings of the VLDB Endowment*, 11(12):2102–2105, 2018.

[56] Christos Faloutsos, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Forecasting big time series: Theory and practice. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*, 2019.

[57] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Classical and contemporary approaches to big time series forecasting. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, New York, NY, USA, 2019. ACM.

[58] Christos Faloutsos, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Forecasting big time series: Theory and practice. In *Companion Proceedings of the Web Conference 2020*, WWW '20, pages 320–321. Association for Computing Machinery, 2020.

[59] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 1367–1376, 2018.

[60] Valentin Flunkert, Quentin Rebjock, Joel Castellon, Laurent Callot, and Tim Januschowski. A simple and effective predictive resource scaling heuristic for large-scale cloud applications. *arXiv preprint arXiv:2008.01215*, 2020.

[61] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[62] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in Neural Information Processing Systems*, 32, 2019.

[63] Victor Garcia Satorras, Syama Sundar Rangapuram, and Tim Januschowski. Multivariate time series forecasting with latent graph inference. *arXiv preprint*, 2022.

[64] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function RNNs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1901–1910, 2019.

[65] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[66] John Geweke. The dynamic factor analysis of economic time series. *Latent variables in socio-economic models*, 1977.

[67] Agathe Girard, Carl Edward Rasmussen, Joaquin Quinonero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, pages 545–552, 2003.

[68] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[69] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[70] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[72] Adèle Gouttes, Kashif Rasul, Mateusz Koren, Johannes Stephan, and Tofigh Naghibi. Probabilistic time series forecasting with implicit quantile networks. *arXiv preprint arXiv:2107.03743*, 2021.

[73] Rafael S. Gutierrez, Adriano O. Solis, and Somnath Mukhopadhyay. Lumpy demand forecasting using neural networks. *International Journal of Production Economics*, 111(2):409–420, February 2008.

[74] Hilaf Hasson, Bernie Wang, Tim Januschowski, and Jan Gasthaus. Probabilistic forecasting: A level-set approach. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[76] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[77] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 08 2002.

[78] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[79] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE, 1995.

[80] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[81] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[82] Daniel Hsu. Time series forecasting based on augmented long short-term memory. *arXiv preprint arXiv:1707.00666*, 2017.

[83] M. J. C. Hu and Halbert E. Root. An adapative data processing system for weather forecasting. *Journal of Applied Metereology*, 1964.

[84] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[85] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688, 2006.

[86] Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with Exponential Smoothing: the State Space Approach*. Springer, 2008.

[87] Rob J Hyndman, Earo Wang, and Nikolay Laptev. Large-scale unusual time series detection. In *IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1616–1619, 2015.

[88] Tim Janke, Mohamed Ghanmi, and Florian Steinke. Implicit generative copulas. *Advances in Neural Information Processing Systems*, 34, 2021.

[89] Tim Januschowski and Stephan Kolassa. A classification of business forecasting problems. *Foresight: The International Journal of Applied Forecasting*, 52:36–43, 2019.

[90] Tim Januschowski, Jan Gasthaus, Yuyang Wang, Syama Sundar Rangapuram, and Laurent Callot. Deep learning for forecasting: Current trends and challenges. *Foresight: The International Journal of Applied Forecasting*, 51:42–47, 2018.

[91] Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. Criteria for classifying forecasting methods. *International Journal of Forecasting*, 2019.

[92] Tim Januschowski, Yuyang Wang, Hilaf Hasson, Timo Erkkila, Kari Torkkila, and Jan Gasthaus. Forecasting with trees. *International Journal of Forecasting*, 2021.

[93] Yunho Jeon and Sihyeon Seong. Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting*, 2021.

[94] Michael I. Jordan. Serial order: A parallel, distributed processing approach. Technical report, Institute for Cognitive Science, University of California, San Diego, 1986.

[95] Michael I. Jordan. Serial order: A parallel, distributed processing approach. In *Advances in Connectionist Theory: Speech*. Erlbaum, 1989.

[96] Kelvin Kan, François-Xavier Aubet, Tim Januschowski, Youngsuk Park, Konstantinos Benidis, Lars Ruthotto, and Jan Gasthaus. Multivariate quantile function forecaster. In *The 25th International Conference on Artificial Intelligence and Statistics*, 2022.

[97] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[98] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[99] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: a highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 2017.

[100] Mehdi Khashei and Mehdi Bijari. A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing*, 11(2):2664–2675, 2011.

[101] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 10236–10245, 2018.

[102] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018.

[103] Roger Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005.

[104] Stephan Kolassa. Why the "best" point forecast depends on the error or accuracy measure. *International Journal of Forecasting*, 36(1):208–211, 2020.

[105] Nikolaos Kourentzes. Intermittent demand forecasts with neural networks. *International Journal of Production Economics*, 143(1):198–206, 2013.

[106] Richard Kurle, Syama Sundar Rangapuram, Emmanuel de Bézenac, Stephan Günnemann, and Jan Gasthaus. Deep rao-blackwellised particle filters for time series forecasting. *Advances in Neural Information Processing Systems*, 33, 2020.

[107] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104. ACM, 2018.

[108] F. Laio and S. Tamea. Verification tools for probabilistic forecasts of continuous hydrological variables. *Hydrology and Earth System Sciences*, 11(4):1267–1277, 2007.

[109] Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[110] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[111] Nikolay Laptev, Jason Yosinsk, Li Li Erran, and Slawek Smyl. Time-series extreme event forecasting with neural networks at Uber. In *ICML Time Series Workshop*. 2017.

[112] Yann LeCun. Generalization and network design strategies. In *Connectionism in perspective*, 1989.

[113] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

[114] Yann LeCun, L.D. Jackel, Leon Bottou, A. Brunot, Corinna Cortes, J. S. Denker, Harris Drucker, I. Guyon, U.A. Muller, Eduard Sackinger, Patrice Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, volume 60, pages 53–60. Perth, Australia, 1995.

[115] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *Predicitng Structured Data*. MIT Press, 2006.

[116] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[117] Xuerong Li, Wei Shang, and Shouyang Wang. Text-based crude oil price forecasting: A deep learning approach. *International Journal of Forecasting*, 35(4):1548 – 1560, 2019.

[118] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[119] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), Feb 2021.

[120] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[121] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. *Advances in Neural Information Processing Systems*, 30, 2017.

[122] Zachary C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57, 2018. ISSN 1542-7730.

[123] Yucheng Low, Deepak Agarwal, and Alexander J Smola. Multiple domain user personalization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 123–131. ACM, 2011.

[124] Rui Luo, Weinan Zhang, Xiaojun Xu, and Jun Wang. A neural stochastic volatility model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[125] Helmut Lütkepohl. Vector autoregressive moving average processes. In *New Introduction to Multiple Time Series Analysis*, pages 419–446. Springer, 2005.

[126] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014.

[127] Danielle C Maddix, Yuyang Wang, and Alex Smola. Deep factors with gaussian processes for forecasting. *arXiv preprint arXiv:1812.00098*, 2018.

[128] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13, 03 2018.

[129] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[130] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Zhi Chen, Anil Gaba, Ilia Tsetlin, and Robert L Winkler. The m5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*, 2021.

[131] Zelda Mariet and Vitaly Kuznetsov. Foundations of sequence-to-sequence modeling for time series. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 408–417. PMLR, 2019.

[132] James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

[133] Hongyuan Mei and Jason M. Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, pages 6754–6764, 2017.

[134] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[135] Pablo Montero-Manso and Rob J Hyndman. Principles and algorithms for forecasting groups of time series: Locality and globality. *arXiv preprint arXiv:2008.00444*, 2020.

[136] Srayanta Mukherjee, Devashish Shankar, Atin Ghosh, Nilam Tathawadekar, Pramod Kompalli, Sunita Sarawagi, and Krishnendu Chaudhury. ARMDN: Associative and recurrent mixture density networks for eretail demand forecasting. *arXiv preprint arXiv:1803.03800*, 2018.

[137] Junier Oliva, Avinava Dubey, Manzil Zaheer, Barnabas Poczos, Ruslan Salakhutdinov, Eric Xing, and Jeff Schneider. Transformation autoregressive networks. In *International Conference on Machine Learning*, pages 3898–3907. PMLR, 2018.

[138] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[139] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. *arXiv preprint arXiv:2002.02887*, 2020.

[140] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

[141] Youngsuk Park, Danielle Maddix, François-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *The 25th International Conference on Artificial Intelligence and Statistics*, 2022.

[142] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1310–1318, 2013.

[143] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[144] Xueheng Qiu, Le Zhang, Ye Ren, Ponnuthurai N Suganthan, and Gehan Amaratunga. Ensemble deep learning for regression and time series forecasting. In *Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, pages 1–6. IEEE, 2014.

[145] Stephan Rabanser, Tim Januschowski, Valentin Flunkert, David Salinas, and Jan Gasthaus. The effectiveness of discretization in forecasting: An empirical study on neural time series models. *arXiv preprint arXiv:2005.10111*, 2020.

[146] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[147] Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 8832–8843. PMLR, 2021.

[148] Syama Sundar Rangapuram, Shubham Shubham Kapoor, Rajbir Nirwan, Pedro Mercado, Tim Januschowski, Yuyang Wang, and Michael Bohlke-Schneider. Coherent probabilistic forecasting for temporal hierarchies, 2022.

[149] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006.

[150] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*, 2020.

[151] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pages 8857–8868. PMLR, 2021.

[152] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[153] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, University of California San Diego, La Jolla Institute for Cognitive Science, 1985.

[154] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[155] David Salinas, Michael Bohlke-Schneider, Laurent Callot, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems*, 2019.

[156] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[157] Harshit Saxena, Omar Aponte, and Katie T. McConky. A hybrid machine learning model for forecasting a billing period's peak electric load days. *International Journal of Forecasting*, 35(4):1288 – 1303, 2019.

[158] Bernhard Schölkopf. Causality for machine learning. *arXiv preprint arXiv:1911.10500*, 2019.

[159] Matthias Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(02): 69–106, 2004.

[160] Matthias W Seeger, David Salinas, and Valentin Flunkert. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, pages 4646–4654, 2016.

[161] Artemios-Anargyros Semenoglou, Evangelos Spiliotis, Spyros Makridakis, and Vassilios Assimakopoulos. Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, 37(3):1072–1084, 2021.

[162] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[163] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021.

[164] Anuj Sharma, Robert Johnson, Florian Engert, and Scott Linderman. Point process latent variable models of larval zebrafish behavior. In *Advances in Neural Information Processing Systems*, pages 10919–10930, 2018.

[165] Oleksandr Shchur, Ali Caner Turkmen, Tim Januschowski, Jan Gasthaus, and Stephan Günnemann. Detecting anomalous event sequences with temporal point processes. *Advances in Neural Information Processing Systems*, 34, 2021.

[166] Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. *arXiv preprint arXiv:2104.03528*, 2021.

[167] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[168] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[169] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

[170] Slawek Smyl and N. Grace Hua. Machine learning methods for GEFCom2017 probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1424–1431, 2019.

[171] Ralph D. Snyder, J. Keith Ord, and Adrian Beaumont. Forecasting the intermittent demand for slow-moving inventories: A modelling approach. *International Journal of Forecasting*, 28(2):485–496, 2012.

[172] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[173] Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[174] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[175] Kamile Stankeviciute, Ahmed M Alaa, and Mihaela van der Schaar. Conformal time-series forecasting. *Advances in Neural Information Processing Systems*, 34, 2021.

[176] Souhaib Ben Taieb, James W Taylor, and Rob J Hyndman. Hierarchical probabilistic forecasting of electricity demand with smart meter data. *Journal of the American Statistical Association*, 116(533):27–43, 2021.

[177] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527:121261, 2019.

[178] Filotas Theodosiou and Nikolaos Kourentzes. Forecasting with deep temporal hierarchies. *Available at SSRN: https://ssrn.com/abstract=3918315 or http://dx.doi.org/10.2139/ssrn.3918315*, 2021.

[179] Jean-François Toubeau, Jérémie Bottieau, François Vallée, and Zacharie De Grève. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34(2):1203–1215, 2018.

[180] Ali Caner Turkmen, Yuyang Wang, and Tim Januschowski. Intermittent demand forecasting with deep renewal processes. *arXiv preprint arXiv:1911.10416*, 2019.

[181] Ali Caner Türkmen, Yuyang Wang, and Alexander J. Smola. Fastpoint: Scalable deep point processes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

[182] Ali Caner Turkmen, Tim Januschowski, Yuyang Wang, and Ali Taylan Cemgil. Forecasting intermittent and sparse time series: A unified probabilistic framework via deep renewal processes. *PlosOne*, 2021.

[183] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *SSW*, 125, 2016.

[184] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[185] François-Xavier Vialard, Roland Kwitt, Suan Wei, and Marc Niethammer. A shooting formulation of deep learning. In *Advances in Neural Information Processing Systems*, 2020.

[186] Chris S Wallace and David L Dowe. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, 2000.

[187] Rui Wang, Danielle Maddix, Christos Faloutsos, Yuyang Wang, and Rose Yu. Bridging physics-based and data-driven modeling for learning dynamical systems. In *Learning for Dynamics and Control*, pages 385–398. PMLR, 2021.

[188] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617, 2019.

[189] Ruofeng Wen and Kari Torkkola. Deep generative quantile-copula models for probabilistic forecasting. *arXiv preprint arXiv:1907.10697*, 2019.

[190] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[191] Shanika L Wickramasuriya, George Athanasopoulos, Rob J Hyndman, et al. Forecasting hierarchical and grouped time series through trace minimization. *Department of Econometrics and Business Statistics, Monash University*, 2015.

[192] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. *Advances in Neural Information Processing Systems*, 33:17105–17115, 2020.

[193] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 753–763, 2020.

[194] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, pages 3247–3257, 2017.

[195] Shuai Xiao, Junchi Yan, Mehrdad Farajtabar, Le Song, Xiaokang Yang, and Hongyuan Zha. Joint modeling of event sequence and time series with attentional twin recurrent neural networks. *arXiv preprint arXiv:1703.08524*, 2017.

[196] Qifa Xu, Xi Liu, Cuixia Jiang, and Keming Yu. Quantile autoregression neural network model with applications to evaluating value at risk. *Applied Soft Computing*, 49:1–12, 2016.

[197] Xing Yan, Weizhong Zhang, Lin Ma, Wei Liu, and Qi Wu. Parsimonious quantile regression of financial asset tail dynamics via sequential learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[198] Jaemin Yoo and U Kang. Attention-based autoregression for accurate and efficient multivariate time series forecasting. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 531–539. SIAM, 2021.

[199] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[200] Hsiang-Fu Yu, Rao N., and I.S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. *Advances in Neural Information Processing Systems*, pages 847–855, 2016.

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

[202] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

[203] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.

[204] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv preprint arXiv:2012.07436*, 2020.

[205] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at Uber. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110, 2017.

[206] Daniel Zügner, François-Xavier Aubet, Victor Garcia Satorras, Tim Januschowski, Stephan Günnemann, and Jan Gasthaus. A study of joint graph inference and forecasting. *arXiv preprint arXiv:2109.04979*, 2021.