PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Review

Cite this article: Lim, B, Zohren S. 2021 Time-series forecasting with deep learning: a survey. *Phil. Trans. R. Soc. A* **379**: 20200209. https://doi.org/10.1098/rsta.2020.0209

Accepted: 28 July 2020

One contribution of 13 to a theme issue 'Machine learning for weather and climate modelling'.

Subject Areas:
artificial intelligence, statistics

Keywords:
deep neural networks, time-series forecasting, uncertainty estimation, hybrid models, interpretability, counterfactual prediction

Author for correspondence:
Bryan Lim
e-mail: blim@robots.ox.ac.uk

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

Time-series forecasting with deep learning: a survey

Bryan Lim and Stefan Zohren

Oxford-Man Institute for Quantitative Finance, Department of Engineering Science, University of Oxford, Oxford, UK

SZ, 0000-0002-3392-0394

Numerous deep learning architectures have been developed to accommodate the diversity of time-series datasets across different domains. In this article, we survey common encoder and decoder designs used in both one-step-ahead and multi-horizon time-series forecasting—describing how temporal information is incorporated into predictions by each model. Next, we highlight recent developments in hybrid deep learning models, which combine well-studied statistical models with neural network components to improve pure methods in either category. Lastly, we outline some ways in which deep learning can also facilitate decision support with time-series data.

This article is part of the theme issue 'Machine learning for weather and climate modelling'.

1. Introduction

Time-series modelling has historically been a key area of academic research—forming an integral part of applications in topics such as climate modelling [1], biological sciences [2] and medicine [3], as well as commercial decision making in retail [4] and finance [5] to name a few. While traditional methods have focused on parametric models informed by domain expertise—such as autoregressive (AR) [6], exponential smoothing [7,8] or structural time-series models [9]—modern machine learning methods provide a means to learn temporal dynamics in a purely data-driven manner [10]. With the increasing data availability and computing power in recent times, machine learning has become a vital part of the next generation of time-series forecasting models.

Deep learning in particular has gained popularity in recent times, inspired by notable achievements in image classification [11], natural language processing [12] and

THE ROYAL SOCIETY PUBLISHING

© 2021 The Author(s) Published by the Royal Society. All rights reserved.

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

reinforcement learning [13]. By incorporating bespoke architectural assumptions—or inductive biases [14]—that reflect the nuances of underlying datasets, deep neural networks are able to learn complex data representations [15], which alleviates the need for manual feature engineering and model design. The availability of open-source backpropagation frameworks [16,17] has also simplified the network training, allowing for the customization for network components and loss functions.

Given the diversity of time-series problems across various domains, numerous neural network design choices have emerged. In this article, we summarize the common approaches to time-series prediction using deep neural networks. Firstly, we describe the state-of-the-art techniques available for common forecasting problems—such as multi-horizon forecasting and uncertainty estimation. Secondly, we analyse the emergence of a new trend in hybrid models, which combine both domain-specific quantitative models with deep learning components to improve forecasting performance. Next, we outline two key approaches in which neural networks can be used to facilitate decision support, specifically through methods in interpretability and counterfactual prediction. Finally, we conclude with some promising future research directions in deep learning for time series prediction—specifically in the form of continuous-time and hierarchical models.

While we endeavour to provide a comprehensive overview of modern methods in deep learning, we note that our survey is by no means all-encompassing. Indeed, a rich body of literature exists for automated approaches to time-series forecasting—including automatic parametric model selection [18], and traditional machine learning methods such as kernel regression [19] and support vector regression [20]. In addition, Gaussian processes [21] have been extensively used for time-series prediction—with recent extensions including deep Gaussian processes [22], and parallels in deep learning via neural processes [23]. Furthermore, older models of neural networks have been used historically in time-series applications, as seen in [24] and [25].

2. Deep learning architectures for time-series forecasting

Time-series forecasting models predict future values of a target $y_{i,t}$ for a given entity i at time t. Each entity represents a logical grouping of temporal information—such as measurements from different weather stations in climatology, or vital signs from different patients in medicine—and can be observed at the same time. In the simplest case, one-step-ahead forecasting models take the form

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \boldsymbol{x}_{i,t-k:t}, \boldsymbol{s}_i), \tag{2.1}$$

where $\hat{y}_{i,t+1}$ is the model forecast, $y_{i,t-k:t} = \{y_{i,t-k}, \ldots, y_{i,t}\}, \boldsymbol{x}_{i,t-k:t} = \{\boldsymbol{x}_{i,t-k}, \ldots, \boldsymbol{x}_{i,t}\}$ are observations of the target and exogenous inputs respectively over a look-back window k, s_i is static metadata associated with the entity (e.g. sensor location), and $f(.)$ is the prediction function learnt by the model. While we focus on univariate forecasting in this survey (i.e. 1-D targets), we note that the same components can be extended to multivariate models without loss of generality [26–30]. For notational simplicity, we omit the entity index i in subsequent sections unless explicitly required.

(a) Basic building blocks

Deep neural networks learn predictive relationships by using a series of nonlinear layers to construct intermediate feature representations [15]. In time-series settings, this can be viewed as encoding relevant historical information into a latent variable $\boldsymbol{z}_t$, with the final forecast produced using $\boldsymbol{z}_t$ alone

$$f(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{s}) = g_{\text{dec}}(\boldsymbol{z}_t) \tag{2.2}$$

and

$$\boldsymbol{z}_t = g_{\text{enc}}(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{s}), \tag{2.3}$$

where $g_{\text{enc}}(.), g_{\text{dec}}(.)$ are encoder and decoder functions respectively, and recalling that subscript i from equation (2.1) been removed to simplify notation (e.g. $y_{i,t}$ replaced by y_t). These encoders and decoders hence form the basic building blocks of deep learning architectures, with the choice of

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

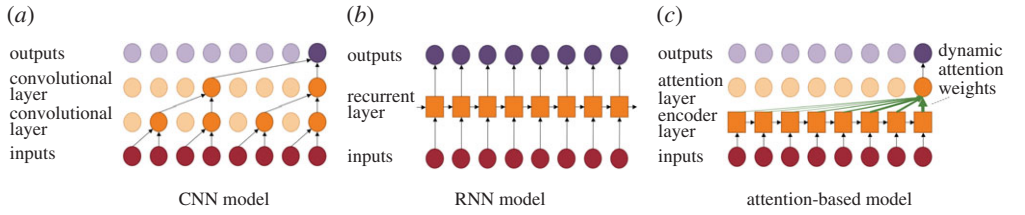

Figure 1. Incorporating temporal information using different encoder architectures. (*a*) CNN model, (*b*) RNN model and (*c*) attention-based model. (Online version in colour.)

network determining the types of relationships that can be learnt by our model. In this section, we examine modern design choices for encoders, as overviewed in figure 1, and their relationship to traditional temporal models. In addition, we explore common network outputs and loss functions used in time-series forecasting applications.

(i) Convolutional neural networks

Traditionally designed for image datasets, convolutional neural networks (CNNs) extract local relationships that are invariant across spatial dimensions [11,31]. To adapt CNNs to time-series datasets, researchers use multiple layers of causal convolutions [32–34]—i.e. convolutional filters designed to ensure only past information is used for forecasting. For an intermediate feature at hidden layer l, each causal convolutional filter takes the form below

$$\boldsymbol{h}_t^{l+1} = A\Big((\boldsymbol{W} * \boldsymbol{h})\,(l,t) \Big) \tag{2.4}$$

and

$$(\boldsymbol{W} * \boldsymbol{h})\,(l,t) = \sum_{\tau=0}^{k} \boldsymbol{W}(l,\tau)\boldsymbol{h}_{t-\tau}^{l}, \tag{2.5}$$

where $\boldsymbol{h}_t^l \in \mathbb{R}^{\mathcal{H}_{in}}$ is an intermediate state at layer l at time t, $*$ is the convolution operator, $\boldsymbol{W}(l,\tau) \in \mathbb{R}^{\mathcal{H}_{out}\times\mathcal{H}_{in}}$ is a fixed filter weight at layer l, and $A(.)$ is an activation function, such as a sigmoid function, representing any architecture-specific nonlinear processing. For CNNs that use a total of L convolutional layers, we note that the encoder output is then $\boldsymbol{z}_t = \boldsymbol{h}_t^L$.

Considering the 1-D case, we can see that equation (2.5) bears a strong resemblance to finite impulse response (FIR) filters in digital signal processing [35]. This leads to two key implications for temporal relationships learnt by CNNs. Firstly, in line with the spatial invariance assumptions for standard CNNs, temporal CNNs assume that relationships are time-invariant—using the same set of filter weights at each time step and across all time. In addition, CNNs are only able to use inputs within their defined lookback window, or receptive field, to make forecasts. As such, the receptive field size k needs to be tuned carefully to ensure that the model can make use of all relevant historical information. It is worth noting that a single causal CNN layer is equivalent to an AR model.

Dilated convolutions. Using standard convolutional layers can be computationally challenging where long-term dependencies are significant, as the number of parameters scales directly with the size of the receptive field. To alleviate this, modern architectures frequently make use of dilated covolutional layers [32,33], which extend equation (2.5) as below

$$(\boldsymbol{W} * \boldsymbol{h})\,(l,t,d_l) = \sum_{\tau=0}^{\lfloor k/d_l \rfloor} \boldsymbol{W}(l,\tau)\boldsymbol{h}_{t-d_l\tau}^{l}, \tag{2.6}$$

where $\lfloor . \rfloor$ is the floor operator and d_l is a layer-specific dilation rate. Dilated convolutions can hence be interpreted as convolutions of a down-sampled version of the lower layer features—reducing resolution to incorporate information from the distant past. As such, by increasing

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

the dilation rate with each layer, dilated convolutions can gradually aggregate information at different time blocks, allowing for more history to be used in an efficient manner. With the WaveNet architecture of [32] for instance, dilation rates are increased in powers of 2 with adjacent time blocks aggregated in each layer—allowing for 2^l time steps to be used at layer l as shown in figure 1*a*.

(ii) Recurrent neural networks

Recurrent neural networks (RNNs) have historically been used in sequence modelling [31], with strong results on a variety of natural language processing tasks [36]. Given the natural interpretation of time-series data as sequences of inputs and targets, many RNN-based architectures have been developed for temporal forecasting applications [37–40]. At their core, RNN cells contain an internal memory state which acts as a compact summary of past information. The memory state is recursively updated with new observations at each time step as shown in figure 1*b*, i.e.

$$\boldsymbol{z}_t = \nu\left(\boldsymbol{z}_{t-1}, y_t, \boldsymbol{x}_t, \boldsymbol{s}\right), \tag{2.7}$$

where $\boldsymbol{z}_t \in \mathbb{R}^{\mathcal{H}}$ here is the hidden internal state of the RNN, and $\nu(.)$ is the learnt memory update function. For instance, the Elman RNN [41], one of the simplest RNN variants, would take the form below

$$y_{t+1} = \gamma_y(\boldsymbol{W}_y \boldsymbol{z}_t + \boldsymbol{b}_y) \tag{2.8}$$

and

$$\boldsymbol{z}_t = \gamma_z(\boldsymbol{W}_{z_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{z_2} y_t + \boldsymbol{W}_{z_3}\boldsymbol{x}_t + \boldsymbol{W}_{z_4}\boldsymbol{s} + \boldsymbol{b}_z), \tag{2.9}$$

where $\boldsymbol{W}_\cdot, \boldsymbol{b}_\cdot$ are the linear weights and biases of the network respectively, and $\gamma_y(.), \gamma_z(.)$ are network activation functions. Note that RNNs do not require the explicit specification of a lookback window as per the CNN case. From a signal processing perspective, the main recurrent layer—i.e. equation (2.9)—thus resembles a nonlinear version of infinite impulse response (IIR) filters.

Long short-term memory Due to the infinite lookback window, older variants of RNNs can suffer from limitations in learning long-range dependencies in the data [42,43]—due to issues with exploding and vanishing gradients [31]. Intuitively, this can be seen as a form of resonance in the memory state. Long short-term memory networks (LSTMs) [44] were hence developed to address these limitations, by improving gradient flow within the network. This is achieved through the use of a cell state $\boldsymbol{c}_t$ which stores long-term information, modulated through a series of gates as below

$$\textit{input gate}: \boldsymbol{i}_t = \sigma(\boldsymbol{W}_{i_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{i_2} y_t + \boldsymbol{W}_{i_3}\boldsymbol{x}_t + \boldsymbol{W}_{i_4}\boldsymbol{s} + \boldsymbol{b}_i), \tag{2.10}$$

$$\textit{output gate}: \boldsymbol{o}_t = \sigma(\boldsymbol{W}_{o_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{o_2} y_t + \boldsymbol{W}_{o_3}\boldsymbol{x}_t + \boldsymbol{W}_{o_4}\boldsymbol{s} + \boldsymbol{b}_o), \tag{2.11}$$

$$\textit{forget gate}: \boldsymbol{f}_t = \sigma(\boldsymbol{W}_{f_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{f_2} y_t + \boldsymbol{W}_{f_3}\boldsymbol{x}_t + \boldsymbol{W}_{f_4}\boldsymbol{s} + \boldsymbol{b}_f), \tag{2.12}$$

where $\boldsymbol{z}_{t-1}$ is the hidden state of the LSTM, and $\sigma(.)$ is the sigmoid activation function. The gates modify the hidden and cell states of the LSTM as below

$$\textit{hidden state}: \boldsymbol{z}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{2.13}$$

$$\textit{cell state}: \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_{c_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{c_2} y_t + \boldsymbol{W}_{c_3}\boldsymbol{x}_t + \boldsymbol{W}_{c_4}\boldsymbol{s} + \boldsymbol{b}_c), \tag{2.14}$$

where $\odot$ is the element-wise (Hadamard) product, and tanh(.) is the tanh activation function.

Relationship to Bayesian filtering As examined in [39], Bayesian filters [45] and RNNs are both similar in their maintenance of a hidden state which is recursively updated over time. For Bayesian filters, such as the Kalman filter [46], inference is performed by updating the sufficient statistics of the latent state—using a series of state transition and error correction steps. As the Bayesian filtering steps use deterministic equations to modify sufficient statistics, the RNN can be viewed as a simultaneous approximation of both steps—with the memory vector containing all relevant information required for prediction.

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

(iii) Attention mechanisms

The development of attention mechanisms [47,48] has also led to improvements in long-term dependency learning—with transformer architectures achieving state-of-the-art performance in multiple natural language processing applications [12,49,50]. Attention layers aggregate temporal features using dynamically generated weights (figure 1*c*), allowing the network to directly focus on significant time steps in the past—even if they are very far back in the lookback window. Conceptually, attention is a mechanism for a key-value lookup based on a given query [51], taking the form below

$$\boldsymbol{h}_t = \sum_{\tau=0}^{k} \alpha(\boldsymbol{\kappa}_t, \boldsymbol{q}_\tau)\boldsymbol{v}_{t-\tau}, \tag{2.15}$$

where the key $\boldsymbol{\kappa}_t$, query $\boldsymbol{q}_\tau$ and value $\boldsymbol{v}_{t-\tau}$ are intermediate features produced at different time steps by lower levels of the network. Furthermore, $\alpha(\boldsymbol{\kappa}_t, \boldsymbol{q}_\tau) \in [0, 1]$ is the attention weight for $t - \tau$ generated at time t, and $\boldsymbol{h}_t$ is the context vector output of the attention layer. Note that multiple attention layers can also be used together as per the CNN case, with the output from the final layer forming the encoded latent variable $\boldsymbol{z}_t$.

Recent work has also demonstrated the benefits of using attention mechanisms in time-series forecasting applications, with improved performance over comparable recurrent networks [52–54]. For instance, [52] use attention to aggregate features extracted by RNN encoders, with attention weights produced as below

$$\boldsymbol{\alpha}(t) = \text{softmax}(\boldsymbol{\eta}_t) \tag{2.16}$$

and

$$\boldsymbol{\eta}_t = \mathbf{W}_{\eta_1} \tanh(\mathbf{W}_{\eta_2}\boldsymbol{\kappa}_{t-1} + \mathbf{W}_{\eta_3}\boldsymbol{q}_\tau + \boldsymbol{b}_\eta), \tag{2.17}$$

where $\boldsymbol{\alpha}(t) = [\alpha(t, 0), \ldots \alpha(t, k)]$ is a vector of attention weights, $\boldsymbol{\kappa}_{t-1}, \boldsymbol{q}_t$ are outputs from LSTM encoders used for feature extraction, and softmax(.) is the softmax activation function. More recently, transformer architectures have also been considered in [53,54], which apply scalar-dot product self-attention [49] to features extracted within the lookback window. From a time-series modelling perspective, attention provides two key benefits. Firstly, networks with attention are able to directly attend to any significant events that occur. In retail forecasting applications, for example, this includes holiday or promotional periods which can have a positive effect on sales. Secondly, as shown in [54], attention-based networks can also learn regime-specific temporal dynamics—by using distinct attention weight patterns for each regime.

(iv) Outputs and loss functions

Given the flexibility of neural networks, deep neural networks have been used to model both discrete [55] and continuous [37,56] targets—by customizing of decoder and output layer of the neural network to match the desired target type. In one-step-ahead prediction problems, this can be as simple as combining a linear transformation of encoder outputs (i.e. equation (2.2)) together with an appropriate output activation for the target. Regardless of the form of the target, predictions can be further divided into two different categories—point estimates and probabilistic forecasts.

Point estimates A common approach to forecasting is to determine the expected value of a future target. This essentially involves reformulating the problem to a classification task for discrete outputs (e.g. forecasting future events), and regression task for continuous outputs—using the encoders described above. For the binary classification case, the final layer of the decoder then features a linear layer with a sigmoid activation function—allowing the network to predict the probability of event occurrence at a given time step. For one-step-ahead forecasts of binary and

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

continuous targets, networks are trained using binary cross-entropy and mean square error loss functions, respectively

$$\mathcal{L}_{\text{classification}} = -\frac{1}{T}\sum_{t=1}^{T} y_t \log(\hat{y}_t) + (1 - y_t)\log(1 - \hat{y}_t) \tag{2.18}$$

and

$$\mathcal{L}_{\text{regression}} = \frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2 . \tag{2.19}$$

While the loss functions above are the most common across applications, we note that the flexibility of neural networks also allows for more complex losses to be adopted—e.g. losses for quantile regression [56] and multinomial classification [32].

Probabilistic outputs While point estimates are crucial to predicting the future value of a target, understanding the uncertainty of a model's forecast can be useful for decision makers in different domains. When forecast uncertainties are wide, for instance, model users can exercise more caution when incorporating predictions into their decision making, or alternatively rely on other sources of information. In some applications, such as financial risk management, having access to the full predictive distribution will allow decision makers to optimize their actions in the presence of rare events—e.g. allowing risk managers to insulate portfolios against market crashes.

A common way to model uncertainties is to use deep neural networks to generate parameters of known distributions [27,37,38]. For example, Gaussian distributions are typically used for forecasting problems with continuous targets, with the networks outputting means and variance parameters for the predictive distributions at each step as below

$$y_{t+\tau} \sim N(\mu(t,\tau), \zeta(t,\tau)^2), \tag{2.20}$$

$$\mu(t,\tau) = \boldsymbol{W}_\mu \boldsymbol{h}_t^L + \boldsymbol{b}_\mu, \tag{2.21}$$

$$\zeta(t,\tau) = \text{softplus}(\boldsymbol{W}_\Sigma \boldsymbol{h}_t^L + \boldsymbol{b}_\Sigma), \tag{2.22}$$

where $\boldsymbol{h}_t^L$ is the final layer of the network, and softplus(.) is the softplus activation function to ensure that standard deviations take only positive values.

(b) Multi-horizon forecasting models

In many applications, it is often beneficial to have access to predictive estimates at multiple points in the future—allowing decision makers to visualize trends over a future horizon, and optimize their actions across the entire path. From a statistical perspective, multi-horizon forecasting can be viewed as a slight modification of one-step-ahead prediction problem (i.e. equation (2.1)) as below

$$\hat{y}_{t+\tau} = f(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{u}_{t-k:t+\tau}, \boldsymbol{s}, \tau), \tag{2.23}$$

where $\tau \in \{1, \ldots, \tau_{\max}\}$ is a discrete forecast horizon, $\boldsymbol{u}_t$ are known future inputs (e.g. date information, such as the day-of-week or month) across the entire horizon, and $\boldsymbol{x}_t$ are inputs that can only be observed historically. In line with traditional econometric approaches [57,58], deep learning architectures for multi-horizon forecasting can be divided into iterative and direct methods—as shown in figure 2 and described in detail below.

(i) Iterative methods

Iterative approaches to multi-horizon forecasting typically make use of AR deep learning architectures [37,39,40,53]—producing multi-horizon forecasts by recursively feeding samples of the target into future time steps (figure 2*a*). By repeating the procedure to generate multiple trajectories, forecasts are then produced using the sampling distributions for target values at each step. For instance, predictive means can be obtained using the Monte Carlo estimate

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

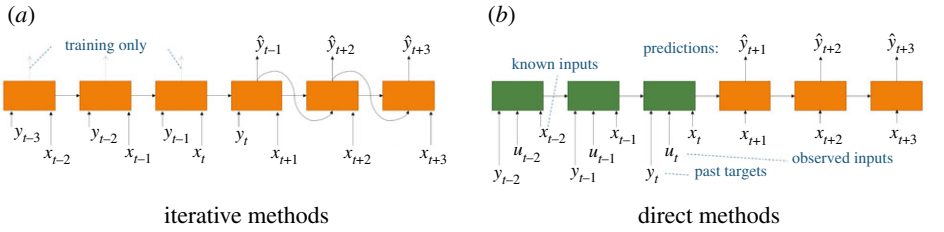

Figure 2. Main types of multi-horizon forecasting models. (*a*) iterative methods and (*b*) direct methods. (Online version in colour.)

$\hat{y}_{t+\tau} = \sum_{j=1}^{J} \tilde{y}_{t+\tau}^{(j)}/J$, where $\tilde{y}_{t+\tau}^{(j)}$ is a sample taken based on the model of equation (2.20). As AR models are trained in the exact same fashion as one-step-ahead prediction models (i.e. via backpropagation through time), the iterative approach allows for the easy generalization of standard models to multi-step forecasting. However, as a small amount of error is produced at each time step, the recursive structure of iterative methods can potentially lead to large error accumulations over longer forecasting horizons. In addition, iterative methods assume that all inputs but the target are known at run-time—requiring only samples of the target to be fed into future time steps. This can be a limitation in many practical scenarios where observed inputs exist, motivating the need for more flexible methods.

(ii) Direct methods

Direct methods alleviate the issues with iterative methods by producing forecasts directly using all available inputs. They typically make use of sequence-to-sequence architectures [52,54,56], using an encoder to summarize past information (i.e. targets, observed inputs and *a priori* known inputs), and a decoder to combine them with known future inputs—as depicted in figure 2*b*. As described in [59], an alternative approach is to use simpler models to directly produce a fixed-length vector matching the desired forecast horizon. This, however, does require the specification of a maximum forecast horizon (i.e. $\tau_{\max}$), with predictions made only at the predefined discrete intervals.

3. Incorporating domain knowledge with hybrid models

Despite its popularity, the efficacy of machine learning for time-series prediction has historically been questioned—as evidenced by forecasting competitions such as the M-competitions [60]. Prior to the M4 competition of 2018 [61], the prevailing wisdom was that sophisticated methods do not produce more accurate forecasts, and simple models with ensembling had a tendency to do better [59,62,63]. Two key reasons have been identified to explain the underperformance of machine methods. Firstly, the flexibility of machine learning methods can be a double-edged sword—making them prone to overfitting [59]. Hence, simpler models may potentially do better in low data regimes, which are particularly common in forecasting problems with a small number of historical observations (e.g. quarterly macroeconomic forecasts). Secondly, similar to stationarity requirements of statistical models, machine learning models can be sensitive to how inputs are pre-processed [26,37,59], which ensure that data distributions at training and test time are similar.

A recent trend in deep learning has been in developing hybrid models which address these limitations, demonstrating improved performance over pure statistical or machine learning models in a variety of applications [38,64–66]. Hybrid methods combine well-studied quantitative time-series models together with deep learning—using deep neural networks to generate model parameters at each time step. On the one hand, hybrid models allow domain experts to inform neural network training using prior information—reducing the hypothesis space of the network

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

and improving generalization. This is especially useful for small datasets [38], where there is a greater risk of overfitting for deep learning models. Furthermore, hybrid models allow for the separation of stationary and non-stationary components, and avoid the need for custom input pre-processing. An example of this is the exponential smoothing RNN (ES-RNN) [64], winner of the M4 competition, which uses exponential smoothing to capture non-stationary trends and learns additional effects with the RNN. In general, hybrid models use deep neural networks in two manners: a) to encode time-varying parameters for non-probabilistic parametric models [64,65,67], and b) to produce distribution parameters for probabilistic models [38,40,66].

(a) Non-probabilistic hybrid models

With parametric time-series models, forecasting equations are typically defined analytically and provide point forecasts for future targets. Non-probabilistic hybrid models hence modify these forecasting equations to combine statistical and deep learning components. The ES-RNN, for example, uses the update equations of the Holt–Winters exponential smoothing model [8]—combining multiplicative level and seasonality components with deep learning outputs as below

$$\hat{y}_{i,t+\tau} = \exp(\boldsymbol{W}_{ES}\boldsymbol{h}^{L}_{i,t+\tau} + \boldsymbol{b}_{ES}) \times l_{i,t} \times \gamma_{i,t+\tau}, \tag{3.1}$$

$$l_{i,t} = \frac{\beta_1^{(i)} y_{i,t}}{\gamma_{i,t}} + (1-\beta_1^{(i)}) l_{i,t-1} \tag{3.2}$$

and

$$\gamma_{i,t} = \frac{\beta_2^{(i)} y_{i,t}}{l_{i,t}} + (1-\beta_2^{(i)}) \gamma_{i,t-\kappa}, \tag{3.3}$$

where $\boldsymbol{h}^{L}_{i,t+\tau}$ is the final layer of the network for the τth-step-ahead forecast, $l_{i,t}$ is a level component, $\gamma_{i,t}$ is a seasonality component with period κ, and $\beta_1^{(i)}, \beta_2^{(i)}$ are entity-specific static coefficients. From the above equations, we can see that the exponential smoothing components $(l_{i,t}, \gamma_{i,t})$ handle the broader (e.g. exponential) trends within the datasets, reducing the need for additional input scaling.

(b) Probabilistic hybrid models

Probabilistic hybrid models can also be used in applications where distribution modelling is important—using probabilistic generative models for temporal dynamics such as Gaussian processes [40] and linear state space models [38]. Rather than modifying forecasting equations, probabilistic hybrid models use neural networks to produce parameters for predictive distributions at each step. For instance, deep state space models [38] encode time-varying parameters for linear state space models as below—performing inference via the Kalman filtering equations [46]

$$y_t = \boldsymbol{a}(\boldsymbol{h}^{L}_{i,t+\tau})^T \boldsymbol{l}_t + \phi(\boldsymbol{h}^{L}_{i,t+\tau})\epsilon_t \tag{3.4}$$

and

$$\boldsymbol{l}_t = \boldsymbol{F}(\boldsymbol{h}^{L}_{i,t+\tau})\boldsymbol{l}_{t-1} + \boldsymbol{q}(\boldsymbol{h}^{L}_{i,t+\tau}) + \boldsymbol{\Sigma}(\boldsymbol{h}^{L}_{i,t+\tau}), \tag{3.5}$$

where $\boldsymbol{l}_t$ is the hidden latent state, $\boldsymbol{a}(.)$, $\boldsymbol{F}(.)$, $\boldsymbol{q}(.)$ are linear transformations of $\boldsymbol{h}^{L}_{i,t+\tau}$, $\phi(.)$, $\boldsymbol{\Sigma}(.)$ are linear transformations with softmax activations, and $\epsilon_t \sim N(0,1)$, $\boldsymbol{\Sigma}_t \sim N(0, \mathbb{I})$ are standard normal random variables.

4. Facilitating decision support using deep neural networks

Although model builders are mainly concerned with the accuracy of their forecasts, end-users typically use predictions to *guide their future actions*. For instance, doctors can make use of

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

clinical forecasts (e.g. probabilities of disease onset and mortality) to help them prioritize tests to order, formulate a diagnosis and determine a course of treatment. As such, while time-series forecasting is a crucial preliminary step, a better understanding of both temporal dynamics and the motivations behind a model's forecast can help users further optimize their actions. In this section, we explore two directions in which neural networks have been extended to facilitate decision support with time-series data—focusing on methods in interpretability and causal inference.

(a) Interpretability with time-series data

With the deployment of neural networks in mission-critical applications [68], there is a increasing need to understand both *how* and *why* a model makes a certain prediction. Moreover, end-users can have little prior knowledge with regards to the relationships present in their data, with datasets growing in size and complexity in recent times. Given the black-box nature of standard neural network architectures, a new body of research has emerged in methods for interpreting deep learning models, which we present in depth below.

Techniques for post hoc interpretability post hoc interpretable models are developed to interpret trained networks, and helping to identify important features or examples without modifying the original weights. Methods can mainly be divided into two main categories. Firstly, one possible approach is to apply simpler interpretable surrogate models between the inputs and outputs of the neural network, and rely on the approximate model to provide explanations. For instance, local interpretable model-agnostic explanations (LIME) [69] identify relevant features by fitting instance-specific linear models to perturbations of the input, with the linear coefficients providing a measure of importance. Shapley additive explanations (SHAP) [70] provide another surrogate approach, which uses Shapley values from cooperative game theory to identify important features across the dataset. Next, gradient-based methods—such as saliency maps [71,72] and influence functions [73]—have been proposed, which analyse network gradients to determine which input features have the greatest impact on loss functions. While *post hoc* interpretability methods can help with feature attributions, they typically ignore any sequential dependencies between inputs—making it difficult to apply them to complex time-series datasets.

Inherent interpretability with attention weights An alternative approach is to directly design architectures with explainable components, typically in the form of strategically placed attention layers. As attention weights are produced as outputs from a softmax layer, the weights are constrained to sum to 1, i.e. $\sum_{\tau=0}^{k} \alpha(t,\tau) = 1$. For time-series models, the outputs of equation (2.15) can hence also be interpreted as a weighted average over temporal features, using the weights supplied by the attention layer at each step. An analysis of attention weights can then be used to understand the relative importance of features at each time step. Instance-wise interpretability studies have been performed in [53,55,74], where the authors used specific examples to show how the magnitudes of $\alpha(t,\tau)$ can indicate which time points were most significant for predictions. By analysing distributions of attention vectors across time, [54] also shows how attention mechanisms can be used to identify persistent temporal relationships—such as seasonal patterns—in the dataset.

(b) Counterfactual predictions and causal inference over time

In addition to understanding the relationships learnt by the networks, deep learning can also help to facilitate decision support by producing predictions outside of their observational datasets, or counterfactual forecasts. Counterfactual predictions are particularly useful for scenario analysis applications—allowing users to evaluate how different sets of actions can impact target trajectories. This can be useful both from a historical angle, i.e. determining what would have happened if a different set of circumstances had occurred, and from a forecasting perspective, i.e. determining which actions to take to optimize future outcomes.

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

While a large class of deep learning methods exists for estimating causal effects in static settings [75–77], the key challenge in time series datasets is the presence of time-dependent confounding effects. This arises due to circular dependencies when actions that can affect the target are also conditional on observations of the target. Without any adjusting for time-dependent confounders, straightforward estimation techniques can results in biased results, as shown in [78]. Recently, several methods have emerged to train deep neural networks while adjusting for time-dependent confounding, based on extensions of statistical techniques and the design of new loss functions. With statistical methods, [79] extend the inverse-probability-of-treatment-weighting (IPTW) approach of marginal structural models in epidemiology—using one set of networks to estimate treatment application probabilities, and a sequence-to-sequence model to learn unbiased predictions. Another approach in [80] extends the G-computation framework, jointly modelling distributions of the target and actions using deep learning. In addition, new loss functions have been proposed in [81], which adopts domain adversarial training to learn balanced representations of patient history.

5. Conclusion and future directions

With the growth in data availability and computing power in recent times, deep neural networks architectures have achieved much success in forecasting problems across multiple domains. In this article, we survey the main architectures used for time-series forecasting—highlighting the key building blocks used in neural network design. We examine how they incorporate temporal information for one-step-ahead predictions, and describe how they can be extended for use in multi-horizon forecasting. Furthermore, we outline the recent trend of hybrid deep learning models, which combine statistical and deep learning components to outperform pure methods in either category. Finally, we summarize two ways in which deep learning can be extended to improve decision support over time, focusing on methods in interpretability and counterfactual prediction.

Although a large number of deep learning models have been developed for time-series forecasting, some limitations still exist. Firstly, deep neural networks typically require time series to be discretized at regular intervals, making it difficult to forecast datasets where observations can be missing or arrive at random intervals. While some preliminary research on continuous-time models has been done via neural ordinary differential equations [82], additional work needs to be done to extend this work for datasets with complex inputs (e.g. static variables) and to benchmark them against existing models. In addition, as mentioned in [83], time series often have a hierarchical structure with logical groupings between trajectories—e.g. in retail forecasting, where product sales in the same geography can be affected by common trends. As such, the development of architectures which explicitly account for such hierarchies could be an interesting research direction, and potentially improve forecasting performance over existing univariate or multivariate models.

Data accessibility. This article has no additional data.

Authors' contributions. B.L. wrote the draft of the manuscript and created the figures. S.Z. supervised the project, providing guidance on content and revising the manuscript.

Competing interests. We declare we have no competing interests.

Funding. Financial support from the Oxford-Man Institute of Quantitative Finance is kindly acknowledged.

References

1. Mudelsee M. 2019 Trend analysis of climate time series: a review of methods. *Earth Sci. Rev.* **190**, 310–322. (doi:10.1016/j.earscirev.2018.12.005)
2. Stoffer DS, Ombao H. 2012 Editorial: special issue on time series analysis in the biological sciences. *J. Time Ser. Anal.* **33**, 701–703. (doi:10.1111/j.1467-9892.2012.00805.x)
3. Topol EJ. 2019 High-performance medicine: the convergence of human and artificial intelligence. *Nat. Med.* **25**, 44–56. (doi:10.1038/s41591-018-0300-7)

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

4. Böse JH, Flunkert V, Gasthaus J, Januschowski T, Lange D, Salinas D, Schelter S, Seeger M, Wang Y. 2017 Probabilistic demand forecasting at scale. *Proc. VLDB Endow.* **10**, 1694–1705. (doi:10.14778/3137765.3137775)
5. Andersen TG, Bollerslev T, Christoffersen PF, Diebold FX. 2005 Volatility forecasting. *Natl Bur. Econ. Res.* 11188. (doi:10.3386/w11188)
6. Box GEP, Jenkins GM. 1976 *Time series analysis: forecasting and control*. San Francisco, CA: Holden-Day.
7. Gardner Jr ES. 1985 Exponential smoothing: the state of the art. *J. Forecast.* **4**, 1–28. (doi:10.1002/for.3980040103)
8. Winters PR. 1960 Forecasting sales by exponentially weighted moving averages. *Manage. Sci.* **6**, 324–342. (doi:10.1287/mnsc.6.3.324)
9. Harvey AC. 1990 *Forecasting, structural time series models and the kalman filter*. Cambridge, UK: Cambridge University Press.
10. Ahmed NK, Atiya AF, Gayar NE, El-Shishiny H. 2010 An empirical comparison of machine learning models for time series forecasting. *Econ. Rev.* **29**, 594–621. (doi:10.1080/07474938.2010.481556)
11. Krizhevsky A, Sutskever I, Hinton GE. 2012 ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS)* (eds F Pereira, CJC Burges, L Bottou, KQ Weinberger), pp. 1097–1105.
12. Devlin J, Chang MW, Lee K, Toutanova K. 2019 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1 (Long and Short Papers), pp. 4171–4186.
13. Silver D *et al.* 2016 Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–503. (doi:10.1038/nature16961)
14. Baxter J. 2000 A model of inductive bias learning. *J. Artif. Int. Res.* **12**, 149–198. (doi:10.1613/jair.731)
15. Bengio Y, Courville A, Vincent P. 2013 Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1798–1828. (doi:10.1109/TPAMI.2013.50)
16. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S. 2015 TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. Available from: http://tensorflow.org/.
17. Paszke A *et al.* 2019 PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32, Vancouver, Canada, 8–14 December 2019*, pp. 8024–8035.
18. Hyndman RJ, Khandakar Y. 2008 Automatic time series forecasting: the forecast package for R. *J. Stat. Softw.* **26**, 1–22. (doi:10.18637/jss.v027.i03)
19. Nadaraya EA. 1964 On estimating regression. *Theory Probab. Appl.* **9**, 141–142. (doi:10.1137/1109020)
20. Smola AJ, Schülkopf B. 2004 A tutorial on support vector regression. *Stat. Comput.* **14**, 199–222. (doi:10.1023/B:STCO.0000035301.49549.88)
21. Williams CKI, Rasmussen CE. 1996 Gaussian processes for regression. In *Advances in Neural Information Processing Systems (NIPS), Denver, CO, 27–30 November 1995*.
22. Damianou A, Lawrence N. 2013 Deep Gaussian Processes. In *Proc. of the Conf. on Artificial Intelligence and Statistics (AISTATS), Scottsdale, AZ, 29 April–1 May 2013*.
23. Garnelo M, Rosenbaum D, Maddison CJ, Ramalho T, Saxton D, Shanahan M, Teh YW, Rezende DJ, Eslami SM. 2018 Conditional neural processes. In *Proc. of the Int. Conf. on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018*.
24. Waibel A. 1989 Modular construction of time-delay neural networks for speech recognition. *Neural Comput.* **1**, 39–46. (doi:10.1162/neco.1989.1.1.39)
25. Wan E. 1994 Time series prediction by using a connectionist network with internal delay lines. In *Time Series Prediction*, pp. 195–217. Boston, MA: Addison-Wesley.
26. Sen R, Yu HF, Dhillon I. 2019 Think globally, act locally: a deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 8–14 December 2019*.

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

27. Wen R, Torkkola K. 2019 Deep generative quantile-copula models for probabilistic forecasting. In *ICML Time Series Workshop, Long Beach, CA, 9–15 June 2019*.
28. Li Y, Yu R, Shahabi C, Liu Y. 2018 Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In *Proc. of the Int. Conf. on Learning Representations ICLR, Vancouver, Canada, 30 April–3 May 2018*.
29. Ghaderi A, Sanandaji BM, Ghaderi F. 2017 Deep forecast: deep learning-based spatio-temporal forecasting. In *ICML Time Series Workshop, Sydney, Australia, 6–11 August 2017*.
30. Salinas D, Bohlke-Schneider M, Callot L, Medico R, Gasthaus J. 2019 High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 8–14 December 2019*.
31. Goodfellow I, Bengio Y, Courville A. 2016 *Deep learning*. Cambridge, MA: MIT Press.
32. van den Oord AV, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K. 2016 WaveNet: a generative model for raw audio. (http://arxiv.org/abs/1609.03499)
33. Bai S, Zico Kolter J, Koltun V. 2018 An Empirical evaluation of generic convolutional and recurrent networks for sequence modeling. (http://arxiv.org/abs/1803.01271)
34. Borovykh A, Bohte S, Oosterlee CW. 2017 Conditional time series forecasting with convolutional neural networks. (http://arxiv.org/abs/1703.04691)
35. Lyons RG. 2004 *Understanding digital signal processing*, 2nd edn. USA: Prentice Hall PTR.
36. Young T, Hazarika D, Poria S, Cambria E. 2018 Recent trends in deep learning based natural language processing [Review Article]. *IEEE Comput. Intell. Mag.* **13**, 55–75. (doi:10.1109/MCI.2018.2840738)
37. Salinas D, Flunkert V, Gasthaus J. 2017 DeepAR: probabilistic forecasting with autoregressive recurrent networks. (http://arxiv.org/abs/1704.04110)
38. Rangapuram SS, Seeger MW, Gasthaus J, Stella L, Wang Y, Januschowski T. 2018 Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems (NIPS), Montreal, Canada, 2–8 December 2018*.
39. Lim B, Zohren S, Roberts S. 2020 Recurrent neural filters: learning independent Bayesian filtering steps for time series prediction. In *Int. Joint Conf. on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020*.
40. Wang Y, Smola A, Maddix D, Gasthaus J, Foster D, Januschowski T. 2019 Deep factors for forecasting. In *Proc. of the Int. Conf. on Machine Learning (ICML), Long Beach, CA, 9–15 June 2019*.
41. Elman JL. 1990 Finding structure in time. *Cogn. Sci.* **14**, 179–211. (doi:10.1207/s15516709cog1402_1)
42. Bengio Y, Simard P, Frasconi P. 1994 Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **5**, 157–166. (doi:10.1109/72.279181)
43. Kolen JF, Kremer SC. 2001 In *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pp. 237–243.
44. Hochreiter S, Schmidhuber J. 1997 Long short-term memory. *Neural Comput.* **9**, 1735–1780. (doi:10.1162/neco.1997.9.8.1735)
45. Srkk S. 2013 *Bayesian filtering and smoothing*. Cambridge, UK: Cambridge University Press.
46. Kalman RE. 1960 A new approach to linear filtering and prediction problems. *Basic Eng.* **82**, 35. (doi:10.1115/1.3662552)
47. Bahdanau D, Cho K, Bengio Y. 2015 Neural machine translation by jointly learning to align and translate. In *Proc. of the Int. Conf. on Learning Representations (ICLR), San Diego, CA, 7–9 May 2015*.
48. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. 2014 Learning phrase representations using RNN Encoder–Decoder for statistical machine translation. In *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP), Dohar, Qatar, 25–29 October 2014*.
49. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. 2017 Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, 4–9 December 2017*.
50. Dai Z, Yang Z, Yang Y, Carbonell J, Le Q, Salakhutdinov R. 2019 Transformer-XL: attentive language models beyond a fixed-length context. In *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, 28 July–2 August 2019*.

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025

51. Graves A, Wayne G, Danihelka I. 2014 Neural turing machines. CoRR. abs/1410.5401.
52. Fan C, Zhang Y, Pan Y, Li X, Zhang C, Yuan R, Wu D, Wang W, Pei J, Huang H. 2019 Multi-horizon time series forecasting with temporal attention learning. In *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*.
53. Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang YX, Yan X. 2019 Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 8–14 December 2019*.
54. Lim B, Arik SO, Loeff N, Pfister T. 2019 Temporal fusion transformers for interpretable multi-horizon time series forecasting. (http://arxiv.org/abs/1912.09363)
55. Choi E, Bahadori MT, Sun J, Kulas JA, Schuetz A, Stewart WF. 2016 RETAIN: an interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016*.
56. Wen R, Torkkola K, Narayanaswamy B, Madeka D. 2017 A multi-horizon quantile recurrent forecaster. In *NIPS 2017 Time Series Workshop*.
57. Taieb SB, Sorjamaa A, Bontempi G. 2010 Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing* **73**, 1950–1957. (doi:10.1016/j.neucom.2009.11.030)
58. Marcellino M, Stock J, Watson M. 2006 A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series. *J. Econom.* **135**, 499–526. (doi:10.1016/j.jeconom.2005.07.020)
59. Makridakis S, Spiliotis E, Assimakopoulos V. 2018 Statistical and machine learning forecasting methods: concerns and ways forward. *PLoS ONE* **13**, 1–26. (doi:10.1371/journal.pone.0194889)
60. Hyndman R. 2020 A brief history of forecasting competitions. *Int. J. Forecast.* **36**, 7–14. (doi:10.1016/j.ijforecast.2019.03.015)
61. 2020 The M4 Competition: 100 000 time series and 61 forecasting methods. *Int. J. Forecast.* **36**, 54–74. (doi:10.1016/j.ijforecast.2019.04.014)
62. Fildes R, Hibon M, Makridakis S, Meade N. 1998 Generalising about univariate forecasting methods: further empirical evidence. *Int. J. Forecast.* **14**, 339–358. (doi:10.1016/S0169-2070(98)00009-0)
63. Makridakis S, Hibon M. 2000 The M3-competition: results, conclusions and implications. *Int. J. Forecast.* **16**, 451–476. The M3- Competition. (doi:10.1016/S0169-2070(00)00057-1)
64. Smyl S. 2020 A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *Int. J. Forecast.* **36**, 75–85. M4 Competition. (doi:10.1016/j.ijforecast.2019.03.017)
65. Lim B, Zohren S, Roberts S. 2019 Enhancing time-series momentum strategies using deep neural networks. *J. Financ. Data Sci.* **2**, Fall 2020. (doi:10.2139/ssrn.3369195)
66. Grover A, Kapoor A, Horvitz E. 2015 A deep hybrid model for weather forecasting. In *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), Sydney, Australia, 10–13 August 2015*.
67. Binkowski M, Marti G, Donnat P. 2018 Autoregressive convolutional neural networks for asynchronous time series. In *Proc. of the Int. Conf. on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018*.
68. Moraffah R, Karami M, Guo R, Raglin A, Liu H. 2020 Causal Interpretability for Machine Learning—Problems, Methods and Evaluation. (http://arxiv.org/abs/2003.03934)
69. Ribeio M, Singh S, Guestrin C. 2016 'Why Should I Trust You?' Explaining the Predictions of Any Classifier. In KDD.
70. Lundberg S, Lee SI. 2017 A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NIPS)*.
71. Simonyan K, Vedaldi A, Zisserman A. 2013 Deep inside convolutional networks: visualising image classification models and saliency maps. (http://arxiv.org/abs/1312.6034)
72. Siddiqui SA, Mercier D, Munir M, Dengel A, Ahmed S. 2019 TSViz: demystification of deep learning models for time-series analysis. *IEEE Access* **7**, 67027–67040. (doi:10.1109/ACCESS.2019.2912823)
73. Koh PW, Liang P. 2017 Understanding black-box predictions via influence functions. In Proc. of the Int. Conf. on Machine Learning (ICML).

74. Bai T, Zhang S, Egleston BL, Vucetic S. 2018 Interpretable representation learning for healthcare via capturing disease progression through time. In *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (KDD), London, UK, 19–23 August, 2018*.
75. Yoon J, Jordon J, van der Schaar M. 2018 GANITE: estimation of individualized treatment effects using generative adversarial nets. In *Int. Conf. on Learning Representations (ICLR), Sydney, Australia, 6–11 August 2017*.
76. Hartford J, Lewis G, Leyton-Brown K, Taddy M. 2017 Deep IV: a flexible approach for counterfactual prediction. In *Proc. of the 34th Int. Conf. on Machine Learning (ICML), London, UK, 19–23 August 2018*.
77. Alaa AM, Weisz M, van der Schaar M. 2017 Deep counterfactual networks with propensity dropout. In *Proc. of the 34th Int. Conf. on Machine Learning (ICML), Vancouver, Canada, 30 April–3 May 2018*.
78. Mansournia MA, Etminan M, Danaei G, Kaufman JS, Collins G. 2017 Handling time varying confounding in observational research. *BMJ* **16**, 359. (doi:10.1136/bmj.j4587)
79. Lim B, Alaa A, van der Schaar M. 2018 Forecasting treatment responses over time using recurrent marginal structural networks. In *NeurIPS*.
80. Li R, Shahn Z, Li J, Lu M, Chakraborty P, Sow D, Ghalwash M, Lehman LW. 2020 G-Net: a deep learning approach to G-computation for counterfactual outcome prediction under dynamic treatment regimes. (http://arxiv.org/abs/2003.10551)
81. Bica I, Alaa AM, Jordon J, van der Schaar M. 2020 Estimating counterfactual treatment outcomes over time through adversarially balanced representations. In *Int. Conf. on Learning Representations(ICLR)*.
82. Chen RTQ, Rubanova Y, Bettencourt J, Duvenaud D. 2018 Neural ordinary differential equations. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NIPS)*.
83. Fry C, Brundage M. 2019 The M4 forecasting competition–a practitioner's view. *Int. J. Forecast.* **36**, 156–160. (doi:10.1016/j.ijforecast.2019.02.013)

Downloaded from https://royalsocietypublishing.org/ on 17 February 2025