E-RNN: Entangled Recurrent Neural Networks for Causal Prediction

Jinsung Yoon [1] **Mihaela van der Schaar** [1 2 3]

Abstract

We propose a novel architecture of recurrent neural networks (RNNs) for causal prediction which we call Entangled RNN (E-RNN). To issue causal predictions, E-RNN can propagate the backward hidden states of Bi-RNN through an additional forward hidden layer. Unlike a 2-layer RNNs, all the hidden states of E-RNN depend on all the inputs seen so far. Furthermore, unlike a Bi-RNN, for causal prediction, E-RNN depends on both the forward and backward hidden states. Importantly, E-RNN is a general architecture that can be combined with various RNN techniques such as multi-layer, dropout, and GRU. Using three real-world datasets, we show that E-RNN significantly and consistently improves the performance of previous RNN architectures with the same complexity.

1. Introduction

Recurrent neural networks (RNNs) are standard architectures aimed at dealing with time-series data streams. RNN is applied in a variety of applications such as clinical risk scoring (Che et al., 2016), speech recognition (Sak et al., 2014), language translation (Auli et al., 2013), traffic forecasting (Yu et al., 2017) etc. Standard RNNs (S-RNNs) (Rumelhart et al., 1988) can propagate the hidden states only in the *forward* direction. Therefore, S-RNNs cannot utilize *future* inputs for *current* prediction. To utilize future inputs for current prediction (i.e. issue non-causal predictions), bi-directional RNNs (Bi-RNNs) (Schuster & Paliwal, 1997) were proposed. Briefly described, Bi-RNN adds a backward hidden layer on top of the forward hidden layer of the S-RNN and use the concatenated states of forward and backward hidden states to issue non-causal predictions. Therefore, with Bi-RNN, the prediction at time t depends on both the past inputs (i.e. inputs at time $\tau \leq t$) and the future inputs (i.e. inputs at time $\tau > t$).

In this paper, we focus on issuing causal predictions which only depend on the past inputs. As will be shown later, for causal prediction, the structure of Bi-RNN reduces to standard RNNs since the current output is *independent* of all the previous backward hidden states.

To address this limitation of Bi-RNNs for causal prediction, we propose a novel RNN architecture, called Entangled RNN (E-RNN). By stacking an additional forward hidden layer on top of Bi-RNN structure, the causal prediction of E-RNN is *dependent* on all the previous backward hidden states. E-RNN can be used in a plethora of applications, ranging from medicine to the finance.

Importantly, E-RNN can be combined with various state-of-the-art RNN techniques such as multi-layer (Parlos et al., 1994), dropout (Srivastava et al., 2014), LSTM (Hochreiter & Schmidhuber, 1997), and GRU (Chung et al., 2015) and leads to performance gains, without the need for any additional assumptions.

2. Problem Formulation

The dataset consists of N arrays. The n-th array is denoted as $\mathcal{X}(n) = \{\mathbf{x}_1(n), \mathbf{x}_2(n), ..., \mathbf{x}_T(n)\}$, where $\mathbf{x}_t(n) \in \mathbb{R}^D$ is a measurement vector at time t, D is the dimension of data streams, and $T(n)$ is the total number of time stamps. The entire data streams can be denoted as $\mathcal{D} = \{\mathcal{X}(n)\}_{n=1}^N = \{\mathbf{x}_1(n), \mathbf{x}_2(n), ..., \mathbf{x}_{T(n)}(n)\}_{n=1}^N$. For each array (i.e. for each instance) $\mathcal{X}(n)$, we define the corresponding label $y(n) \in \mathcal{Y} = \mathbb{R}$ for regression, and $y(n) \in \mathcal{Y} = \{1, 2, ..., C\}$ for classification problem. Causal prediction only uses the input $x(\tau)$ for values of $\tau \leq t$ to issue the prediction at time t. On the other hand, non-causal prediction uses inputs $x(\tau)$ for both values of $\tau \leq t$ (previous) and $\tau > t$ (future) to issue the prediction at time t.

[1]University of California, Los Angeles, California, USA. [2]University of Oxford, Oxford, United Kingdom. [3]Alan Turing Institute, London, United Kingdom. . Correspondence to: Jinsung Yoon <jsyoon0823@g.ucla.edu>.

Presented at the ICML 2017 Workshop on Principled Approaches to Deep Learning, Sydney, Australia, 2017. Copyright 2017 by the author(s).

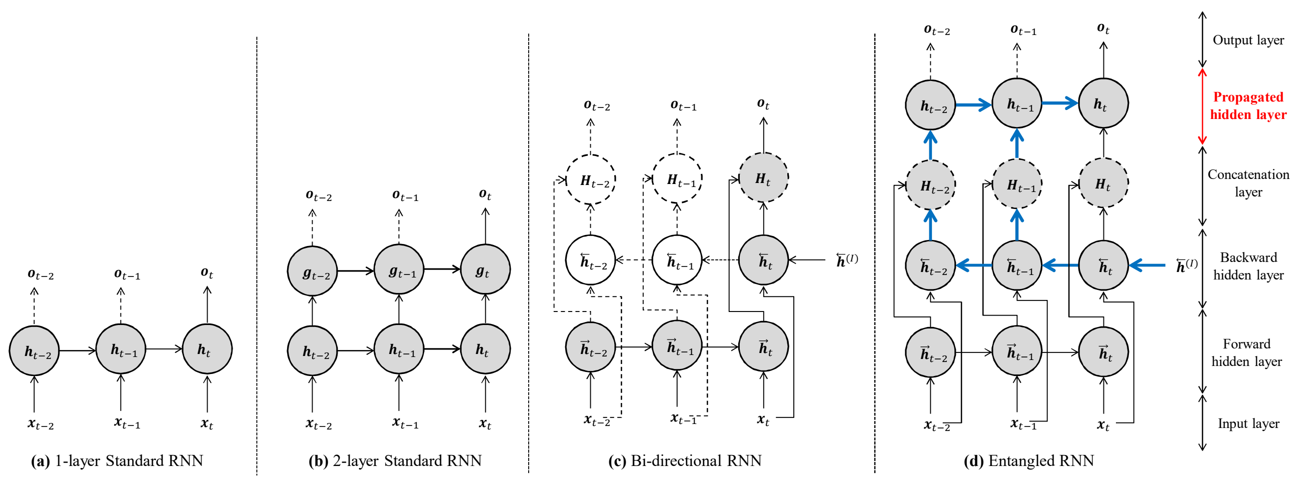

Figure 1. Architectures of RNN for causal prediction (Grey node: Dependent nodes to o_t, Dash line: Independent connection to o_t, Thick blue line: New connections by E-RNN, Dash circle: Concatenated states, $\overleftarrow{\mathbf{h}}^{(I)}$: Initial state of the backward hidden layer).

3. Entangled RNN

In this section, we introduce Entangled RNN for causal prediction. To contrast E-RNN to S-RNN and Bi-RNN, we first briefly describe the conventional RNN architectures. Then, we propose E-RNN and highlight its advantages with respect to conventional RNN architectures. Figure 1 describes the main architectural differences between S-RNN, 2-layer S-RNN, Bi-RNN, and E-RNN, respectively.

3.1. Standard RNN and Bi-directional RNN

Let us define the hidden state at time t as $\mathbf{h}_t \in \mathbb{R}^K$, where K is the dimension of each hidden state. The output state at time t is $\mathbf{o}_t \in \mathbb{R}^M$, where M is the dimension of each output state. The hidden and output states of S-RNN are updated as follows (See Figure 1(a)):

$$\begin{aligned} \mathbf{h}_t &= \sigma(b + W\mathbf{h}_{t-1} + U\mathbf{x}_t) \\ \mathbf{o}_t &= c + V\mathbf{h}_t = c + V \times \sigma(b + W\mathbf{h}_{t-1} + U\mathbf{x}_t) \end{aligned} \tag{1}$$

where $b \in \mathbb{R}^K, W \in \mathbb{R}^{K\times K}, U \in \mathbb{R}^{D\times K}, c \in \mathbb{R}^M$ and $V \in \mathbb{R}^{K\times M}$ are parameters (σ is a non-linear activation function such as sigmoid, tanh or ReLu). As can be seen in Equation 1, $\mathbf{o}_t$ depends on the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ which depends on the previous input $\mathbf{x}_{t-1}$ and the previous hidden state $\mathbf{h}_{t-2}$. Therefore, $\mathbf{o}_t$ depends on all the previous inputs $\{\mathbf{x}_\tau\}_{\tau=1}^{t}$ and is independent of the future inputs $\{\mathbf{x}_\tau\}_{\tau=t+1}^{T}$. The same dependency between causal prediction ($\mathbf{o}_t$) and the inputs is true for multi-layer standard RNN (See the Figure 1(b)).

To address this limitation, for non-causal prediction, Bi-directional RNN (Bi-RNN) was proposed, which added a backward hidden layer to the standard RNN architecture to utilize the future inputs $\{\mathbf{x}_\tau\}_{\tau=t+1}^{T}$. The hidden and output states of Bi-RNN are updated as follows (See Figure 1 (c)):

$$\begin{aligned} \overrightarrow{\mathbf{h}}_t &= \overrightarrow{\sigma}(\overrightarrow{b} + \overrightarrow{W}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{U}\mathbf{x}_t) \\ \overleftarrow{\mathbf{h}}_t &= \overleftarrow{\sigma}(\overleftarrow{b} + \overleftarrow{W}\overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{U}\mathbf{x}_t) \\ \mathbf{o}_t &= c + V\mathbf{H}_t = c + V[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] = c + \overrightarrow{V}\overrightarrow{\mathbf{h}}_t + \overleftarrow{V}\overleftarrow{\mathbf{h}}_t \\ &= c + \overrightarrow{V}\overrightarrow{\sigma}(\overrightarrow{b} + \overrightarrow{W}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{U}\mathbf{x}_t) \\ &\qquad + \overleftarrow{V}\overleftarrow{\sigma}(\overleftarrow{b} + \overleftarrow{W}\overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{U}\mathbf{x}_t) \\ &= c + \begin{bmatrix} \overrightarrow{V} & \overleftarrow{V} \end{bmatrix} \times \begin{bmatrix} \overrightarrow{\sigma}(\overrightarrow{b} + \overrightarrow{W}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{U}\mathbf{x}_t) \\ \overleftarrow{\sigma}(\overleftarrow{b} + \overleftarrow{W}\overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{U}\mathbf{x}_t) \end{bmatrix} \end{aligned} \tag{2}$$

where the right and left arrows specify the forward and backward hidden layers, respectively. [1] Equation 2 shows that $\mathbf{o}_t$ depends on the current input $\mathbf{x}_t$, the previous forward hidden state $\overrightarrow{\mathbf{h}}_{t-1}$ (which depends on the previous input $\mathbf{x}_{t-1}$, and the previous forward hidden state $\overrightarrow{\mathbf{h}}_{t-2}$), and the future backward hidden state $\overleftarrow{\mathbf{h}}_{t+1}$ (which depends on the future input $\mathbf{x}_{t+1}$, and the future backward hidden state $\overleftarrow{\mathbf{h}}_{t+2}$). Therefore, $\mathbf{o}_t$ depends on both the previous $\{\mathbf{x}_\tau\}_{\tau=1}^{t}$ and the future $\{\mathbf{x}_\tau\}_{\tau=t+1}^{T}$ inputs.

For non-causal prediction, Bi-RNN can successfully utilize the information of the backward hidden states. However, for causal prediction at time t, the backward hidden state $\overleftarrow{\mathbf{h}}_{t+1}$ cannot be used because it depends on the future inputs. As can be seen in Figure 1 (c), all the previous hidden

[1] Note that Bi-RNN is a single layer RNN structure even if it has both forward and backward hidden layers. As it can be seen in Equation 2, the forward and the backward hidden states are just concatenated and do not interact each other.

Architecture	**Causal Prediction** (Recursive Representations)	**Dependency**
S-RNN	$o_t = c + V\mathbf{h}_t \quad = c + V \times \sigma(b + W\mathbf{h}_{t-1} + U\mathbf{x}_t)$	$\{\mathbf{x}_\tau\}_{\tau=1}^t, \{\mathbf{h}_\tau\}_{\tau=1}^t$
Bi-RNN	$o_t = c + V[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] \quad = c + \overrightarrow{V}\overrightarrow{\sigma}(\overrightarrow{b} + \overrightarrow{W}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{U}\mathbf{x}_t) + \overleftarrow{V}\overleftarrow{\sigma}(\overleftarrow{b} + \overleftarrow{W}\overleftarrow{\mathbf{h}}^{(I)} + \overleftarrow{U}\mathbf{x}_t)$	$\{\mathbf{x}_\tau\}_{\tau=1}^t, \{\overrightarrow{\mathbf{h}}_\tau\}_{\tau=1}^t$
E-RNN	$o_t = c + V\mathbf{h}_t \quad = c + V \times \sigma(b + W\mathbf{h}_{t-1} + U[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t])$	$\{\mathbf{x}_\tau\}_{\tau=1}^t, \{\overrightarrow{\mathbf{h}}_\tau\}_{\tau=1}^t, \{\overleftarrow{\mathbf{h}}_\tau\}_{\tau=1}^t$

Table 1. Comparisons of S-RNN, Bi-RNN, and E-RNN for causal prediction.

states are independent of the current output. Therefore, for causal prediction, the output at time t can be re-written as

$$\mathbf{o}_t = c + \hat{V}\hat{\sigma}(\hat{b} + \hat{W}\overrightarrow{\mathbf{h}}_{t-1} + \hat{U}\mathbf{x}_t) \tag{3}$$

By looking at Equations 1 and 3, we can see that S-RNN and Bi-RNN have the same structure for causal prediction.

3.2. Entangled RNN

To enable utilizing the backward hidden states for causal predictions, we propose Entangled RNN. E-RNN stacks a forward hidden layer on top of backward hidden layer (See Figure 1 (d) - propagated hidden layer) of Bi-RNN to propagate the backward hidden states to the current output. The hidden and output states of E-RNN are updated as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{h}}_t &= \overrightarrow{\sigma}(\overrightarrow{b} + \overrightarrow{W}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{U}\mathbf{x}_t) \\
\overleftarrow{\mathbf{h}}_\tau &= \begin{cases} \overleftarrow{\sigma}(\overleftarrow{b} + \overleftarrow{W}\overleftarrow{\mathbf{h}}^{(I)} + \overleftarrow{U}\mathbf{x}_t), & \text{if } \tau = t. \\ \overleftarrow{\sigma}(\overleftarrow{b} + \overleftarrow{W}\overleftarrow{\mathbf{h}}_{\tau+1} + \overleftarrow{U}\mathbf{x}_\tau), & \text{otherwise.} \end{cases} \\
\mathbf{h}_t &= \sigma(b + W\mathbf{h}_{t-1} + U[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]) \\
\mathbf{o}_t &= c + V\mathbf{h}_t
\end{aligned} \tag{4}$$

where $\overrightarrow{\mathbf{h}}^{(I)}$ and $\overleftarrow{\mathbf{h}}^{(I)}$ are the initial states of the forward and backward hidden layer, respectively (we set both as the zero vectors). Equation 4 shows that the output $\mathbf{o}_t$ of E-RNN depends on the propagated hidden state $\mathbf{h}_t$. The propagated hidden state $\mathbf{h}_t$ depends on the previous propagated hidden state $\mathbf{h}_{t-1}$ (which depends on the previous propagated hidden state $\mathbf{h}_{t-2}$, the previous forward hidden state $\overrightarrow{\mathbf{h}}_{t-1}$, and the previous backward hidden state $\overleftarrow{\mathbf{h}}_{t-1}$), the current forward and the backward hidden states. Therefore, at time t, the output $\mathbf{o}_t$ depends on the entire inputs $\{\mathbf{x}_\tau\}_{\tau=1}^t$, forward $\{\overrightarrow{\mathbf{h}}_\tau\}_{\tau=1}^t$ and backward $\{\overleftarrow{\mathbf{h}}_\tau\}_{\tau=1}^t$ hidden states. Table 1 summarizes the differences of representations and dependencies for the causal prediction between E-RNN and conventional RNN architectures.

Importantly, E-RNN can be combined with any state-of-the-art RNN techniques without additional assumptions. For instance, each hidden layer (backward, forward, and propagated) can be iteratively stacked (multi-layer RNN). Furthermore, each hidden state can be replaced by GRU to capture long-term dependencies or add drop-out for regularization. In the next section, we quantitatively show the advantages of E-RNN compared to conventional RNNs.

4. Experiments

In this section, we experimentally show the performance of E-RNN in comparison with standard RNN (S-RNN) and Bi-RNN for causal prediction. We use three real-world datasets: (1) using physiological data streams of patients in regular hospital wards (see data description in (Alaa et al., 2017)) to predict clinical deterioration for last 24 hours, (2) using physiological data streams of patients in the intensive care units (ICU) (see data description in (Johnson et al., 2016)) to predict death (of patients) for last 24 hours, (3) using historical stock price datasets of Google to predict the future stock price of Google. (http://finance.yahoo.com/quote/GOOG)

4.1. Simulation Setting

Each hidden state uses a *sigmoid* activation function (*tanh* for regression problems) and the weights are initialized using the *Xavier* initialization method (Glorot & Bengio, 2010). The dimensions of hidden states for the forward, backward, and aggregation layers are all $2D$. Note that we use two-layer RNN architectures for standard RNN ($4D$ and $2D$ dimensions of forward hidden states for the first and the second layer, respectively) and Bi-RNN ($4D$ and $2D$ dimensions of forward/backward hidden states for the first and the second layers, respectively) to fairly compare with E-RNN with the same number of parameters (same complexity). We use a fully connected layer at the end of the output layer and *softmax* activation function to make causal classification predictions (*linear* for regression problems). We use the *Adam* optimizer (Kingma & Ba, 2014) to optimize the parameters and the binary cross-entropy as the loss function (mean square error for regression problems). The learning rate is 0.01, the number of epochs is 50, and the batch size is 100. Initial states for the backward and the

Setting	Metric	E-RNN	S-RNN (/Bi-RNN)	Gain
A (Standard)	AUROC	**0.9542**	0.9447	**17.2%**
	AUSPC	**0.7358**	0.7229	**4.7%**
B (Dropout)	AUROC	**0.9552**	0.9451	**18.4%**
	AUSPC	**0.7579**	0.7432	**5.7%**
C (GRU)	AUROC	**0.9612**	0.9550	**13.8%**
	AUSPC	**0.7712**	0.7543	**6.9%**
D (Deep)	AUROC	**0.9548**	0.9461	**16.1%**
	AUSPC	**0.7512**	0.7301	**7.8%**

Table 2. Performance comparisons using dataset of (Alaa et al., 2017) to predict clinical deterioration

Setting	Metric	E-RNN	S-RNN (/Bi-RNN)	Gain
A (Standard)	AUROC	**0.9277**	0.9135	**16.4%**
	AUSPC	**0.4962**	0.4842	**2.3%**
B (Dropout)	AUROC	**0.9445**	0.9399	**7.7%**
	AUSPC	**0.5217**	0.5059	**3.2%**
C (GRU)	AUROC	**0.9348**	0.9282	**9.2%**
	AUSPC	**0.5212**	0.5124	**1.8%**
D (Deep)	AUROC	**0.9334**	0.9257	**10.4%**
	AUSPC	**0.5033**	0.4901	**2.6%**

Table 3. Performance comparisons using dataset of (Johnson et al., 2016) to predict death (of patients)

forward hidden layers are set to the zero vector ($\overrightarrow{\mathbf{h}}^{(I)} = \mathbf{0}$, $\overleftarrow{\mathbf{h}}^{(I)} = \mathbf{0}$). We call this the standard setting (**Setting A**).

To highlight the general applicability of E-RNN, we add various architectural changes on top of the standard setting:
Setting B: Adds dropout (p = 0.75) in all the hidden layers;
Setting C: Replaces each hidden state from the standard RNN cell to GRU cell with the tanh activation function;
Setting D: Doubles the depth of hidden layers.
All the other settings are exactly the same as in **Setting A**.

We use two performance metrics to evaluate the accuracy of causal prediction for classification problems: area under the receiver operating characteristics curve (AUROC) and area under the sensitivity/precision curve (AUSPC). For regression problems, we use root mean square error (RMSE) as the performance metric. To highlight the improvements obtained by E-RNN, we also state the relative gain.

4.2. Simulation Results

Table 2 and 3 show the performance of E-RNN in comparison with S-RNN/Bi-RNN in terms of AUSPC and AUROC for four different settings and using two medical datasets. Since the architectures of S-RNN and Bi-RNN for causal prediction coincide (only forward layers affect the output as explained in section 3), the performance of S-RNN and Bi-RNN is the same. As it can be seen, E-RNN consistently outperforms in the various settings. More specifically, with **Setting A**, the gains are 17.2% and 4.7% in comparison with S-RNN/Bi-RNN in terms of AUROC and AUSPC, respectively. Furthermore, by successfully utilizing the backward hidden states, E-RNN consistently improves the performance of various RNN architectures (dropout, GRU, and multi-layer) for causal prediction.

Table 4 represents the performance of E-RNN for stock price prediction. (This is a regression problem). In terms of RMSE, E-RNN consistently outperforms S-RNN/Bi-RNN. Furthermore, Figure 2 shows the learning curve of both E-RNN and S-RNN/Bi-RNN for setting A. This figure shows that the validation loss of E-RNN converges to a lower value than that of the S-RNN/Bi-RNN due to its utilization.

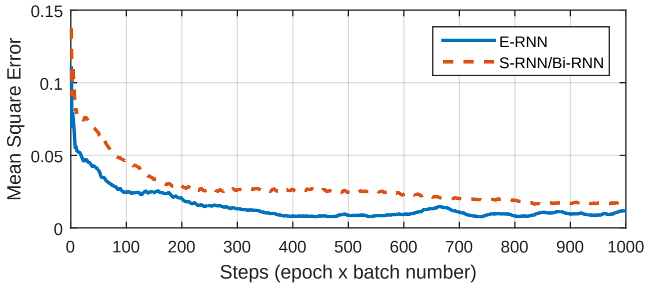

Figure 2. Learning for stock price prediction (setting A).

5. Conclusion

We proposed Entangled RNN, a novel RNN architecture for causal prediction. Experiments show that E-RNNs obtain consistent performance improvements compared to standard RNN for both classification and regression problems. E-RNN can be applied in a plethora of applications, ranging from medicine to the finance.

Setting (Metric: RMSE)	E-RNN	S-RNN (/Bi-RNN)	Gain
A (Standard)	**0.1098**	0.1268	**13.4%**
B (Dropout)	**0.1043**	0.1198	**12.9%**
C (GRU)	**0.0640**	0.0702	**8.8%**
D (Deep)	**0.1101**	0.1215	**9.4%**

Table 4. Performance comparisons for stock price prediction.

References

Alaa, Ahmed M, Yoon, Jinsung, Hu, Scott, and van der Schaar, Mihaela. Personalized risk scoring for critical care prognosis using mixtures of gaussian processes. *IEEE Transactions on Biomedical Engineering*, 2017.

Auli, Michael, Galley, Michel, Quirk, Chris, and Zweig, Geoffrey. Joint language and translation modeling with recurrent neural networks. In *EMNLP*, volume 3, pp. 0, 2013.

Che, Zhengping, Purushotham, Sanjay, Cho, Kyunghyun, Sontag, David, and Liu, Yan. Recurrent neural networks for multivariate time series with missing values. *arXiv preprint arXiv:1606.01865*, 2016.

Chung, Junyoung, Gulcehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pp. 2067–2075, 2015.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pp. 249–256, 2010.

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

Johnson, Alistair EW, Pollard, Tom J, Shen, Lu, Lehman, Li-wei H, Feng, Mengling, Ghassemi, Mohammad, Moody, Benjamin, Szolovits, Peter, Celi, Leo Anthony, and Mark, Roger G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Parlos, Alexander G, Chong, Kil To, and Atiya, Amir F. Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks*, 5(2):255–266, 1994.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Sak, Haşim, Senior, Andrew, and Beaufays, Françoise. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Yu, Rose, Li, Yaguang, Shahabi, Cyrus, Demiryurek, Ugur, and Liu, Yan. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM), Houston, TE, USA*, pp. 27–29, 2017.