Exploring Interpretable LSTM Neural Networks over Multi-Variable Data

Tian Guo [1] **Tao Lin** [2] **Nino Antulov-Fantulin** [1]

Abstract

For recurrent neural networks trained on time series with target and exogenous variables, in addition to accurate prediction, it is also desired to provide interpretable insights into the data. In this paper, we explore the structure of LSTM recurrent neural networks to learn variable-wise hidden states, with the aim to capture different dynamics in multi-variable time series and distinguish the contribution of variables to the prediction. With these variable-wise hidden states, a mixture attention mechanism is proposed to model the generative process of the target. Then we develop associated training methods to jointly learn network parameters, variable and temporal importance w.r.t the prediction of the target variable. Extensive experiments on real datasets demonstrate enhanced prediction performance by capturing the dynamics of different variables. Meanwhile, we evaluate the interpretation results both qualitatively and quantitatively. It exhibits the prospect as an end-to-end framework for both forecasting and knowledge extraction over multi-variable data.

1. Introduction

Recently, recurrent neural networks (RNNs), especially long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRU) (Cho et al., 2014), have been proven to be powerful sequence modeling tools in various tasks e.g. language modelling, machine translation, health informatics, time series, and speech (Ke et al., 2018; Lin et al., 2017; Guo et al., 2016; Lipton et al., 2015; Sutskever et al., 2014; Bahdanau et al., 2014). In this paper, we focus on RNNs over multi-variable time series consisting of target and exogenous variables. RNNs trained over such multi-variable data capture nonlinear correlation of historical values of target and exogenous variables to the future target values.

[1]ETH, Zürich, Switzerland [2]EPFL, Switzerland. Correspondence to: Tian Guo <tian.guo@gess.ethz.ch>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

In addition to forecasting, interpretable RNNs are desirable for gaining insights into the important part of data for RNNs achieving good prediction performance (Hu et al., 2018; Foerster et al., 2017; Lipton, 2016). In this paper, we focus on two types of importance interpretation: variable importance and variable-wise temporal importance. First, in RNNs variables differ in predictive power on the target, thereby contributing differently to the prediction (Feng et al., 2018; Riemer et al., 2016). Second, variables also present different temporal relevance to the target one (Kirchgässner et al., 2012). For instance, for a variable instantaneously correlated to the target, its short historical data contributes more to the prediction. The ability to acquire this knowledge enables additional applications, e.g. variable selection.

However, current RNNs fall short of the aforementioned interpretability for multi-variable data due to their opaque hidden states. Specifically, when fed with the multi-variable observations of the target and exogenous variables, RNNs blindly blend the information of all variables into the hidden states used for prediction. It is intractable to distinguish the contribution of individual variables into the prediction through the sequence of hidden states (Zhang et al., 2017).

Meanwhile, individual variables typically present different dynamics. This information is implicitly neglected by the hidden states mixing multi-variable data, thereby potentially hindering the prediction performance.

Existing works aiming to enhance the interpretability of recurrent neural networks rarely touch the internal structure of RNNs to overcome the opacity of hidden states on multi-variable data. They still fall short of aforementioned two types of interpretation (Montavon et al., 2018; Foerster et al., 2017; Che et al., 2016). One category of the approaches is to perform post-analyzing on trained RNNs by perturbation on training data or gradient based methods (Ancona et al., 2018; Ribeiro et al., 2018; Lundberg & Lee, 2017; Shrikumar et al., 2017). Another category is to build attention mechanism on hidden states of RNNs to characterize the importance of different time steps (Qin et al., 2017; Choi et al., 2016).

In this paper we aim to achieve a unified framework of accurate forecasting and importance interpretation. In particular, the contribution is fourfold:

- We explore the structure of LSTM to enable variable-wise hidden states capturing individual variable's dynamics. It facilitates the prediction and interpretation. This family of LSTM is referred to as **I**nterpretable **M**ulti-**V**ariable LSTM, i.e. IMV-LSTM.

- A novel mixture attention mechanism is designed to summarize variable-wise hidden states and model the generative process of the target.
- We develop a training method based on probabilistic mixture attention to learn network parameter, variable and temporal importance measures simultaneously.
- Extensive experimental evaluation of IMV-LSTM against statistical, machine learning and deep learning based baselines demonstrate the superior prediction performance and interpretability of IMV-LSTM. The idea of IMV-LSTM easily applies to other RNN structures, e.g. GRU and stacked recurrent layers.

2. Related Work

Recent research on the interpretable RNNs can be categorized into two groups: attention methods and post-analyzing on trained models. Attention mechanism has gained tremendous popularity (Xu et al., 2018; Choi et al., 2018; Guo et al., 2018; Lai et al., 2017; Qin et al., 2017; Cinar et al., 2017; Choi et al., 2016; Vinyals et al., 2015; Bahdanau et al., 2014). However, current attention mechanism is mainly applied to hidden states across time steps. Qin et al. (2017); Choi et al. (2016) built attention on conventional hidden states of encoder networks. Since the hidden states encode information from all input variables, the derived attention is biased when used to measure the importance of corresponding variables. The contribution coefficients defined on attention values is biased as well (Choi et al., 2016). Moreover, weighting input data by attentions (Xu et al., 2018; Qin et al., 2017; Choi et al., 2016) does not consider the direction of correlation with the target, which could impair the prediction performance. Current attention based methods seldom provide variable-wise temporal interpretability.

As for post-analyzing interpretation, Murdoch et al. (2018); Murdoch & Szlam (2017); Arras et al. (2017) extracted temporal importance scores over words or phrases of individual sequences by decomposing the memory cells of trained RNNs. In perturbation-based approaches perturbed samples might be different from the original data distribution (Ribeiro et al., 2018). Gradient-based methods analyze the features that output was most sensitive to (Ancona et al., 2018; Shrikumar et al., 2017). Above methods mostly focused on one type of importance and are computationally inefficient. They rarely enhance the predicting performance.

Wang et al. (2018) focused on the importance of each middle layer to the output. Chu et al. (2018) proposed interpreting solutions for piece-wise linear neural networks. Foerster et al. (2017) introduced input-switched linear affine transformations into RNNs to analyze the contribution of input steps, wihch could lead to the loss in prediction performance. Our paper focuses on exploring the internal structure of LSTM so as to learn accurate forecasting and importance measures simultaneously.

Another line of related research is about tensorization and decomposition of hidden states in RNNs. Do et al. (2017); Novikov et al. (2015) proposed to represent hidden states as matrices. He et al. (2017) developed tensorized LSTM to enhance the capacity of networks without additional parameters. Kuchaiev & Ginsburg (2017); Neil et al. (2016); Koutnik et al. (2014) proposed to partition the hidden layer into separated modules with different updates. These hidden state tensors and update processes do not maintain variable-wise correspondence and lack the desirable interpretability.

3. Interpretable Multi-Variable LSTM

In the following we will first explore the internal structure of LSTM to enable hidden states to encode individual variables, such that the contribution from individual variables to the prediction can be distinguished. Then, mixture attention is designed to summarize these variable-wise hidden states for predicting. The described method can be easily extended to multi-step ahead prediction via iterative methods as well as vector regression (Fox et al., 2018; Cheng et al., 2006).

Assume we have N-1 exogenous time series and a target series $\mathbf{y}$ of length T, where $\mathbf{y} = [y_1, \cdots, y_T]$ and $\mathbf{y} \in \mathsf{R}^T$. Vectors are assumed to be in column form throughout this paper. By stacking exogenous time series and target series, we define a multi-variable input series as $\mathbf{X}_T = \{\mathbf{x}_1, \cdots, \mathbf{x}_T\}$, where $\mathbf{x}_t = [\mathbf{x}_t^1, \cdots, \mathbf{x}_t^{N-1}, y_t]$. Both of $\mathbf{x}_t^n$ and y_t can be multi-dimensional vector. $\mathbf{x}_t \in \mathsf{R}^N$ is the multi-variable input at time step t. It is also free for $\mathbf{X}_T$ to merely include exogenous variables, which does not affect the methods presented below.

Given $\mathbf{X}_T$, we aim to learn a non-linear mapping to predict the next values of the target series, namely $\hat{y}_{T+1} = \mathcal{F}(\mathbf{X}_T)$.

Meanwhile, the other desirable byproduct of learning $\mathcal{F}(\mathbf{X}_T)$ is the variable and temporal importance measures. Mathematically, we aim to derive variable importance vector $\mathbf{I} \in R^N_{0}$, $\sum_{n=1}^{N} \mathrm{I}_n = 1$ and variable-wise temporal importance vector $\mathbf{T}^n \in R^{T-1}_{0}$ (w.r.t. variable n), $\sum_{k=1}^{T-1} \mathrm{T}_k^n = 1$. Elements of these vectors are normalized (i.e. sum to one) and reflect the relative importance of the corresponding variable or time instant w.r.t. the prediction.

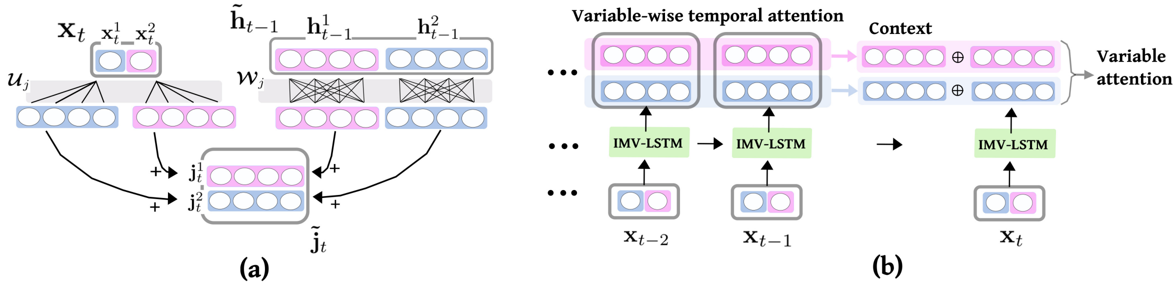


Figure 1: A toy example of a IMV-LSTM with a two-variable input sequence and the hidden matrix of 4-dimensions per variable. Circles represent one dimensional elements. Purple and blue colors correspond to two variables. Blocks containing rectangles with circles inside represent input data and hidden matrix. Panel (a) exhibits the derivation of hidden update $\tilde{\mathbf{j}}_t$. Grey areas represent transition weights. Panel (b) demonstrates the mixture attention process. (best viewed in color)

3.1. Network Architecture

The idea of IMV-LSTM is to make use of hidden state matrix and to develop associated update scheme, such that each element (e.g. row) of the hidden matrix encapsulates information exclusively from a certain variable of the input.

To distinguish from the hidden state and gate vectors in a standard LSTM, hidden state and gate matrices in IMV-LSTM are denoted with tildes. Specifically, we define the hidden state matrix at time step t as $\tilde{\mathbf{h}}_t = [\mathbf{h}^1_t, \cdots, \mathbf{h}^N_t]^{>}$, where $\tilde{\mathbf{h}}_t \in \mathbb{R}^{N\ \ d}$, $\mathbf{h}^n_t \in \mathbb{R}^d$. The overall size of the layer is derived as $D = N \cdot d$. The element $\mathbf{h}^n_t$ of $\tilde{\mathbf{h}}_t$ is the hidden state vector specific to n-th input variable.

Then, we define the input-to-hidden transition as $\boldsymbol{\mathcal{U}}_j = [\mathbf{U}^1_j, \cdots, \mathbf{U}^N_j]^{>}$, where $\boldsymbol{\mathcal{U}}_j \in \mathbb{R}^{N\ \ d\ \ d_0}$, $\mathbf{U}^n_j \in \mathbb{R}^{d\ \ d_0}$ and d_0 is the dimension of individual variables at each time step. The hidden-to-hidden transition is defined as: $\boldsymbol{\mathcal{W}}_j = [\mathbf{W}^1_j \cdots \mathbf{W}^N_j]$, where $\boldsymbol{\mathcal{W}}_j \in \mathbb{R}^{N\ \ d\ \ d}$ and $\mathbf{W}^n_j \in \mathbb{R}^{d\ \ d}$.

As standard LSTM neural networks (Hochreiter & Schmidhuber, 1997), IMV-LSTM has the input $\mathbf{i}_t$, forget $\mathbf{f}_t$, output gates $\mathbf{o}_t$ and the memory cells $\mathbf{c}_t$ in the update process. Given the newly incoming input $\mathbf{x}_t$ at time t and the hidden state matrix $\tilde{\mathbf{h}}_{t\ \ 1}$, the hidden state update is defined as:

$$\tilde{\mathbf{j}}_t = \tanh\left(\boldsymbol{\mathcal{W}}_j \circledast \tilde{\mathbf{h}}_{t\ \ 1} + \boldsymbol{\mathcal{U}}_j \circledast \mathbf{x}_t + \mathbf{b}_j\right), \qquad (1)$$

where $\tilde{\mathbf{j}}_t = [\mathbf{j}^1_t, \cdots, \mathbf{j}^N_t]^{>}$ has the same shape as hidden state matrix $\mathbb{R}^{N\ \ d}$. Each element $\mathbf{j}^n_t \in \mathbb{R}^d$ corresponds to the update of the hidden state w.r.t. input variable n. Term $\boldsymbol{\mathcal{W}}_j \circledast \tilde{\mathbf{h}}_{t\ \ 1}$ and $\boldsymbol{\mathcal{U}}_j \circledast \mathbf{x}_t$ respectively capture the update from the hidden states at the previous step and the new input. The tensor-dot operation $\circledast$ is defined as the product of two tensors along the N axis, e.g., $\boldsymbol{\mathcal{W}}_j \circledast \tilde{\mathbf{h}}_{t\ \ 1} = [\mathbf{W}^1_j \mathbf{h}^1_{t\ \ 1}, \cdots, \mathbf{W}^N_j \mathbf{h}^N_{t\ \ 1}]^{>}$ where $\mathbf{W}^n_j \mathbf{h}^n_{t\ \ 1} \in \mathbb{R}^d$.

Depending on different update schemes of gates and memory cells, we proposed two realizations of IMV-LSTM, i.e. IMV-Full in Equation set 1 and IMV-Tensor in Equation set 2. In these two sets of equations, $\text{vec}(\cdot)$ refers to the vectorization operation, which concatenates columns of a matrix into a vector. The concatenation operation is denoted by $\oplus$ and element-wise multiplication is denoted by $\odot$. Operator $\text{matricization}(\cdot)$ reshapes a vector of $\mathbb{R}^D$ into a matrix of $\mathbb{R}^{N\ \ d}$.

$$\begin{bmatrix}\mathbf{i}_t\\ \mathbf{f}_t\\ \mathbf{o}_t\end{bmatrix} = \sigma\left(\mathbf{W}\left[\mathbf{x}_t \oplus \text{vec}(\tilde{\mathbf{h}}_{t\ \ 1})\right] + \mathbf{b}\right) \qquad (2)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t\ \ 1} + \mathbf{i}_t \odot \text{vec}(\tilde{\mathbf{j}}_t) \qquad (3)$$

$$\tilde{\mathbf{h}}_t = \text{matricization}(\mathbf{o}_t \odot \tanh(\mathbf{c}_t)) \qquad (4)$$

Equation set 1: IMV-Full

$$\begin{bmatrix}\tilde{\mathbf{i}}_t\\ \tilde{\mathbf{f}}_t\\ \tilde{\mathbf{o}}_t\end{bmatrix} = \sigma\left(\boldsymbol{\mathcal{W}} \circledast \tilde{\mathbf{h}}_{t\ \ 1} + \boldsymbol{\mathcal{U}} \circledast \mathbf{x}_t + \mathbf{b}\right) \qquad (5)$$

$$\tilde{\mathbf{c}}_t = \tilde{\mathbf{f}}_t \odot \tilde{\mathbf{c}}_{t\ \ 1} + \tilde{\mathbf{i}}_t \odot \tilde{\mathbf{j}}_t \qquad (6)$$

$$\tilde{\mathbf{h}}_t = \tilde{\mathbf{o}}_t \odot \tanh(\tilde{\mathbf{c}}_t) \qquad (7)$$

Equation set 2: IMV-Tensor

IMV-Full: With vectorization in Eq. (2) and (3), IMV-Full updates gates and memories using full $\tilde{\mathbf{h}}_{\mathbf{t}\ \ 1}$ and $\tilde{\mathbf{j}}_{\mathbf{t}}$ regardless of the variable-wise data in them. By simple replacement of the hidden update in standard LSTM by $\tilde{\mathbf{j}}_t$, IMV-Full behaves identically to standard LSTM while enjoying the interpretability shown below.

IMV-Tensor: By applying tensor-dot operations in Eq. (5), gates and memory cells are matrices as well, elements of which have the correspondence to input variables as hidden state matrix $\tilde{\mathbf{h}}_t$ does. $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{U}}$ have the same shapes as $\boldsymbol{\mathcal{W}}_{\boldsymbol{j}}$ and $\boldsymbol{\mathcal{U}}_{\boldsymbol{j}}$ in Eq. (1).

In IMV-Full and IMV-Tensor, gates only scale $\tilde{\mathbf{j}}_{\mathbf{t}}$ and $\tilde{\mathbf{c}}_{t\ \ 1}$

and thus retain the variable-wise data organization in $\tilde{\mathbf{h}}_t$. Meanwhile, based on tensorized hidden state Eq. (1) and gate update Eq. (5), IMV-Tensor can also be considered as a set of parallel LSTMs, each of which processes one variable series and then merges via the mixture. The derived hidden states specific to each variable are aggregated on both temporal and variable level through the mixture attention.

Next, we provide the analysis about the complexity of IMV-LSTM through Lemma 3.1 and Lemma 3.2.

Lemma 3.1. *Given time series of N variables, assume a standard LSTM and IMV-LSTM layer both have size D, i.e. D neurons in the layer. Then, compared to the number of parameters of the standard LSTM, IMV-Full and IMV-Tensor respectively reduce the network complexity by $(N-1)D+(1-1/N)D\cdot D$ and $4(N-1)D+4(1-1/N)D\cdot D$ number of parameters.*

Proof. In a standard LSTM of layer size D, trainable parameters lie in the hidden and gate update functions. In total, these update functions have $4D\cdot D+4N\cdot D+4D$ parameters, where $4D\cdot D+4N\cdot D$ comes from the transition and $4D$ corresponds to the bias terms. For IMV-Full, assume each input variable corresponds to one-dimensional time series. Based on Eq. 1, the hidden update has $2D+D^2/N$ trainable parameters. Equation set 1 gives rise to the number of parameters equal to that of the standard LSTM. Therefore, the reduce number of parameters is $(N-1)D+(1-1/N)D\cdot D$. As for IMV-Tensor, more parameter reduction stems from that the gate update functions in Equation set 2 make use of the tensor-dot operation as Eq. 1. □

Lemma 3.2. *For time series of N variables and the recurrent layer of size D, IMV-Full and IMV-Tensor respectively have the computation complexity at each update step as: $\mathcal{O}(D^2+N\cdot D)$ and $\mathcal{O}(D^2/N+D)$.*

Proof. Assume that D neurons of the recurrent layer in IMV-Full and IMV-Tensor are evenly assigned to N input variables, namely each input variable has $d=D/N$ corresponding neurons. For IMV-Full, based on Eq. 1, the hidden update has computation complexity $N\cdot d^2+N\cdot d$, while the gate update process has the complexity $D^2+N\cdot D$. Overall, the computation complexity is $\mathcal{O}(D^2+N\cdot D)$, which is identical to the complexity of a standard LSTM. As for IMV-Tensor, since the gate update functions in Equation set 2 make use of the tensor-dot operation as Eq. 1, gate update functions have the same computation complexity as Eq. 1. The overall complexity is $\mathcal{O}(D^2/N+D)$, which is $1/N$ of the complexity of a standard LSTM. □

Basically, Lemma 3.1 and Lemma 3.2 indicate that a high number of input variables leads to a large portion of parameter and computation reduction in IMV-LSTM family.

3.2. Mixture Attention

After feeding a sequence of $\{\mathbf{x}_1,\cdots,\mathbf{x}_T\}$ into IMV-Full or IMV-Tensor, we obtain a sequence of hidden state matrices $\{\tilde{\mathbf{h}}_1,\cdots,\tilde{\mathbf{h}}_T\}$, where the sequence of hidden states specific to variable n is extracted as $\{\mathbf{h}_1^n,\cdots,\mathbf{h}_T^n\}$.

The idea of mixture attention mechanism as follows. Temporal attention is first applied to the sequence of hidden states corresponding to each variable, so as to obtain the summarized history of each variable. Then by using the history enriched hidden state of each variable, variable attention is derived to merge variable-wise states. These two steps are assembled into a probabilistic mixture model (Zong et al., 2018; Graves, 2013; Bishop, 1994), which facilitates the subsequent learning, predicting, and interpreting.

In particular, the mixture attention is formulated as:

$$
\begin{aligned}
& p(y_{T+1}\,|\mathbf{X}_T) \\
&= \sum_{n=1}^{N} p(y_{T+1}|z_{T+1}=n,\mathbf{X}_T)\cdot \Pr(z_{T+1}=n|\mathbf{X}_T) \\
&= \sum_{n=1}^{N} p(y_{T+1}\,|\,z_{T+1}=n,\mathbf{h}_1^n,\cdots,\mathbf{h}_T^n) \\
&\quad \cdot \Pr(z_{T+1}=n\,|\,\tilde{\mathbf{h}}_1,\cdots,\tilde{\mathbf{h}}_T) \\
&= \sum_{n=1}^{N} p(y_{T+1}\,|\,z_{T+1}=n,\ \underbrace{\mathbf{h}_T^n\oplus\mathbf{g}^n}_{\substack{\text{variable-wise}\\ \text{temporal attention}}}\ ) \\
&\quad \cdot \underbrace{\Pr(z_{T+1}=n\,|\,\mathbf{h}_T^1\oplus\mathbf{g}^1,\cdots,\mathbf{h}_T^N\oplus\mathbf{g}^N)}_{\text{Variable attention}}
\end{aligned}
\tag{8}
$$

In Eq. (8), we introduce a latent random variable z_{T+1} into the the density function of y_{T+1} to govern the generation process. z_{T+1} is a discrete variable over the set of values $\{1,\cdots,N\}$ corresponding to N input variables. Mathematically, $p(y_{T+1}\,|\,z_{T+1}=n,\mathbf{h}_T^n\oplus\mathbf{g}^n)$ characterizes the density of y_{T+1} conditioned on historical data of variable n, while the prior of z_{T+1}, i.e. $\Pr(z_{T+1}=n\,|\,\mathbf{h}_T^1\oplus\mathbf{g}^1,\cdots,\mathbf{h}_T^N\oplus\mathbf{g}^N)$ controls to what extent y_{T+1} is driven by variable n.

Context vector $\mathbf{g}^n$ is computed as the temporal attention weighted sum of hidden states of variable n, i.e., $\mathbf{g}^n=\sum_t \alpha_t^n\mathbf{h}_t^n$. The attention weight α_t^n is evaluated as $\alpha_t^n=\frac{\exp(\mathrm{f}_n(\mathbf{h}_t^n))}{\sum_k \exp(\mathrm{f}_n(\mathbf{h}_k^n))}$, where $\mathrm{f}_n(\cdot)$ can be a flexible function specific to variable n, e.g. neural networks.

For $p(y_{T+1}\,|\,z_{T+1}=n,\mathbf{h}_T^n\oplus\mathbf{g}^n)$, without loss of generality, we use a Gaussian output distribution parameterized by $[\,\mu_n,\sigma_n]=\varphi_n(\,\mathbf{h}_T^n\oplus\mathbf{g}^n)$, where $\varphi_n(\cdot)$ can be a feed-forward neural network. It is free to use other distributions.

$\Pr(z_{T+1}=n\,|\,\mathbf{h}_T^1\oplus\mathbf{g}^1,\cdots,\mathbf{h}_T^N\oplus\mathbf{g}^N)$ is derived by a softmax function over $\{\mathrm{f}(\,\mathbf{h}_T^n\oplus\mathbf{g}^n)\}_N$, where $\mathrm{f}(\cdot)$ can be a feedforward neural network shared by all variables.

3.3. Learning to Interpret and Predict

In the learning phase, the set of parameters in the neural network and mixture attention is denoted by Θ. Given a set of M training sequences $\{\mathbf{X}_T\}_M$ and $\{y_{T+1}\}_M$, we aim to learn both Θ and importance vectors $\mathbf{I}$ and $\{\mathbf{T}^n\}_N$ for prediction and insights into the data.

Next, we first illustrate the burden of directly interpreting attention values and then present the training method combining parameter and importance vector learning, without the need of post analyzing.

Importance vectors $\mathbf{I}$ and $\{\mathbf{T}^n\}_N$ reflect the global relations in variables, while the attention values derived above are specific to data instances. Moreover, it is nontrivial to decipher variable and temporal importance from attentions.

For instance, during the training on PLANT dataset used in the experiment section, we collect variable and variable-wise temporal attention values of training instances. In Fig 2, left panels plots the histograms of variable attention of three variables in PLANT at two different epochs. It is difficult to fully discriminate variable importance from these histograms. Likewise, in the right panel, histograms of temporal attentions at certain time lags of variable "P-temperature" at two different epochs does not ease the importance interpretation. Time lag represents the look-back time step w.r..t the current one. Similar phenomena are observed in other variables and datasets during the experiments.

In the following, we develop the training procedure based on the Expectation–Maximization (EM) framework for the probabilistic model with latent variables, i.e. Eq. (8) in this paper. Index m in the following corresponds to the training data instance. It is omitted in $\mathbf{h}_T^n$ and $\mathbf{g}^n$ for simplicity.

The loss function to minimize is derived as:

$$
\begin{aligned}
&\mathcal{L}(\Theta, \mathbf{I}) = \\
&\quad -\sum_{m=1}^{M} \mathsf{E}_{q_m^n}\left[\log p(y_{T+1,m} \,|\, z_{T+1,m} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n)\right] \\
&\quad - \mathsf{E}_{q_m^n}\left[\log \Pr(z_{T+1,m} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \cdots, \mathbf{h}_T^N \oplus \mathbf{g}^N)\right] \\
&\quad - \mathsf{E}_{q_m^n}\left[\log \Pr(z_{T+1,m} = n \,|\, \mathbf{I})\right]
\end{aligned}
\tag{9}
$$

The desirable property of this loss function is as:

Lemma 3.3. *The negative log-likelihood defined by Eq. (8) is upper-bounded by the loss function Eq. (9) in the EM process:*

$$
-\log \prod_m p(y_{T+1,m} \,|\mathbf{X}_{T,m}\,; \Theta) \le \mathcal{L}(\Theta, \mathbf{I})
$$

The proof is provided in the supplementary material.

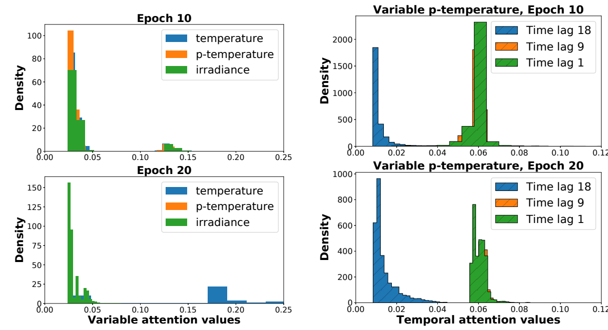


Figure 2: Left: Histograms of variable attention at different epochs. Right: Histograms of temporal attention of variable "P-temperature". It is nontrivial to interpret variable and temporal importance from attention values on training instances.

Therefore, minimizing Eq. (9) enables to simultaneously learn the network parameters and importance vectors without the need of post processing on trained networks.

In particular, in Eq. (9) the first two terms are derived from the standard EM procedure. For the last term $\mathsf{E}_{q_m^n}[\log \Pr(z_{T+1,m} = n \,|\, \mathbf{I})$, intuitively it serves as a regularization on the posterior of latent variable $z_{T+1,m}$ and encourages individual instances to follow the global pattern $\mathbf{I}$, which parameterizes a discrete distribution on $z_{T+1,m}$. And q_m^n represents the posterior of $z_{T+1,m}$ by Eq. (10):

$$
\begin{aligned}
&q_m^n := \Pr(z_{T+1,m} = n |\, \mathbf{X}_{T,m}, y_{T+1,m}\,; \Theta) \\
&\propto p(y_{T+1,m}|z_{T+1,m} = n, \mathbf{X}_{T,m}) \cdot \Pr(z_{T+1,m} = n|\mathbf{X}_{T,m}) \\
&\approx p(y_{T+1,m} \,|\, z_{T+1,m} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n) \\
&\quad \cdot \Pr(z_{T+1,m} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \cdots, \mathbf{h}_T^N \oplus \mathbf{g}^N)
\end{aligned}
\tag{10}
$$

During the training phase, network parameters Θ and importance vectors are alternatively learned. In a certain round of the loss function minimization, we first fix the current value of Θ and evaluate q_m^n for the batch of data. Then, since the first two terms in the loss functions solely depend on network parameters Θ, they are minimized via gradient descent to update Θ. For the last term, fortunately we can derive a simple closed-form solution of $\mathbf{I}$ as:

$$
\mathbf{I} = \tfrac{1}{M} \textstyle\sum_m \mathbf{q}_m, \;\; \mathbf{q}_m = [q_m^1, \cdots, q_m^n]^\top \tag{11}
$$

, which takes into account both variable attention and predictive likelihood in the importance vector.

As for temporal importance, it can also be derived through EM, but it requires a hierarchical mixture. For the sake of computing efficiency, the variable-wise temporal importance vector is derived from attention values as follows:

$$
\mathbf{T}^n = \tfrac{1}{M} \textstyle\sum_m \boldsymbol{\alpha}_m^n, \;\; \boldsymbol{\alpha}_m = [\alpha_{1,m}^n, \cdots, \alpha_{T,m}^n]^\top \tag{12}
$$

This process iterates until convergence. After the training, we obtain the neural networks ready for predicting as well as the variable and temporal importance vectors. Then, in the predicting phase, the prediction of y_{T+1} is obtained by the weighted sum of means as:

$$\hat{y}_{T+1} = \sum_n \mu_n \Pr(z_{T+1} = n j \mathbf{h}_T^1 \quad \mathbf{g}^1, \quad , \mathbf{h}_T^N \quad \mathbf{g}^N). \quad (13)$$

4. Experiments

4.1. Datasets

PM2.5: It contains hourly PM2.5 data and the associated meteorological data in Beijing of China. PM2.5 measurement is the target series. The exogenous time series include dew point, temperature, pressure, combined wind direction, accumulated wind speed, hours of snow, and hours of rain. Totally we have $41,700$ multi-variable sequences.

PLANT: This records the time series of energy production of a photo-voltaic power plant in Italy (Ceci et al., 2017). Exogenous data consists of 9 weather conditions variables (such as temperature, cloud coverage, etc.). The power production is the target. It provides 20842 sequences split into training (70%), validation (10%) and testing sets (20%).

SML is a public dataset used for indoor temperature forecasting. Same as (Qin et al., 2017), the room temperature is taken as the target series and another 16 time series are exogenous series. The data were sampled every minute. The first 3200, the following 400 and the last 537 data points are respectively used for training, validation, and test.

Due to the page limitation, experimental results on additional datasets are in the supplementary material.

4.2. Baselines and Evaluation Setup

The first category of statistics baselines includes:

STRX is the structural time series model with exogenous variables (Scott & Varian, 2014; Radinsky et al., 2012). It is consisted of unobserved components via state space models.

ARIMAX is the auto-regressive integrated moving average with regression terms on exogenous variables (Hyndman & Athanasopoulos, 2014). It is a special case of vector auto-regression in this scenario.

The second category of machine learning baselines includes:

RF refers to random forests, an ensemble learning method consisting of several decision trees (Liaw et al., 2002) and was used in time series prediction (Patel et al., 2015).

XGT refers to the extreme gradient boosting (Chen & Guestrin, 2016). It is the application of boosting methods to regression trees (Friedman, 2001).

ENET represents Elastic-Net, which is a regularized regression method combining both L1 and L2 penalties of the lasso and ridge methods (Zou & Hastie, 2005) and used in time series analysis (Liu et al., 2010; Bai & Ng, 2008).

The third category of deep learning baselines includes:

RETAIN uses RNNs to respectively learn weights on input data for predicting (Choi et al., 2016). It defines contribution coefficients on attentions to represent feature importance.

DUAL is an encoder-decoder architecture using an encoder to learn attentions and feeding pre-weighted input data into a decoder for forecasting (Qin et al., 2017). It uses temporal-wise variable attentions to reflect variable importance.

In ARIMAX, the orders of auto-regression and moving-average terms are set as the window size of the training data. For RF and XGT, hyper-parameter tree depth and the number of iterations are chosen from range $[3, 10]$ and $[2, 200]$ via grid search. For XGT, L2 regularization is added by searching within $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$. As for ENET, the coefficients for L2 and L1 penalties are selected from $\{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1, 2\}$. For machine learning baselines, multi-variable input sequences are flattened into feature vectors.

We implemented IMV-LSTM and deep learning baselines with Tensorflow. We used Adam with the mini-batch size 64 (Kingma & Ba, 2014). For the size of recurrent and dense layers in the baselines, we conduct grid search over $\{16, 32, 64, 128, 256, 512\}$. The size of IMV-LSTM layers is set by the number of neurons per variable selected from $\{10, 15, 20, 25\}$. Dropout is selected in $\{0, 0.2, 0.5\}$. Learning rate is searched in $\{0.0005, 0.001, 0.005, 0.01, 0.05\}$. L2 regularization is added with the coefficient chosen from $\{0.0001, 0.001, 0.01, 0.1, 1.0\}$. We train each approach 5 times and report average performance. The window size (i.e. T) for PM2.5 and SML is set to 10 according to (Qin et al., 2017), while for PLANT it is 20 to test long dependency.

We consider two metrics to measure the prediction performance. RMSE is defined as RMSE $=\sqrt{\sum_k (y_k - \hat{y}_k)^2 / K}$. MAE is defined as MAE $= \sum_k |y_k - \hat{y}_k| / K$.

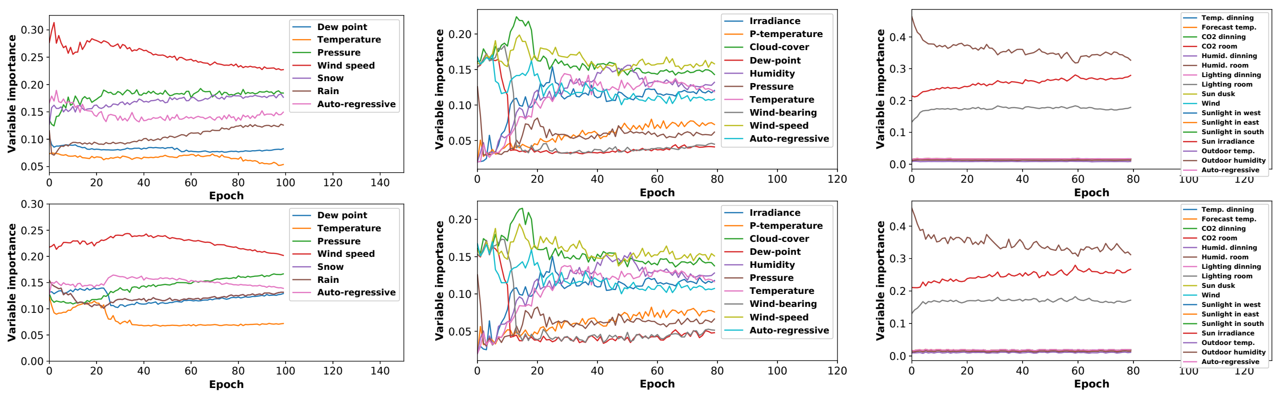
4.3. Prediction Performance

We report the prediction errors in Table 1, each cell of which presents the average RMSE and MAE with standard errors. In particular, IMV-LSTM family outperforms baselines by around 80% at most. Deep learning baselines mostly outperform other baselines. Boosting method XGT presents comparable performance with deep learning baselines in PLANT and SML datasets.

Insights. For multi-variable data carrying different patterns, properly modeling individual variables and their interaction is important for the prediction performance. IMV-Full keeps the variable interaction in the gate updating. IMV-Tensor maintains variable-wise hidden states independently and only captures their interaction via the mixture attention. Experimentally, IMV-Full and IMV-Tensor present comparable

Table 1: RMSE and MAE with std. errors

Dataset	PM2.5			PLANT			SML		
STRX	52.51	0.82, 47.35	0.92	231.43	0.19, 193.23	0.43	0.039	0.001, 0.033	0.001
ARIMAX	42.51	1.13, 40.23	0.83	225.54	0.23, 193.42	0.41	0.060	0.002, 0.053	0.002
RF	38.84	1.12, 22.27	0.63	164.23	0.65, 130.90	0.15	0.045	0.001, 0.032	0.001
XGT	25.28	1.01, 15.93	0.72	164.10	0.54, 131.47	0.21	0.017	0.001, 0.013	0.001
ENET	26.31	1.33, 15.91	0.51	168.22	0.49, 137.04	0.38	0.018	0.001, 0.015	0.001
DUAL	25.31	0.91, 16.21	0.42	163.29	0.54, 130.87	0.12	0.019	0.001, 0.015	0.001
RETAIN	31.12	0.97, 20.11	0.76	250.69	0.36, 190.11	0.15	0.048	0.001, 0.037	0.001
IMV-Full	24.47	0.34, 15.23	0.61	157.32	0.21, 128.42	0.15	0.015	0.002, 0.012	0.001
IMV-Tensor	**24.29**	**0.45, 14.87**	**0.44**	**156.32**	**0.31, 127.42**	**0.21**	**0.009**	**0.0009, 0.006**	**0.0005**



(a) PM2.5 (b) PLANT (c) SML

Figure 3: Variable importance over epochs during the training. Top and bottom panels in each sub-figure respectively correspond to IMV-Full and IMV-Tensor. In Sec. 4.4., we provide evidence from domain knowledge and show the agreement with the discoveries by IMV-Full and IMV-Tensor.

performance, though mixture on independent variable-wise hidden states in IMV-Tensor leads to the best performance. Note that IMV-Full and IMV-Tensor are of single network structure. Instead of composite network architectures in baselines, mixture of well-maintained variable-wise hidden states in IMV-LSTM also improves the prediction performance and empowers the interpretability shown below.

4.4. Interpretation

In this part, we qualitatively analyze the meaningfulness of variable and temporal importance. Fig. 3 and Fig. 4 respectively show the variable and temporal importance values during the training under the best hyper-parameters. The importance values learned by IMV-Full and IMV-Tensor could be slightly different, because in IMV-Tensor the gate and memory update scheme evolve independently, thereby leading to different hidden states to IMV-Full. IMV-LSTM is easier to understand, compared to baseline RETAIN and DUAL, since they do not show in global level importance interpretation like in Fig. 3 and 4.

Variable importance. In Fig. 3, top and bottom panels in each sub-fig show the variable importance values w.r.t. training epochs from IMV-Full and IMV-Tensor. Overall, variable importance values converge during the training and the ranking of variable importance is identified at the end of the training. Variables with high importance values contribute more to the prediction of IMV-LSTM.

In Fig. 3(a), for PM2.5 dataset, variables "Wind speed", "Pressure", "Snow", "Rain" are high ranked by IMV-LSTM. According to a recent work studying air pollution (Liang et al., 2015), "Dew Point" and "Pressure" are both related to PM2.5 and they are also inter-correlated. One "Pressure" variable is enough to learn accurate forecasting and thus has the high importance value. Strong wind can bring dry and fresh air. "Snow" and "Rain" amount are related to the air quality as well. Variables important for IMV-LSTM are in line with the domain knowledge in (Liang et al., 2015).

Fig. 3(b) shows that in PLANT dataset in addition to "Irradiance" and "Cloud cover", "Wind speed", "Humidity" as well as "Temperature" are also high ranked and relatively used more in IMV-LSTM to provide accurate forecasting. As is discussed in (Mekhilef et al., 2012; Ghazi & Ip, 2014), humidity causes dust deposition and consequentially degradation in solar cell efficiency. Increased wind can move heat from the cell surface, which leads to better efficiency.

Fig. 3(c) demonstrates that variables "Humid. room", "CO2 room", and "Lighting room" are relatively more important

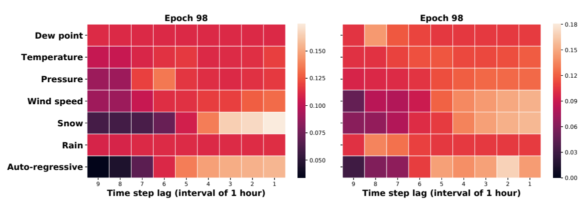
Table 2: RMSE and MAE with std. errors under top 50% important variables

Dataset	PM2.5			PLANT			SML		
DUAL	25.09	0.04, 15.59	0.08	171.30	0.17, 154.15	0.20	0.026	0.002, 0.018	0.002
RETAIN	49.25	0.11, 34.03	0.24	226.38	0.72, 167.90	0.81	0.060	0.001, 0.044	0.004
IMV-Full-P	25.14	0.54, 16.01	0.52	165.04	0.08, 129.09	0.09	0.016	0.001, 0.013	0.0009
IMV-Tensor-P	24.84	0.43, 15.57	0.61	161.98	0.11, 131.17	0.12	0.013	0.0008, 0.009	0.0005
IMV-Full	24.32	0.32, 15.47	0.02	162.14	0.10, 128.83	0.12	0.015	0.001, 0.011	0.002
IMV-Tensor	**24.12**	**0.03, 15.10**	**0.01**	**157.64**	**0.14, 128.16**	**0.13**	**0.007**	**0.0005, 0.006**	**0.0003**

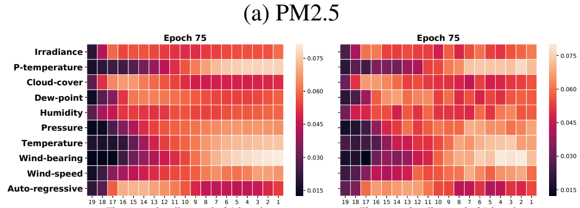
for IMV-LSTM ("Humid." represents humidity). As is suggested in (Nguyen et al., 2014; Höppe, 1993), humidity is correlated to the indoor temperature.

Temporal importance. Fig. 4 demonstrate the temporal importance values of each variable at the ending of the training. The lighter the color, the more the corresponding data contributes to the prediction.

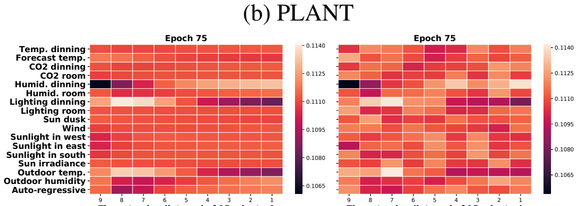
Specifically, in Fig. 4(a), short history of variables "Snow" and "Wind speed" contributes more to the prediction. PM2.5 itself has relatively long-term auto-correlation, i.e. around 5 hours. Fig. 4(b) shows that recent data of aforementioned important variables "Wind" and "Temperature" are highly used for prediction, while "Cloud-cover" is long-term correlated to the target, i.e. around 13 hours. In Fig. 4(c) temporal importance values are mostly uniform, though "Humid. dinning" has short correlation, "Outdoor temp." and "Lighting dinning" are relatively long-term correlated to the target.



(a) PM2.5



(b) PLANT



(c) SML

Figure 4: Variable-wise temporal importance interpretation. Left and right panels respectively correspond to IMV-Full and IMV-Tensor. Different patterns of decays in the temporal importance of variables are observed, which brings the additional interpretation of the dynamics of each variable. (Best viewed in color)

4.5. Variable Selection

In this group of experiments, we quantitatively evaluate the efficacy of variable importance through the lens of prediction tasks. We focus on IMV-LSTM family and RNN baselines, i.e. DUAL and RETAIN.

Specifically, for each approach, we first rank variables respectively according to the variable importance in IMV-LSTM, variable attention in DUAL and contribution coefficients in RETAIN. Meanwhile, we add one more group of baselines denoted by IMV-Full-P and IMV-Tensor-P. The label "-P" represents that the Pearson correlation is used to rank the variables with the highest (absolute) correlation values to the target and the selected data is fed to IMV-LSTM.

Then we rebuild datasets only consisting of top 50% ranked variables by respective methods, retrain each model with these new datasets and obtain the errors in Table 2.

Insights. Ideally, effective variable selection enables the corresponding retrained models to have comparable errors in comparison to their counterparts trained on full data in Table 1. IMV-Full and IMV-Tensor present comparable and even lower errors in Table 2, while DUAL and RETAIN have higher errors mostly. Pearson correlation measures linear relation. Selecting variables based on it neglects non-linear correlation and is not suitable for LSTM to attain the best performance. An additional advantage of variable selection is the training efficiency, e.g. training time of each epoch in IMV-Tensor is reduced from ∼16 to ∼11 sec.

5. Conclusion and Discussion

In this paper, we explore the internal structures of LSTMs for interpretable prediction on multi-variable time series. Based on the hidden state matrix, we present two realizations i.e. IMV-Full and IMV-Tensor, which enable to infer and quantify variable importance and variable-wise temporal importance w.r.t. the target. Extensive experiments provide insights into achieving superior prediction performance and importance interpretation for LSTM.

Regarding high order effect, e.g. variable interaction in data, it can be captured by adding additional rows into hidden state matrices and additional elements into importance vectors accordingly. This will be the future work.

Acknowledgements

The work has been funded by the EU Horizon 2020 SoBigData project under grant agreement No. 654024.

References

Ancona, M., Ceolini, E., Oztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

Arras, L., Montavon, G., Müller, K.-R., and Samek, W. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2014.

Bai, J. and Ng, S. Forecasting economic time series using targeted predictors. *Journal of Econometrics*, 146(2): 304–317, 2008.

Bishop, C. M. Mixture density networks. 1994.

Ceci, M., Corizzo, R., Fumarola, F., Malerba, D., and Rashkovska, A. Predictive modeling of pv energy production: How to set up the learning task for a better prediction? *IEEE Transactions on Industrial Informatics*, 13(3):956–966, 2017.

Che, Z., Purushotham, S., Khemani, R., and Liu, Y. Interpretable deep models for icu outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, pp. 371. American Medical Informatics Association, 2016.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *SIGKDD*, pp. 785–794. ACM, 2016.

Cheng, H., Tan, P.-N., Gao, J., and Scripps, J. Multistep-ahead time series prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 765–774. Springer, 2006.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Choi, E., Bahadori, M. T., Sun, J., Kulas, J., Schuetz, A., and Stewart, W. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pp. 3504–3512, 2016.

Choi, H., Cho, K., and Bengio, Y. Fine-grained attention mechanism for neural machine translation. *Neurocomputing*, 284:171–176, 2018.

Chu, L., Hu, X., Hu, J., Wang, L., and Pei, J. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1244–1253, New York, NY, USA, 2018. ACM.

Cinar, Y. G., Mirisaee, H., Goswami, P., Gaussier, E., Aït-Bachir, A., and Strijov, V. Position-based content attention for time series forecasting with sequence-to-sequence rnns. In *International Conference on Neural Information Processing*, pp. 533–544. Springer, 2017.

Do, K., Tran, T., and Venkatesh, S. Matrix-centric neural networks. *arXiv preprint arXiv:1703.01454*, 2017.

Feng, J., Williamson, B. D., Carone, M., and Simon, N. Nonparametric variable importance using an augmented neural network with multi-task learning. In *International Conference on Machine Learning*, pp. 1495–1504, 2018.

Foerster, J. N., Gilmer, J., Sohl-Dickstein, J., Chorowski, J., and Sussillo, D. Input switched affine networks: An rnn architecture designed for interpretability. In *International Conference on Machine Learning*, pp. 1136–1145, 2017.

Fox, I., Ang, L., Jaiswal, M., Pop-Busui, R., and Wiens, J. Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’18, pp. 1387–1395. ACM, 2018.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Ghazi, S. and Ip, K. The effect of weather conditions on the efficiency of pv panels in the southeast of uk. *Renewable Energy*, 69:50–59, 2014.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., and Funaya, K. Robust online time series prediction with recurrent neural networks. In *2016 IEEE DSAA*, pp. 816–825. IEEE, 2016.

Guo, T., Lin, T., and Lu, Y. An interpretable lstm neural network for autoregressive exogenous model. In *workshop track at International Conference on Learning Representations*, 2018.

He, Z., Gao, S., Xiao, L., Liu, D., He, H., and Barber, D. Wider and deeper, cheaper and faster: Tensorized lstms for sequence learning. In *Advances in Neural Information Processing Systems*, pp. 1–11, 2017.

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

Höppe, P. Indoor climate. *Experientia*, 49(9):775–779, 1993.

Hu, Z., Liu, W., Bian, J., Liu, X., and Liu, T.-Y. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 261–269. ACM, 2018.

Hyndman, R. J. and Athanasopoulos, G. *Forecasting: principles and practice*. OTexts, 2014.

Ke, N. R., Zolna, K., Sordoni, A., Lin, Z., Trischler, A., Bengio, Y., Pineau, J., Charlin, L., and Pal, C. Focused hierarchical rnns for conditional sequence processing. *arXiv preprint arXiv:1806.04342*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirchgässner, G., Wolters, J., and Hassler, U. *Introduction to modern time series analysis*. Springer Science & Business Media, 2012.

Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. A clockwork rnn. In *International Conference on Machine Learning*, pp. 1863–1871, 2014.

Kuchaiev, O. and Ginsburg, B. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*, 2017.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. *arXiv preprint arXiv:1703.07015*, 2017.

Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., Huang, H., and Chen, S. X. Assessing beijing's pm2. 5 pollution: severity, weather impact, apec and winter heating. In *Proc. R. Soc. A*, volume 471, pp. 20150257. The Royal Society, 2015.

Liaw, A., Wiener, M., et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Lin, T., Guo, T., and Aberer, K. Hybrid neural networks for learning the trend in time series. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 2273–2279, 2017.

Lipton, Z. C. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

Lipton, Z. C., Kale, D. C., Elkan, C., and Wetzell, R. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

Liu, Y., Niculescu-Mizil, A., Lozano, A. C., and Lu, Y. Learning temporal causal graphs for relational time-series analysis. In *ICML*, pp. 687–694, 2010.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

Mekhilef, S., Saidur, R., and Kamalisarvestani, M. Effect of dust, humidity and air velocity on efficiency of photovoltaic cells. *Renewable and sustainable energy reviews*, 16(5):2920–2925, 2012.

Montavon, G., Samek, W., and Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

Murdoch, W. J. and Szlam, A. Automatic rule extraction from long short term memory networks. *International Conference on Learning Representations*, 2017.

Murdoch, W. J., Liu, P. J., and Yu, B. Beyond word importance: Contextual decomposition to extract interactions from lstms. *arXiv preprint arXiv:1801.05453*, 2018.

Neil, D., Pfeiffer, M., and Liu, S.-C. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, pp. 3882–3890, 2016.

Nguyen, J. L., Schwartz, J., and Dockery, D. W. The relationship between indoor and outdoor temperature, apparent temperature, relative humidity, and absolute humidity. *Indoor air*, 24(1):103–112, 2014.

Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. P. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 442–450, 2015.

Patel, J., Shah, S., Thakkar, P., and Kotecha, K. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015.

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., and Cottrell, G. W. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 2627–2633. AAAI Press, 2017.

Radinsky, K., Svore, K., Dumais, S., Teevan, J., Bocharov, A., and Horvitz, E. Modeling and predicting behavioral dynamics on the web. In *WWW*, pp. 599–608. ACM, 2012.

Ribeiro, M. T., Singh, S., and Guestrin, C. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

Riemer, M., Vempaty, A., Calmon, F., Heath, F., Hull, R., and Khabiri, E. Correcting forecasts with multifactor neural attention. In *International Conference on Machine Learning*, pp. 3010–3019, 2016.

Scott, S. L. and Varian, H. R. Predicting the present with bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5 (1-2):4–23, 2014.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

Wang, J., Wang, Z., Li, J., and Wu, J. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery*, pp. 2437–2446, New York, NY, USA, 2018. ACM.

Xu, Y., Biswal, S., Deshpande, S. R., Maher, K. O., and Sun, J. Raim: Recurrent attentive and intensive model of multimodal patient monitoring data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2565–2573. ACM, 2018.

Zhang, L., Aggarwal, C., and Qi, G.-J. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141–2149. ACM, 2017.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.