A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction

Yao Qin\(^1\), Dongjin Song\(^2\), Haifeng Cheng\(^2\), Wei Cheng\(^2\), Guofei Jiang\(^2\), Garrison W. Cottrell\(^1\)

\(^1\)University of California, San Diego, \(^2\)NEC Laboratories America, Inc.
{yaq007, gary}@eng.ucsd.edu, {dsong, Haifeng, weicheng, gfj}@nec-labs.com

Abstract

The Nonlinear autoregressive exogenous (NARX) model, which predicts the current value of a time series based upon its previous values as well as the current and past values of multiple driving (exogenous) series, has been studied for decades. Despite the fact that various NARX models have been developed, few of them can capture the long-term temporal dependencies appropriately and select the relevant driving series to make predictions. In this paper, we propose a dual-stage attention-based recurrent neural network (DA-RNN) to address these two issues. In the first stage, we introduce an input attention mechanism to adaptively extract relevant driving series (a.k.a., input features) at each time step by referring to the previous encoder hidden state. In the second stage, we use a temporal attention mechanism to select relevant encoder hidden states across all time steps. With this dual-stage attention scheme, our model can not only make predictions effectively, but can also be easily interpreted. Thorough empirical studies based upon the SML 2010 dataset and the NASDAQ 100 Stock dataset demonstrate that the DA-RNN can outperform state-of-the-art methods for time series prediction.

1 Introduction

Time series prediction algorithms have been widely applied in many areas, e.g., financial market prediction [Wu et al., 2013], weather forecasting [Chakraborty et al., 2012], and complex dynamical system analysis [Liu and Hauskrecht, 2015]. Although the well-known autoregressive moving average (ARMA) model [Whittle, 1951] and its variants [ASTERIOTI and Hall, 2011; Brockwell and Davis, 2009] have shown their effectiveness for various real world applications, they cannot model nonlinear relationships and do not differentiate among the exogenous (driving) input terms. To address this issue, various nonlinear autoregressive exogenous (NARX) models [Lin et al., 1996; Gao and Er, 2005; Diaconescu, 2008; Yan et al., 2013] have been developed.

Typically, given the previous values of the target series, \( (y_1, y_2, \cdots, y_{t-1}) \) with \( y_{t-1} \in \mathbb{R} \), as well as the current and past values of \( n \) driving (exogenous) series, i.e., \( (x_1, x_2, \cdots, x_t) \) with \( x_t \in \mathbb{R}^n \), the NARX model aims to learn a nonlinear mapping to the current value of target series \( y_t \), i.e., \( \hat{y}_t = F(y_1, y_2, \cdots, y_{t-1}, x_1, x_2, \cdots, x_t) \), where \( F(\cdot) \) is the mapping function to learn.

Despite the fact that a substantial effort has been made for time series prediction via kernel methods [Chen et al., 2008], ensemble methods [Bouchachia and Bouchachia, 2008], and Gaussian processes [Frigola and Rasmussen, 2014], the drawback is that most of these approaches employ a predefined nonlinear form and may not be able to capture the true underlying nonlinear relationship appropriately. Recurrent neural networks (RNNs) [Elman, 1991], a type of deep neural network specially designed for sequence modeling, have received a great amount of attention due to their flexibility in capturing nonlinear relationships. In particular, RNNs have shown their success in NARX time series forecasting in recent years [Gao and Er, 2005; Diaconescu, 2008]. Traditional RNNs, however, suffer from the problem of vanishing gradients [Bengio et al., 1994] and thus have difficulty capturing long-term dependencies.

Recently, long short-term memory units (LSTM) [Hochreiter and Schmidhuber, 1997] and the gated recurrent unit (GRU) [Cho et al., 2014b] have overcome this limitation and achieved great success in various applications, e.g., neural machine translation [Bahdanau et al., 2014], speech recognition [Graves et al., 2013], and image processing [Kaparthy and Li, 2015]. Therefore, it is natural to consider state-of-the-art RNN methods, e.g., encoder-decoder networks [Cho et al., 2014b; Sutskever et al., 2014] and attention-based encoder-decoder networks [Bahdanau et al., 2014], for time series prediction.

Based upon LSTM or GRU units, encoder-decoder networks [Kalchbrenner and Blunsom, 2013; Cho et al., 2014a; Cho et al., 2014b; Sutskever et al., 2014] have become popular due to their success in machine translation. The key idea is to encode the source sentence as a fixed-length vector and use the decoder to generate a translation. One problem with encoder-decoder networks is that their performance will deteriorate rapidly as the length of input sequence increases [Cho et al., 2014a]. In time series analysis, this could be a concern since we usually expect to make predictions based upon

\(^{1}\)Most of this work was performed while the first author was an intern at NEC Labs America.
a relatively long segment of the target series as well as driving series. To resolve this issue, the attention-based encoder-decoder network [Bahdanau et al., 2014] employs an attention mechanism to select parts of hidden states across all the time steps. Recently, a hierarchical attention network [Yang et al., 2016], which uses two layers of attention mechanism to select relevant encoder hidden states across all the time steps, was also developed. Although attention-based encoder-decoder networks and hierarchical attention networks have shown their efficacy for machine translation, image captioning [Xu et al., 2015], and document classification, they may not be suitable for time series prediction. This is because when multiple driving (exogenous) series are available, the network cannot explicitly select relevant driving series to make predictions. In addition, they have mainly been used for classification, rather than time series prediction.

To address these aforementioned issues, and inspired by some theories of human attention [Hübner et al., 2010] that posit that human behavior is well-modeled by a two-stage attention mechanism, we propose a novel dual-stage attention-based recurrent neural network (DA-RNN) to perform time series prediction. In the first stage, we develop a new attention mechanism to adaptively extract the relevant driving series at each time step by referring to the previous encoder hidden state. In the second stage, a temporal attention mechanism is used to select relevant encoder hidden states across all time steps. These two attention models are well integrated within an LSTM-based recurrent neural network (RNN) and can be jointly trained using standard back propagation. In this way, the DA-RNN can adaptively select the most relevant input features as well as capture the long-term temporal dependencies of a time series appropriately. To justify the effectiveness of the DA-RNN, we compare it with state-of-the-art approaches using the SML 2010 dataset and the NASDAQ 100 Stock dataset with a large number of driving series. Extensive experiments not only demonstrate the effectiveness of the proposed approach, but also show that the DA-RNN is easy to interpret, and robust to noisy inputs.

2 Dual-Stage Attention-Based RNN

In this section, we first introduce the notation we use in this work and the problem we aim to study. Then, we present the motivation and details of the DA-RNN for time series prediction.

2.1 Notation and Problem Statement

Given $n$ driving series, i.e., $X = (x^1, x^2, \ldots, x^n)^\top \in \mathbb{R}^{n \times T}$, where $T$ is the length of window size, we use $x^k = (x^k_1, x^k_2, \ldots, x^k_T)^\top \in \mathbb{R}^T$ to represent a driving series of length $T$ and employ $x_t = (x^1_t, x^2_t, \ldots, x^n_t)^\top \in \mathbb{R}^n$ to denote a vector of $n$ exogenous (driving) input series at time $t$.

Typically, given the previous values of the target series, i.e., $(y_1, y_2, \ldots, y_{T-1})$ with $y_t \in \mathbb{R}$, as well as the current and past values of $n$ driving (exogenous) series, i.e., $(x_1, x_2, \ldots, x_T)$ with $x_t \in \mathbb{R}^n$, the NARX model aims to learn a nonlinear mapping to the current value of the target series $y_T$:

$$
\hat{y}_T = F(y_1, \ldots, y_{T-1}, x_1, \ldots, x_T).
$$

where $F(\cdot)$ is a nonlinear mapping function we aim to learn.

2.2 Model

Some theories of human attention [Hübner et al., 2010] argue that behavioral results are best modeled by a two-stage attention mechanism. The first stage selects the elementary stimulus features while the second stage uses categorical information to decode the stimulus. Inspired by these theories, we propose a novel dual-stage attention-based recurrent neural network (DA-RNN) for time series prediction. In the encoder, we introduce a novel input attention mechanism that can adaptively select the relevant driving series. In the decoder, a temporal attention mechanism is used to automatically select relevant encoder hidden states across all time steps. For the objective, a square loss is used. With these two attention mechanisms, the DA-RNN can adaptively select the most relevant input features and capture the long-term temporal dependencies of a time series. A graphical illustration of the proposed model is shown in Figure 1.
Encoder with Input Attention
The encoder is essentially an RNN that encodes the input sequences into a feature representation in machine translation [Cho et al., 2014b; Sutskever et al., 2014]. For time series prediction, given the input sequence \( \mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) \) with \( \mathbf{x}_i \in \mathbb{R}^n \), where \( n \) is the number of driving (exogenous) series, the encoder can be applied to learn a mapping from \( \mathbf{x}_i \) to \( \mathbf{h}_t \) (at time step \( t \)) with

\[
\mathbf{h}_t = f_1(\mathbf{h}_{t-1}, \mathbf{x}_t),
\]

where \( \mathbf{h}_t \in \mathbb{R}^m \) is the hidden state of the encoder at time \( t \), \( m \) is the size of the hidden state, and \( f_1 \) is a non-linear activation function that could be an LSTM [Hochreiter and Schmidhuber, 1997] or gated recurrent unit (GRU) [Cho et al., 2014b]. In this paper, we use an LSTM unit as \( f_1 \) to capture long-term dependencies. Each LSTM unit has a memory cell with the state \( \mathbf{s}_t \) at time \( t \). Access to the memory cell will be controlled by three sigmoid gates: forget gate \( \mathbf{f}_t \), input gate \( \mathbf{i}_t \), and output gate \( \mathbf{o}_t \). The update of an LSTM unit can be summarized as follows:

\[
\mathbf{f}_t = \sigma(W_f [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_f)
\]

\[
\mathbf{i}_t = \sigma(W_i [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_i)
\]

\[
\mathbf{o}_t = \sigma(W_o [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_o)
\]

\[
\mathbf{s}_t = \mathbf{f}_t \odot \mathbf{s}_{t-1} + \mathbf{i}_t \odot \tanh(W_t [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_s)
\]

\[
\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{s}_t)
\]

where \( [\mathbf{h}_{t-1}; \mathbf{x}_t] \in \mathbb{R}^{m+n} \) is a concatenation of the previous hidden state \( \mathbf{h}_{t-1} \) and the current input \( \mathbf{x}_t \). \( W_f, W_i, W_o, W_t \in \mathbb{R}^{m \times (m+n)} \), and \( \mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_s \in \mathbb{R}^m \) are parameters to learn. \( \sigma \) and \( \odot \) are a logistic sigmoid function and an element-wise multiplication, respectively. The key reason for using an LSTM unit is that the cell state sums activities over time, which can overcome the problem of vanishing gradients and better capture long-term dependencies of time series.

Inspired by the theory that the human attention system can select elementary stimulus features in the early stages of processing [Hubner et al., 2010], we propose an input attention-based encoder that can adaptively select the relevant driving series, which is of practical meaning in time series prediction. Given the \( k \)-th input driving series \( \mathbf{x}_k^T = (x_{k1}^T, x_{k2}^T, \ldots, x_{kT}^T) \in \mathbb{R}^T \), we can construct an input attention mechanism via a deterministic attention model, i.e., a multilayer perceptron, by referring to the previous hidden state \( \mathbf{h}_{t-1} \) and the cell state \( \mathbf{s}_{t-1} \) in the encoder LSTM unit with

\[
e^{k_t} = \mathbf{v}_c^T \tanh(W_c [\mathbf{h}_{t-1}; \mathbf{s}_{t-1}] + \mathbf{U}_c \mathbf{x}_k^T)
\]

and

\[
\alpha_t^k = \frac{\exp(e^{k_t})}{\sum_{k=1}^K \exp(e^{k_t})}
\]

where \( \mathbf{v}_c \in \mathbb{R}^T, W_c \in \mathbb{R}^{T \times 2m} \), and \( \mathbf{U}_c \in \mathbb{R}^{T \times T} \) are parameters to learn. We omit the bias terms in Eqn. (16) to be succinct. \( \alpha_t^k \) is the attention weight measuring the importance of the \( k \)-th input feature (driving series) at time \( t \). A softmax function is applied to \( e^{k_t} \) to ensure all the attention weights sum to 1. The input attention mechanism is a feed forward network that can be jointly trained with other components of the RNN. With these attention weights, we can adaptively extract the driving series with

\[
\hat{\mathbf{x}}_t = (\alpha_1^t x_1^t, \alpha_2^t x_2^t, \ldots, \alpha_n^t x_n^t)^T.
\]

Then the hidden state at time \( t \) can be updated as:

\[
\mathbf{h}_t = f_1(\mathbf{h}_{t-1}, \hat{\mathbf{x}}_t),
\]

where \( f_1 \) is an LSTM unit that can be computed according to Eqn. (3) - (7) with \( \mathbf{x}_t \) replaced by the newly computed \( \hat{\mathbf{x}}_t \). With the proposed input attention mechanism, the encoder can selectively focus on certain driving series rather than treating all the input driving series equally.

Decoder with Temporal Attention
To predict the output \( \mathbf{y}_T \), we use another LSTM-based recurrent neural network to decode the encoded input information. However, as suggested by Cho et al. [2014a], the performance of the encoder-decoder network can deteriorate rapidly as the length of the input sequence increases. Therefore, following the encoder with input attention, a temporal attention mechanism is used in the decoder to adaptively select relevant encoder hidden states across all time steps. Specifically, the attention weight of each encoder hidden state at time \( t \) is calculated based upon the previous decoder hidden state \( \mathbf{d}_{t-1} \in \mathbb{R}^p \) and the cell state of the LSTM unit \( \mathbf{s}_{t-1}^e \in \mathbb{R}^p \) with

\[
l'_t = \mathbf{v}_d^T \tanh(\mathbf{W}_d [\mathbf{d}_{t-1}; \mathbf{s}_{t-1}^e] + \mathbf{U}_d \mathbf{h}_t), \quad 1 \leq i \leq T
\]

and

\[
\beta_i^t = \frac{\exp(l'_i)}{\sum_{j=1}^T \exp(l'_j)}
\]

where \( [\mathbf{d}_{t-1}; \mathbf{s}_{t-1}^e] \in \mathbb{R}^{2p} \) is a concatenation of the previous hidden state and cell state of the LSTM unit. \( \mathbf{v}_d \in \mathbb{R}^m, \mathbf{W}_d \in \mathbb{R}^{m \times 2p} \), and \( \mathbf{U}_d \in \mathbb{R}^{m \times m} \) are parameters to learn. The bias terms here have been omitted for clarity. The attention weight \( \beta_i^t \) represents the importance of the \( i \)-th encoder hidden state for the prediction. Since each encoder hidden state \( \mathbf{h}_i \) is mapped to a temporal component of the input, the attention mechanism computes the context vector \( \mathbf{c}_t \) as a weighted sum of all the encoder hidden states \( \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T\} \),

\[
\mathbf{c}_t = \sum_{i=1}^T \beta_i^t \mathbf{h}_i.
\]

Note that the context vector \( \mathbf{c}_t \) is distinct at each time step.

Once we get the weighted summed context vectors, we can combine them with the given target series \( \{y_1, y_2, \ldots, y_{T-1}\} \):

\[
\tilde{y}_{t-1} = \mathbf{w}^T [y_{t-1}; \mathbf{c}_{t-1}] + \tilde{b},
\]

where \( [y_{t-1}; \mathbf{c}_{t-1}] \in \mathbb{R}^{m+1} \) is a concatenation of the decoder input \( y_{t-1} \) and the computed context vector \( \mathbf{c}_{t-1} \). Parameters \( \mathbf{w} \in \mathbb{R}^{m+1} \) and \( \tilde{b} \in \mathbb{R} \) map the concatenation to the size the decoder input. The newly computed \( \tilde{y}_{t-1} \) can be used for the update of the decoder hidden state at time \( t \):

\[
\mathbf{d}_t = f_2(\mathbf{d}_{t-1}, \tilde{y}_{t-1}).
\]

We choose the nonlinear function \( f_2 \) as an LSTM unit [Hochreiter and Schmidhuber, 1997], which has been widely used in modeling long-term dependencies. Then \( \mathbf{d}_t \) can be updated as:

\[
\mathbf{f}'_t = \sigma(W'_f [\mathbf{d}_{t-1}; \tilde{y}_{t-1}] + \mathbf{b}'_f)
\]
\[
\begin{align*}
\mathbf{d}_t &= \sigma(\mathbf{s}_t') \odot \tanh(\mathbf{s}_t) \\
\mathbf{s}_t' &= \mathbf{f}_t' \odot \mathbf{s}_{t-1}' + \mathbf{i}_t' \odot \tanh(\mathbf{W}_s \mathbf{d}_{t-1} + \cdot \mathbf{b}_s) \\
\mathbf{o}_t' &= \sigma(\mathbf{W}_o \mathbf{d}_{t-1} + \cdot \mathbf{b}_o) \\
i_t' &= \sigma(\mathbf{W}_i \mathbf{d}_{t-1} + \cdot \mathbf{b}_i)
\end{align*}
\]

where \(\mathbf{d}_{t-1}; \mathbf{y}_{t-1} \in \mathbb{R}^{p+1}\) is a concatenation of the previous hidden state \(\mathbf{d}_{t-1}\) and the decoder input \(\mathbf{y}_{t-1}\). \(\mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_s, \mathbf{W}_i\) are parameters to learn. \(\sigma\) and \(\odot\) are a logistic sigmoid function and an element-wise multiplication, respectively.

For NARX modeling, we aim to use the DA-RNN to approximate the function \(F\) so as to obtain an estimate of the current output \(\hat{y}_T\) with the observation of all inputs as well as previous outputs. Specifically, \(\hat{y}_T\) can be obtained with

\[
\begin{align*}
\hat{y}_T &= F(y_1, \cdots, y_{T-1}, \mathbf{x}_1, \cdots, \mathbf{x}_T) \\
&= \mathbf{v}_y^\top(\mathbf{W}_y[\mathbf{d}_T; \mathbf{c}_T] + \mathbf{b}_w) + b_v,
\end{align*}
\]

where \([\mathbf{d}_T; \mathbf{c}_T] \in \mathbb{R}^{p+m}\) is a concatenation of the decoder hidden state and the context vector. The parameters \(\mathbf{W}_y \in \mathbb{R}^{p \times (p+m)}\) and \(\mathbf{b}_w \in \mathbb{R}^p\) map the concatenation to the size of the decoder hidden states. The linear function with weights \(\mathbf{v}_y \in \mathbb{R}^p\) and bias \(b_v \in \mathbb{R}\) produces the final prediction result.

### Training Procedure

We use minibatch stochastic gradient descent (SGD) together with the Adam optimizer [Kingma and Ba, 2014] to train the model. The size of the minibatch is 128. The learning rate starts from 0.001 and is reduced by 10\% after each 10000 iterations. The proposed DA-RNN is smooth and differentiable, so the parameters can be learned by standard back propagation with mean squared error as the objective function:

\[
\mathcal{O}(y_T, \hat{y}_T) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_T - y_T)^2,
\]

where \(N\) is the number of training samples. We implemented the DA-RNN in the Tensorflow framework [Abadi et al., 2015].

### 3 Experiments

In this section, we first describe two datasets for empirical studies. Then, we introduce the parameter settings of DA-RNN and the evaluation metrics. Finally, we compare the proposed DA-RNN against four different baseline methods, interpret the input attention as well as the temporal attention of DA-RNN, and study its parameter sensitivity.

#### 3.1 Datasets and Setup

To test the performance of different methods for time series prediction, we use two different datasets as shown in Table 1:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>driving series</th>
<th>train</th>
<th>valid</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>SML 2010</td>
<td>16</td>
<td>3,200</td>
<td>400</td>
<td>537</td>
</tr>
<tr>
<td>NASDAQ 100 Stock</td>
<td>81</td>
<td>35,100</td>
<td>1,100</td>
<td>2,730</td>
</tr>
</tbody>
</table>

In the NASDAQ 100 Stock dataset[^1] we collected the stock prices of 81 major corporations under NASDAQ 100, which are used as the driving time series. The index value of the NASDAQ 100 is used as the target series. The frequency of the data collection is minute-by-minute. This data covers the period from July 26, 2016 to December 22, 2016, 105 days in total. Each day contains 390 data points from the opening to closing of the market except that there are 210 data points on November 25 and 180 data points on December 22. In our experiments, we use the first 35,100 data points as the training set and the following 2,730 data points as the validation set. The last 2,730 data points are used as the test set. This dataset is publicly available and will be continuously enlarged to aid the research in this direction.

#### 3.2 Parameter Settings and Evaluation Metrics

There are three parameters in the DA-RNN, i.e., the number of time steps in the window \(T\), the size of hidden states for the encoder \(m\), and the size of hidden states for the decoder \(p\). To determine the window size \(T\), we conducted a grid search over \(T \in \{3, 5, 10, 15, 25\}\). The one (\(T = 10\)) that achieves the best performance over validation set is used for test. For the size of hidden states for encoder \((m)\) and decoder \((p)\), we set \(m = p\) for simplicity and conduct grid search over \(m = p \in \{16, 32, 64, 128, 256\}\). Those two (\(i.e., m = p = 64, 128\)) that achieve the best performance over

[^1]: [http://cseweb.ucsd.edu/~yaq007/NASDAQ100_stock_data.html](http://cseweb.ucsd.edu/~yaq007/NASDAQ100_stock_data.html)
The validation set are used for evaluation. For all the RNN based approaches (i.e., NARX RNN, Encoder-Decoder, Attention RNN, Input-Att-RNN and DA-RNN), we train them 10 times and report their average performance and standard deviations for comparison.

To measure the effectiveness of various methods for time series prediction, we consider three different evaluation metrics. Among them, root mean squared error (RMSE) [Plautowski et al., 1996] and mean absolute error (MAE) are two scale-dependent measures, and mean absolute percentage error (MAPE) is a scale-dependent measure. Specifically, assuming $y_i$ is the target at time $t$ and $\hat{y}_i$ is the predicted value at time $t$, RMSE is defined as $\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$ and MAE is denoted as $\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$. When comparing the prediction performance across different datasets, mean absolute percentage error is popular because it measures the prediction deviation proportion in terms of the true values, i.e., $\text{MAPE} = \frac{1}{N} \sum_{i=1}^{N} |\frac{y_i - \hat{y}_i}{y_i}| \times 100\%$.

### 3.3 Results-I: Time Series Prediction

To demonstrate the effectiveness of the DA-RNN, we compare it against 4 baseline methods. Among them, the autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model [Asteriou and Hall, 2011]. NARX recurrent neural network (NARX RNN) is a classic method to address time series prediction [Diaconescu, 2008]. The encoder-decoder network (Encoder-Decoder) [Cho et al., 2014b] and attention-based encoder-decoder network (Attention RNN) [Bahdanau et al., 2014] were originally used for machine translation tasks, in which each time step of the decoder output should be used to produce a probability distribution over the translated word codebook. To perform time series prediction, we modify these two approaches by changing the output to be a single scalar value, and use a squared loss as the objective function (as we did for the DA-RNN). The input to these networks is no longer words or word representations, but the $n$ scalar driving series of length $T$. Additionally, the decoder has the additional input of the previous values of the target series as the given information.

### Table 2: Time series prediction results over the SML 2010 Dataset and NASDAQ 100 Stock Dataset (best performance displayed in boldface). The size of encoder hidden states $m$ and decoder hidden states $p$ are set as $m = p = 64$ and 128.

<table>
<thead>
<tr>
<th>Models</th>
<th>SML 2010 Dataset</th>
<th>NASDAQ 100 Stock Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE (× 10⁻²%)</td>
<td>MAPE (× 10⁻²%)</td>
</tr>
<tr>
<td>ARIMA [2011]</td>
<td>1.95</td>
<td>0.41</td>
</tr>
<tr>
<td>NARX RNN [2008]</td>
<td>1.79±0.07</td>
<td>8.64±0.29</td>
</tr>
<tr>
<td>Encoder-Decoder (64) [2014b]</td>
<td>2.59±0.07</td>
<td>12.1±0.34</td>
</tr>
<tr>
<td>Encoder-Decoder (128) [2014b]</td>
<td>1.91±0.02</td>
<td>9.00±0.10</td>
</tr>
<tr>
<td>Attention RNN (64) [2014]</td>
<td>1.78±0.03</td>
<td>8.46±0.09</td>
</tr>
<tr>
<td>Attention RNN (128) [2014]</td>
<td>1.77±0.02</td>
<td>8.45±0.09</td>
</tr>
<tr>
<td>Input-Att-RNN (64)</td>
<td>1.88±0.04</td>
<td>8.89±0.19</td>
</tr>
<tr>
<td>Input-Att-RNN (128)</td>
<td>1.70±0.03</td>
<td>8.09±0.15</td>
</tr>
<tr>
<td>DA-RNN (64)</td>
<td>1.53±0.01</td>
<td>7.31±0.05</td>
</tr>
<tr>
<td>DA-RNN (128)</td>
<td>1.50±0.01</td>
<td>7.14±0.07</td>
</tr>
</tbody>
</table>

![Figure 3](image-url) Figure 3: Plot of the input attention weights for DA-RNN from a single encoder time step. The first 81 weights are on 81 original driving series and the last 81 weights are on 81 noisy driving series. (left) Input attention weights on NASDAQ100 training set. (right) Input attention weights on NASDAQ100 test set.

Furthermore, we show the effectiveness of DA-RNN via step-by-step justification. Specifically, we compare dual-stage attention-based recurrent neural network (DA-RNN) against the setting that only employs its input attention mechanism (Input-Att-RNN). For all RNN-based methods, the encoder takes $n$ driving series of length $T$ as the input and the decoder takes the previous values of the target series as the given information for fair comparison. The time series prediction results of DA-RNN and baseline methods over the two datasets are shown in Table 2.

In Table 2, we observe that the RMSE of ARIMA is generally worse than RNN based approaches. This is because ARIMA only considers the target series ($y_1, \cdots, y_{t-1}$) and ignores the driving series ($x_1, \cdots, x_t$). For RNN based approaches, the performance of NARX RNN and Encoder-Decoder are comparable. Attention RNN generally outperforms Encoder-Decoder since it is capable to select relevant hidden states across all the time steps in the encoder. Within DA-RNN, the input attention RNN (Input-Att-RNN) consistently outperforms Encoder-Decoder as well as Attention RNN. This suggests that adaptively extracting driving series can provide more reliable input features to make accurate predictions.

With integration of the input attention mechanism as well as temporal attention mechanism, our DA-
RNN achieves the best MAE, MAPE, and RMSE across two datasets since it not only uses an input attention mechanism to extract relevant driving series, but also employs a temporal attention mechanism to select relevant hidden features across all time steps.

For visual comparison, we show the prediction result of Encoder-Decoder \((m = p = 128)\), Attention RNN \((m = p = 128)\) and DA-RNN \((m = p = 64)\) over the NASDAQ 100 Stock dataset in Figure 4. We observe that DA-RNN generally fits the ground truth much better than Encoder-Decoder and Attention RNN.

### 3.4 Results-II: Interpretation

To study the effectiveness of the input attention mechanism within DA-RNN, we test it with noisy driving (exogenous) series as the input. Specifically, within NASDAQ 100 Stock dataset, we generate 81 additional noisy driving series by randomly permuting the original 81 driving series. Then, we put these 81 noisy driving series together with the 81 original driving series as the input and test the effectiveness of DA-RNN. When the length of time steps \(T\) is 10 and the size of hidden states is \(m = p = 128\), DA-RNN achieves \(\text{MAE}: 0.28 \pm 0.007\), \(\text{MAPE}: (0.56 \pm 0.01) \times 10^{-2}\) and \(\text{RMSE}: 0.42 \pm 0.009\), which are comparable to its performance in Table 2. This indicates that DA-RNN is robust to noisy inputs.

To further investigate the input attention mechanism, we plot the input attention weights of DA-RNN for the 162 input driving series (the first 81 are original and the last 81 are noisy) in Figure 5. The plotted attention weights in Figure 5 are taken from a single encoder time step and similar patterns can also be observed for other time steps. We find that the input attention mechanism can automatically assign larger weights for the 81 original driving series and smaller weights for the 81 noisy driving series in an online fashion using the activation of the input attention network to scale these weights. This demonstrates that input attention mechanism can aid DA-RNN to select relevant input driving series and suppress noisy input driving series.

To investigate the effectiveness of the temporal attention mechanism within DA-RNN, we compare DA-RNN to Input-Attn-RNN when the length of time steps \(T\) varies from 3, 5, 10, 15, to 25. The detailed results over two datasets are shown in Figure 4. We observe that when \(T\) is relatively large, DA-RNN can significantly outperform Input-Attn-RNN. This suggests that temporal attention mechanism can capture long-term dependencies by selecting relevant encoder hidden states across all the time steps.

### 3.5 Results-III: Parameter Sensitivity

We study the sensitivity of DA-RNN with respect to its parameters, \(i.e., \text{the length of time steps } T \text{ and the size of hidden states for encoder } m \text{ (decoder } p)\). When we vary \(T\) or \(m (p)\), we keep the others fixed. By setting \(m = p = 128\), we plot the RMSE versus different lengths of time steps in the window \(T\) in Figure 4. It is easily observed that the performance of DA-RNN and Input-Attn-RNN will be worse when the length of time steps is too short or too long while DA-RNN is relatively more robust than Input-Attn-RNN. By setting \(T = 10\), we also plot the RMSE versus different sizes of hidden states for encoder and decoder \((m = p \in \{16, 32, 64, 128, 256\})\) in Figure 5. We notice that DA-RNN usually achieves the best performance when \(m = p = 64\) or 128. Moreover, we can also conclude that DA-RNN is more robust to parameters than Input-Attn-RNN.

### 4 Conclusion

In this paper, we proposed a novel dual-stage attention-based recurrent neural network (DA-RNN), which consists of an encoder with an input attention mechanism and a decoder with a temporal attention mechanism. The newly introduced input attention mechanism can adaptively select the relevant driving series. The temporal attention mechanism can naturally capture the long-range temporal information of the encoded inputs. Based upon these two attention mechanisms, the DA-RNN can not only adaptively select the most relevant input features, but can also capture the long-term temporal dependencies of a time series appropriately. Extensive experiments on the SML 2010 dataset and the NASDAQ 100 Stock dataset demonstrated that our proposed DA-RNN can outperform state-of-the-art methods for time series prediction.

The proposed dual-stage attention-based recurrent neural network (DA-RNN) not only can be used for time series prediction, but also has the potential to serve as a general feature learning tool in computer vision tasks [Pu et al., 2016; Qin et al., 2015]. In the future, we are going to employ DA-RNN to perform ranking and binary coding [Song et al., 2015; Song et al., 2016].

### 5 Acknowledgements

GWC is supported in part by NSF cooperative agreement SMA 1041755 to the Temporal Dynamics of Learning Center, and a gift from Hewlett Packard. GWC and YQ were also partially supported by Guangzhou Science and Technology Planning Project (Grant No. 201704030051).
References


