RNN and its application in time series prediction

Weimeng Luo

Columbia University

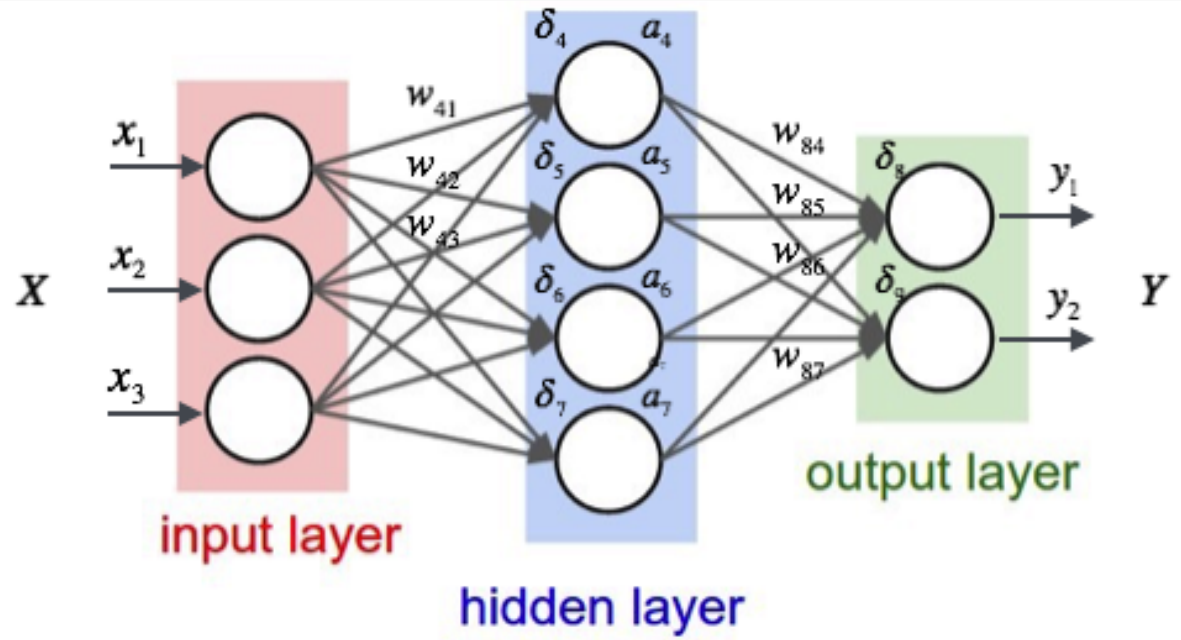
May 13, 2018

Agenda

- RNN basics
- RNN applications
- RNN experiments
- A novel approach for time series prediction

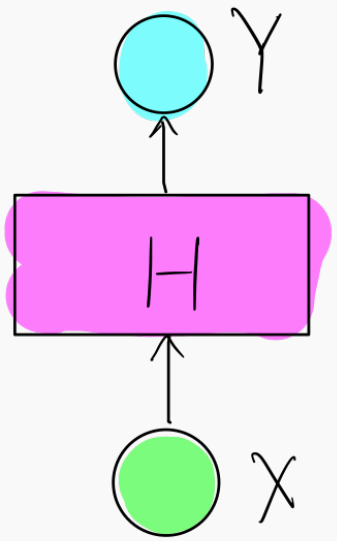
Neural network (Feedforward)

Problem formulation: $X \in \mathbb{R}^{n \times x}, Y \in \mathbb{R}^{n \times y}$



$$H = \phi_1(XW_{xh} + b_n)$$
$$Y = \phi_2(HW_{hy} + b_y)$$
$$Y = f(X)$$

Neural network



$Y_t = f(X_t)$

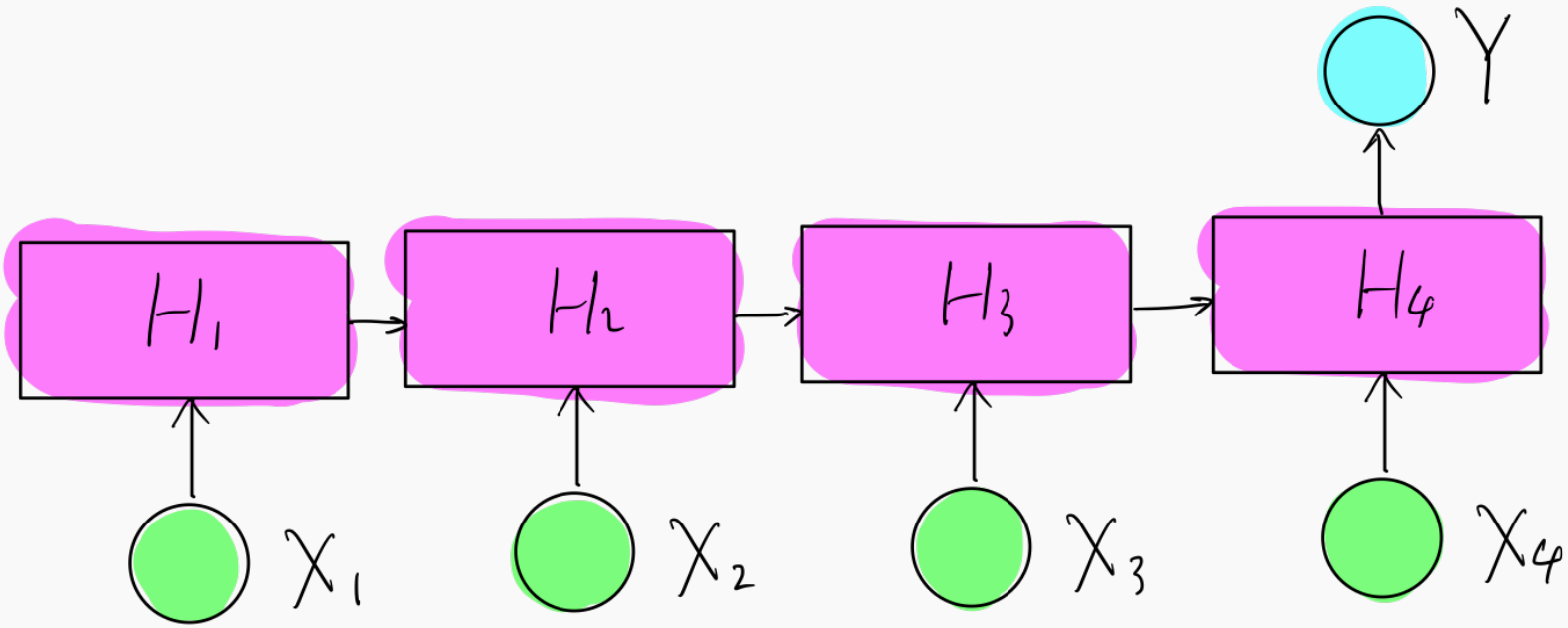
What about X is sequence model? Like $X_1, X_2, \ldots, X_t$

Example

- Language model: I love Tiananmen Square in Beijing
- Time series: hourly temperature $x_1, x_2, \ldots, x_t$

$Y_t = f(X_1, X_2, \ldots, X_t)$?

RNN (Recurrent Neural Network)



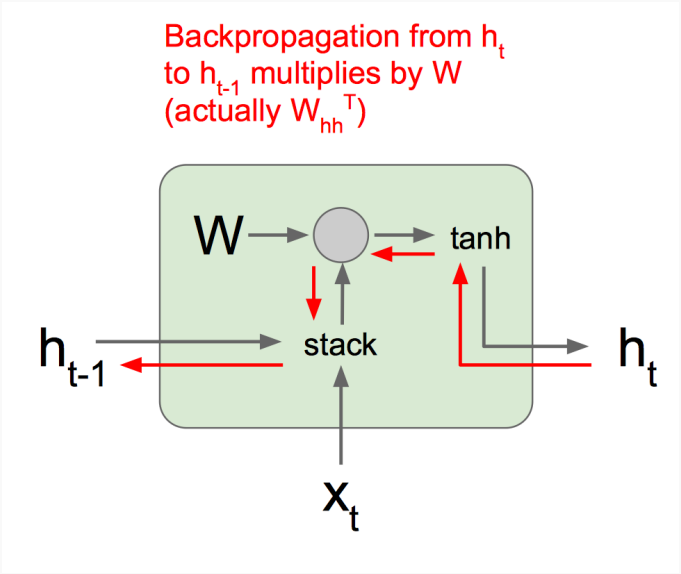
What about X is sequence data? Like $X_1, X_2, \ldots, X_t$

$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_n)$

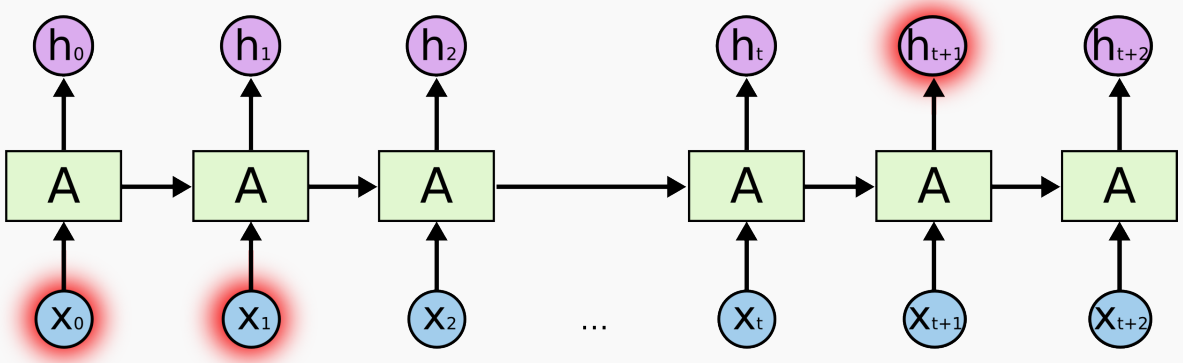
ϕ is often tanh

Drawbacks of RNN

- Gradient vanishing or exploding during backpropagation



- The problem of long-term dependencies



LSTM (Long Short-Term Memory) units

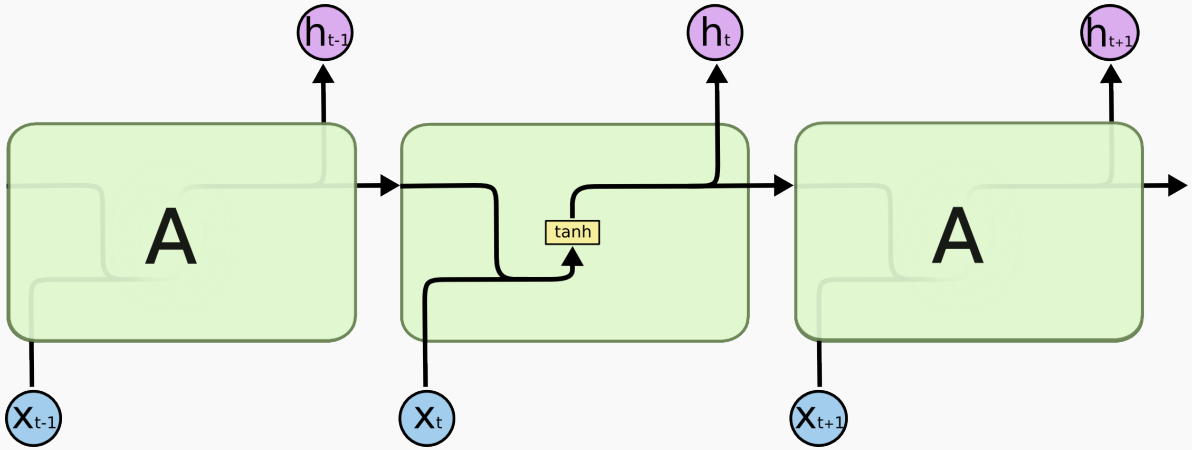


Figure: Vanilla RNN

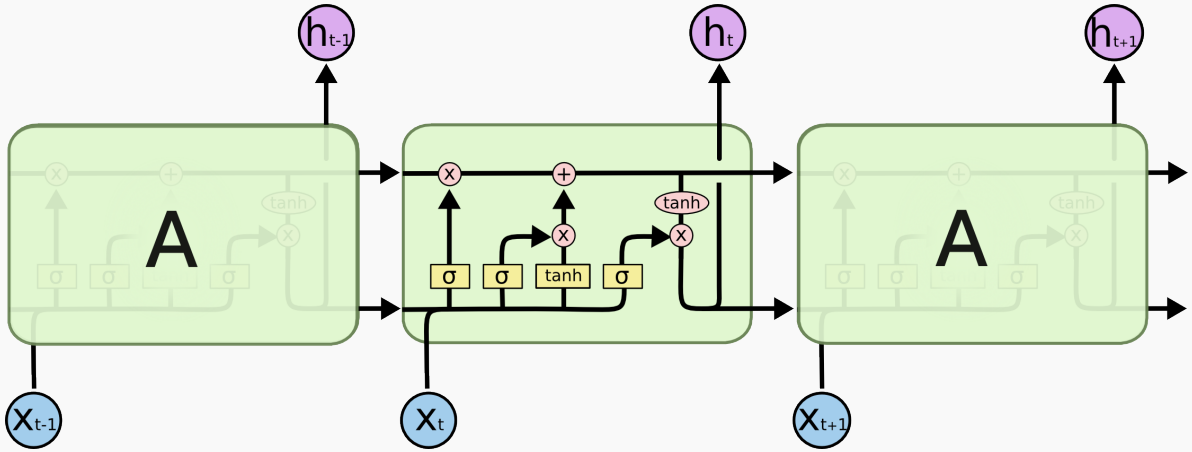


Figure: RNN with LSTM units

LSTM (Long Short-Term Memory) units

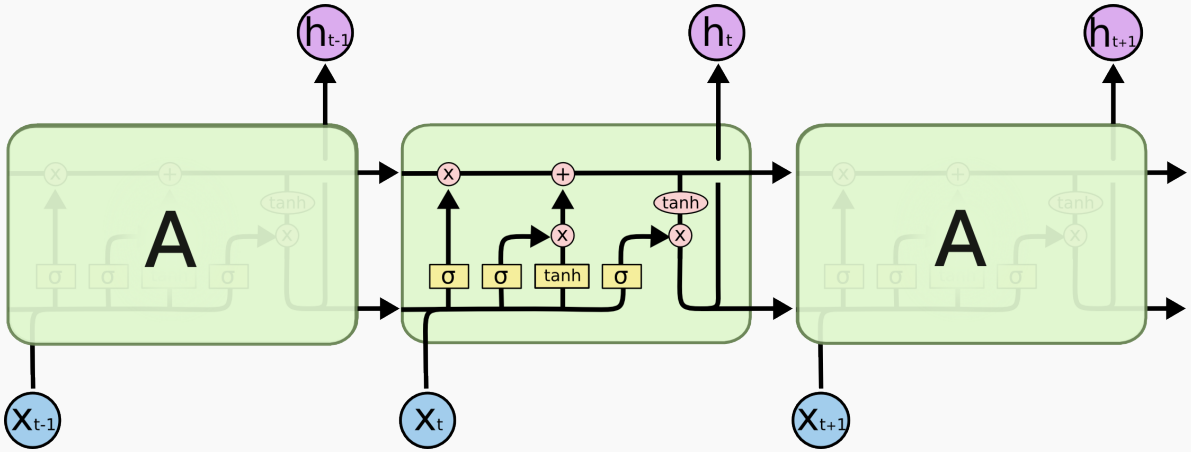


Figure: RNN with LSTM units

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)$$

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)$$

$$\tilde{C}_t = tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c)$$

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$

$$H_t = O_t \odot tanh(C_t)$$

Applications of RNN

- Text generation
- Image caption
- Speech recognition
- Machine translation
- Text summarization

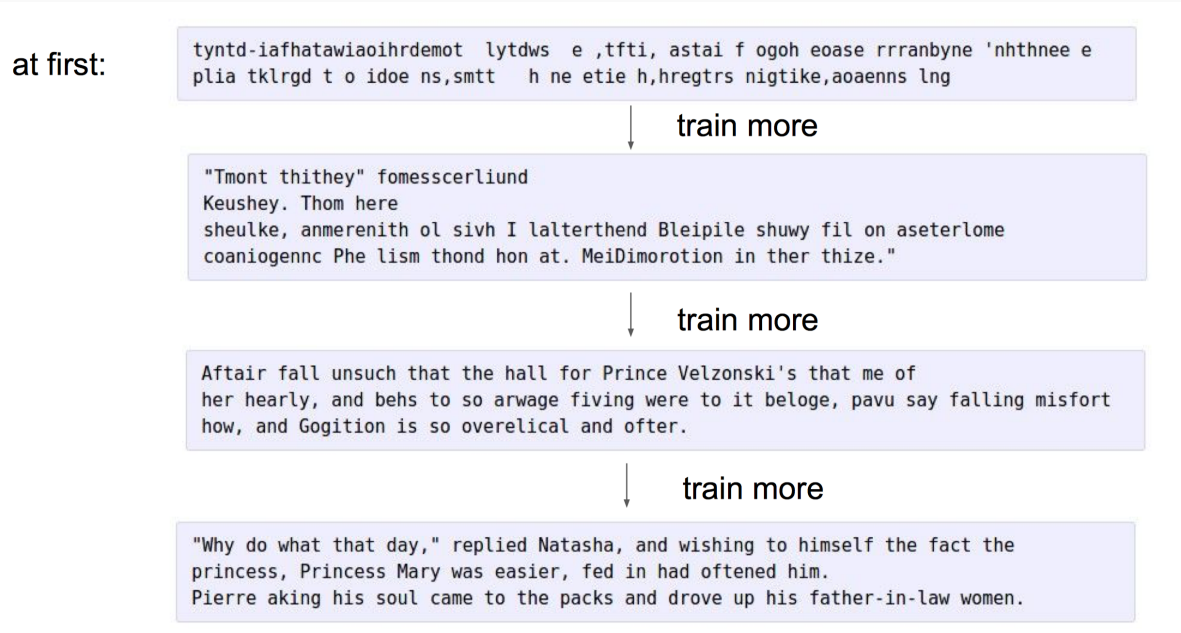


Figure: Text generation

Applications of RNN

- Text generation
- Image caption
- Speech recognition
- Machine translation
- Text summarization

Figure: Image caption

Problem Setup

Air quality prediction in Beijing

	pm25	t2m	d2m	u10	v10	sp	t500	t850	t1000	q500	...	q1000	u500	u850
2014-01-01 08:00:00	51.0	270.487368	258.980250	2.507680	-0.658156	97997.728951	248.554676	270.020533	276.617789	0.000131	...	0.001265	23.046411	10.548611
2014-01-01 14:00:00	23.0	283.838611	258.241520	3.154037	-0.050984	98048.881796	249.264215	270.571515	282.831868	0.000194	...	0.001557	23.175371	8.060430
2014-01-01 20:00:00	66.0	273.009328	261.723400	1.842218	-1.248453	98354.616140	248.722481	271.406578	282.505506	0.000195	...	0.001777	23.576161	6.588762
2014-01-02 02:00:00	127.0	270.101179	262.794066	0.369782	-1.550008	98568.984998	248.943138	271.072288	278.768867	0.000185	...	0.001637	18.130736	5.157552
2014-01-02 08:00:00	119.0	268.554291	262.665035	0.382165	-1.082520	98565.141143	248.187354	271.501711	275.817080	0.000187	...	0.001611	17.444210	7.410570

- Target series: PM25 data
  Driving series: Meteorologic data (Temperature, Specific humidity, Wind speed, Pressure)
  From 01/01/2014 to 12/31/2017, time step: 6 hours

Parameter Settings

- X: The time series of data in the past(pm25, meteorology).
  Y: The time series of pm25 data in the following days
- Lookback steps: 10
  Predicting steps: 3
- Training set: 4380, Test set: 1096
- Hidden size: 50
  Use grid search to search for the optimal hidden size
- Minibatch stochastic gradient descent (SGD):
  batch size 72
- Optimizer: Adam
- Learning rate: 0.001
- Train 10 times to calculate average

Loss function?

- $\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_t^i - \hat{y}_t^i|$
- $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_t^i - \hat{y}_t^i)^2$

	MAE			MSE		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
1	21.00 ± 0.91	35.67 ± 1.02	0.57 ± 0.24	23.40 ± 2.52	37.05 ± 2.14	1.21 ± 0.69
2	27.49 ± 1.84	43.96 ± 2.11	0.65 ± 0.17	29.70 ± 3.18	45.13 ± 2.58	0.84 ± 0.30
3	30.48 ± 2.20	48.16 ± 2.43	0.79 ± 0.32	34.07 ± 3.76	50.11 ± 3.20	0.90 ± 0.48

Table: Comparison between different loss function

MAE performs better than MSE!

Result

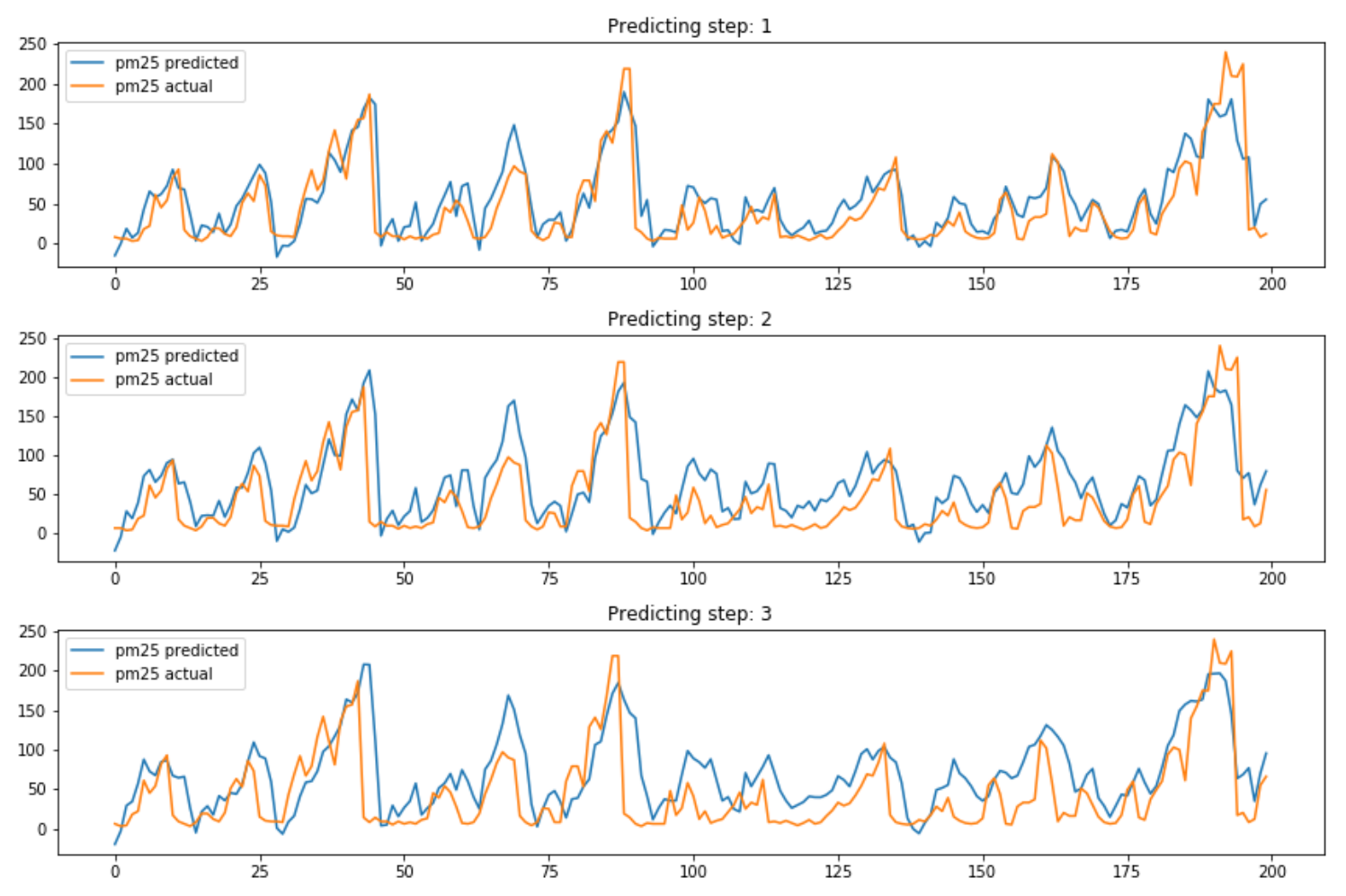


Figure: LSTM

Result

Evaluation metrics

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_t^i - \hat{y}_t^i|$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_t^i - \hat{y}_t^i)^2}$$

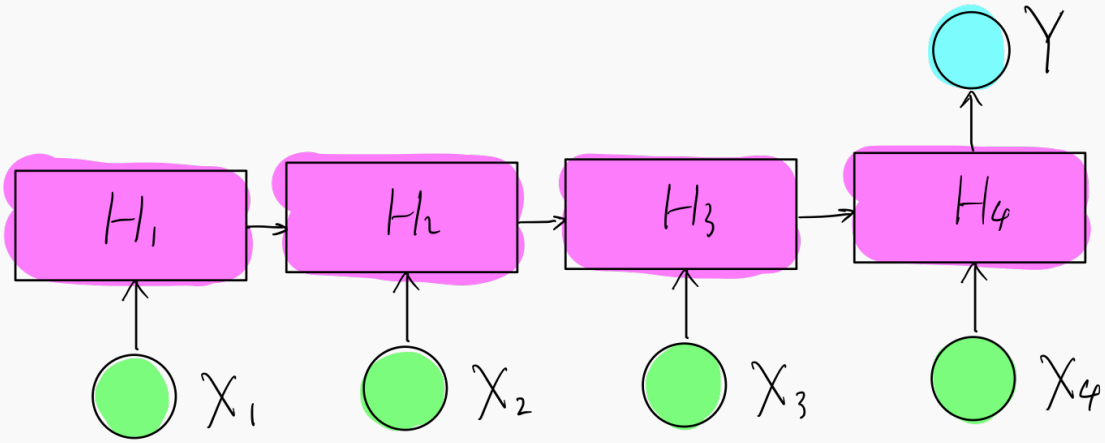
$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_t^i - \hat{y}_t^i}{y_t^i}\right|$$

	LSTM			ARIMA		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
1	21.00 ± 0.91	35.67 ± 1.02	0.57 ± 0.24	21.73	38.04	0.50
2	27.49 ± 1.84	43.96 ± 2.11	0.65 ± 0.17	29.85	47.53	0.67
3	30.48 ± 2.20	48.16 ± 2.43	0.79 ± 0.32	34.80	54.21	0.81

Table: Comparison between RNN and ARIMA

Seq2Seq

RNN: N vs 1



How to model N vs M?

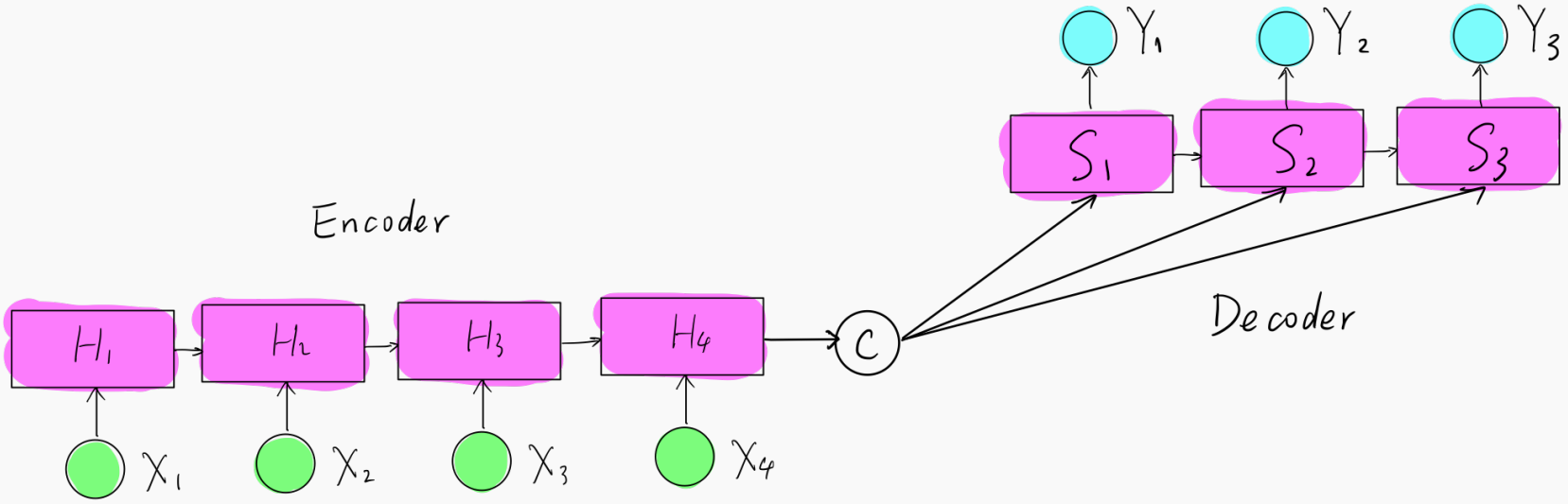
Example: language translation

English: They are watching.

French: Ils regardent.

Seq2Seq

How to model N vs M?

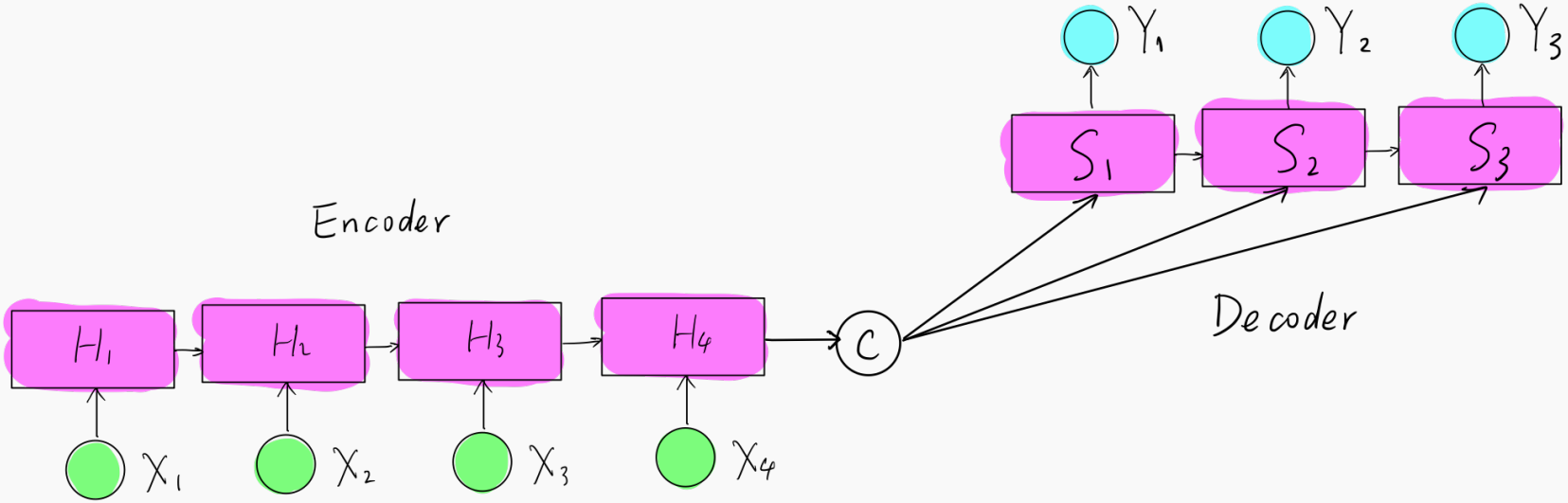


- Encoder: An ordinary RNN
  $h_t = f(x_t, h_{t-1})$
- Context vector: $c = q(h_1, \ldots, h_T)$
  Contains information in the input sequence
  We can simply set, $c = h_T$

Seq2Seq

How to model N vs M?

$\mathbb{P}(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t'=1}^{T'} \mathbb{P}(y_{t'} | y_1, \ldots, y_{t'-1}, c)$



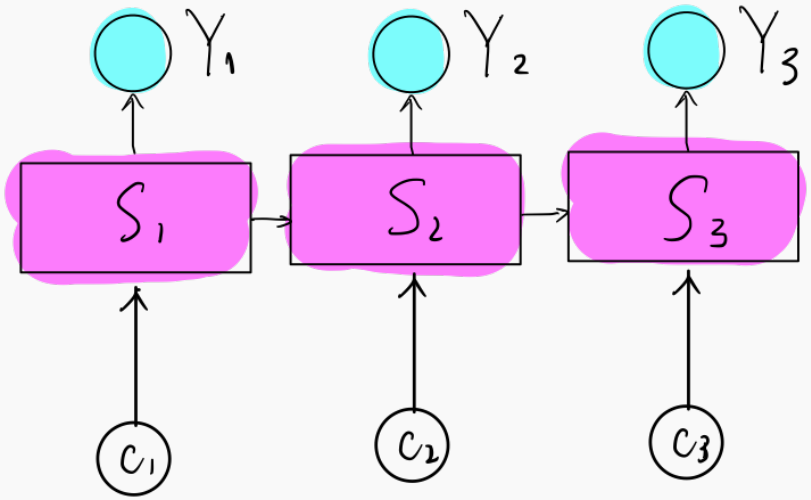
Decoder: An ordinary RNN

$\mathbb{P}(y_{t'} | y_1, \ldots, y_{t'-1}, c) = p(y_{t'-1}, s_{t'}, c)$

$s_{t'} = g(y_{t'-1}, s_{t'-1}, c)$

Attention-based mechanism

Content vector is same for all the output

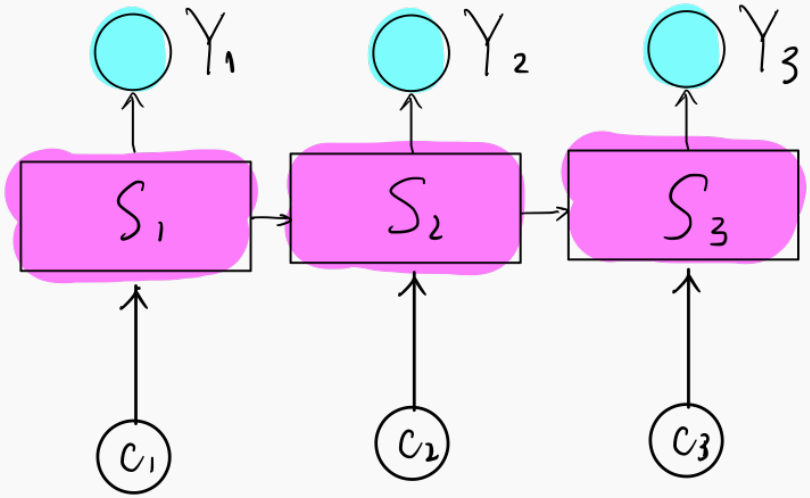


Intuition: Calculate different context vector for different time

$$s_{t'} = g(y_{t'-1}, s_{t'-1}, c_{t'})$$

$$c_{t'} = \sum_{t=1}^{T} \alpha_{t't} h_t$$

Attention-based mechanism



Intuition: Calculate different context vector for different time

$$s_{t'} = g(y_{t'-1}, s_{t'-1}, c_{t'})$$

$$c_{t'} = \sum_{t=1}^{T} \alpha_{t't} h_t$$

$$\alpha_{t't} = \frac{\exp(e_{t't})}{\sum_{k=1}^{T} \exp(e_{t'k})}$$

$$e_{t't} = a(s_{t'-1}, h_t), \text{e.g.}, e_{t't} = v^T tanh(W_s s_{t'-1} + W_h h_t)$$

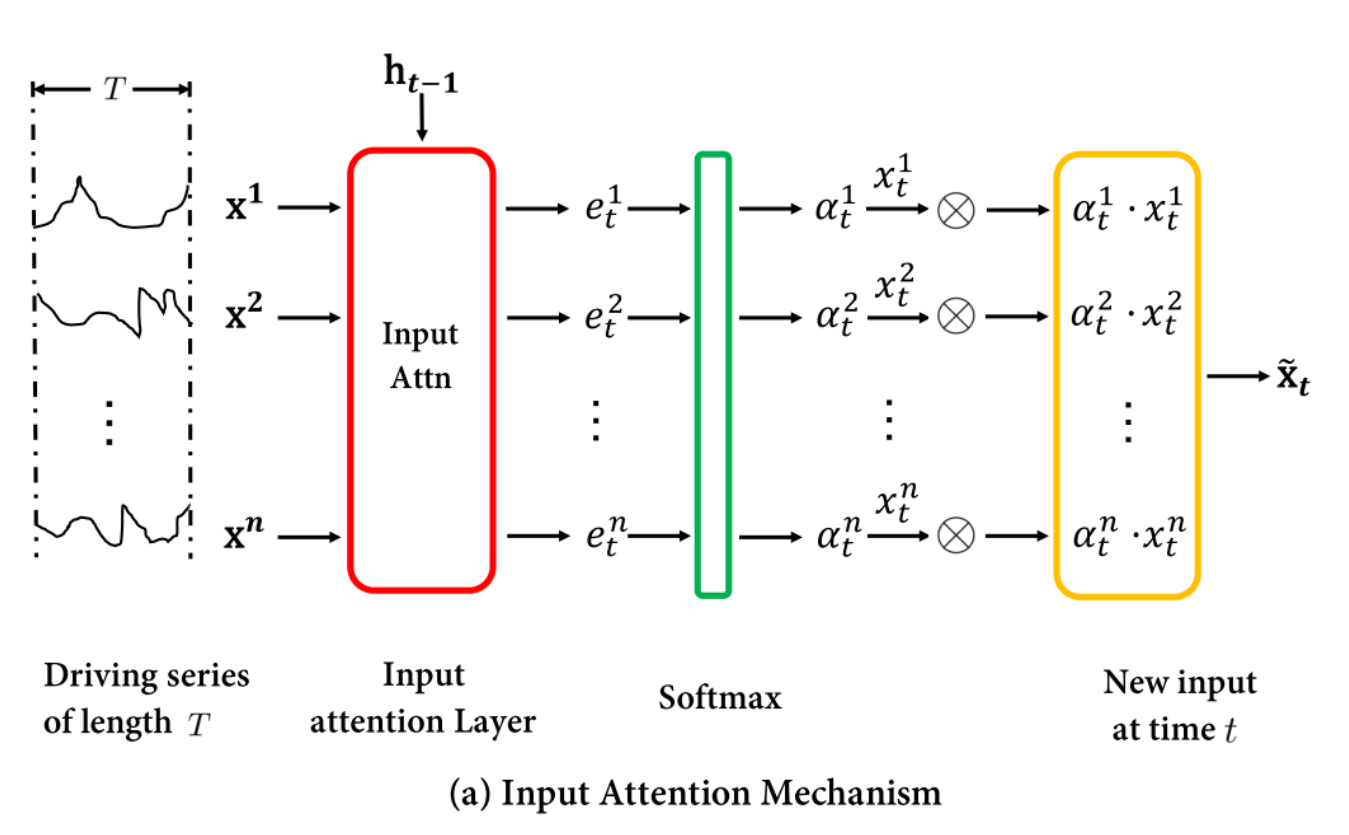
A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction

Traditional attention-based Seq2Seq model:

- Attention mechanism in the input of decoder, used for selecting relevant encoder hidden state.
- The network cannot explicitly select relevant driving series to make predictions.

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

First stage

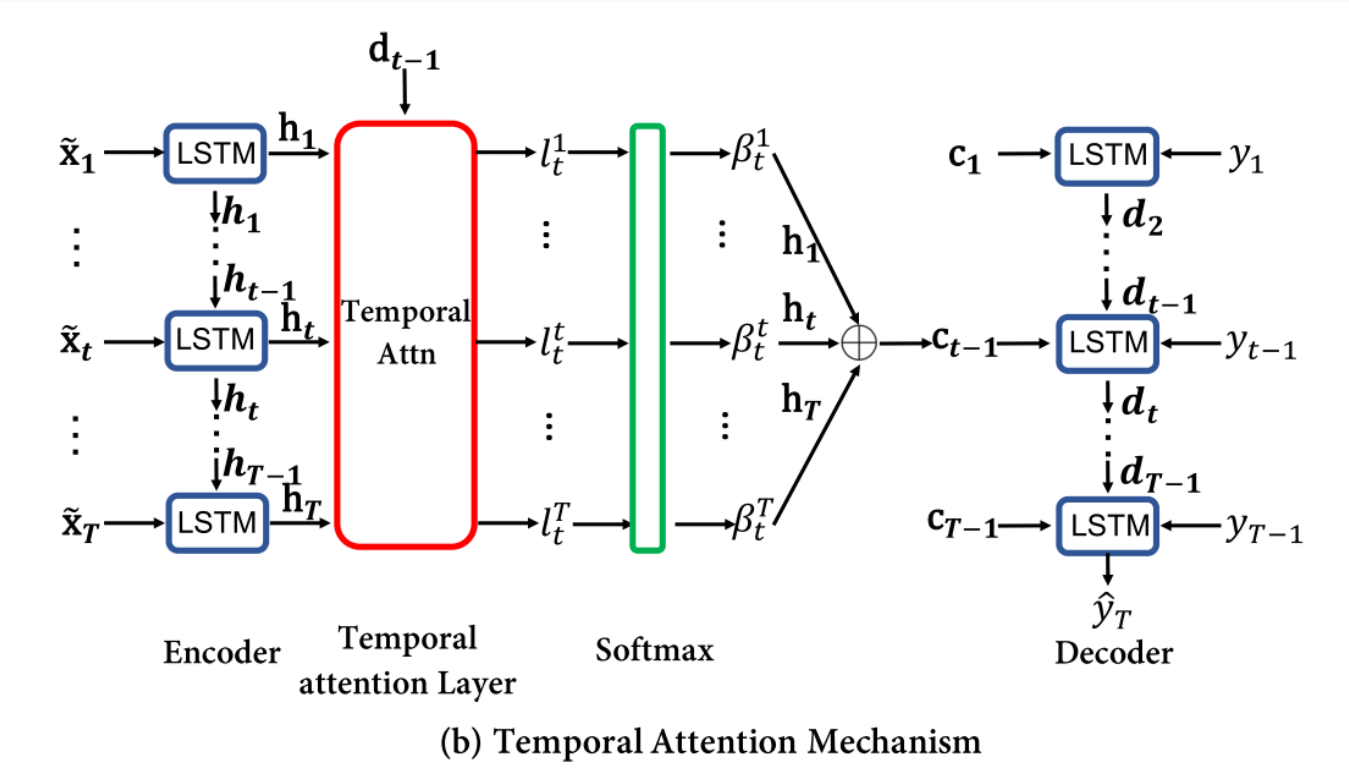


(a) Input Attention Mechanism

Introduce an input attention mechanism to adaptively extract relevant driving series (a.k.a., input features) at each time step

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

Second stage



(b) Temporal Attention Mechanism

Use a temporal attention mechanism to select relevant encoder hidden states across all time steps.

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

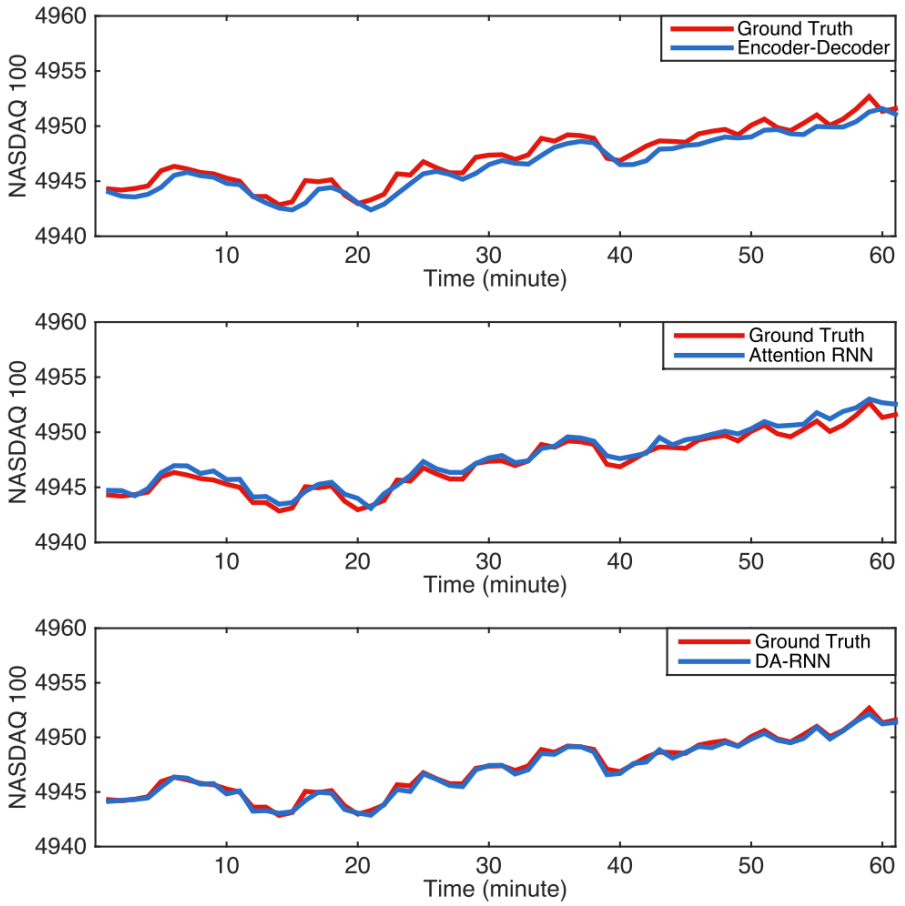


Figure 2: NASDAQ 100 Index vs. Time. Encoder-Decoder (top) and Attention RNN (middle), are compared with DA-RNN (bottom).

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

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.

Models	SML 2010 Dataset			NASDAQ 100 Stock Dataset		
	MAE ($\times 10^{-2}$%)	**MAPE** ($\times 10^{-2}$%)	**RMSE** ($\times 10^{-2}$%)	**MAE**	**MAPE** ($\times 10^{-2}$%)	**RMSE**
ARIMA [2011]	1.95	9.29	2.65	0.91	1.84	1.45
NARX RNN [2008]	1.79±0.07	8.64±0.29	2.34±0.08	0.75±0.09	1.51 ±0.17	0.98±0.10
Encoder-Decoder (64) [2014b]	2.59±0.07	12.1±0.34	3.37±0.07	0.97±0.06	1.96±0.12	1.27±0.05
Encoder-Decoder (128) [2014b]	1.91±0.02	9.00±0.10	2.52±0.04	0.72 ±0.03	1.46±0.06	1.00±0.03
Attention RNN (64) [2014]	1.78±0.03	8.46±0.09	2.32±0.03	0.76±0.08	1.54±0.02	1.00±0.09
Attention RNN (128) [2014]	1.77±0.02	8.45±0.09	2.33±0.03	0.71±0.05	1.43±0.09	0.96±0.05
Input-Attn-RNN (64)	1.88±0.04	8.89±0.19	2.50±0.05	0.28±0.02	0.57±0.04	0.41±0.03
Input-Attn-RNN (128)	1.70±0.03	8.09±0.15	2.24±0.03	0.26±0.02	0.53±0.03	0.39±0.03
DA-RNN (64)	1.53±0.01	7.31±0.05	2.02±0.01	**0.21± 0.002**	**0.43± 0.005**	**0.31± 0.003**
DA-RNN (128)	**1.50± 0.01**	**7.14± 0.07**	**1.97± 0.01**	0.22± 0.002	0.45± 0.005	0.33± 0.003

Thank you!