LSTM networks for time series simulation

Xusong Wang

A report submitted in partial fulfillment for the
Seminar of

Machine Learning in Numerical Simulation

Institute for Parallel and Distributed Systems
Universität Stuttgart

Supervisor: M.Sc. Raphael Leiteritz

Winter Semester 2019/2020

Contents

Chapter 1

Introduction

1.1 Neural Networks(NN)

Neural networks are biologically motivated models of computation. They are well satisfied with the machine perception task, it is not like the hand-engineered features. A neural network conclude a set of artificial neurons,which are labeled j, associating the neuron are the link function(also know as activation function) labeled $l_j(.)$, the output value v_j of each neuron j is calculated as below:[6]

$$v_j = l_j(\sum_{j'} w_{jj'}.v'_j). = l_j(a_j).$$

By applying its activation function, with a weighted sum of the values of its input node: $w_{jj'}$ is a weight associated from node j' to j ,and activation also noted as $a_j = \sum_{j'} w_{jj'}.v'_j$

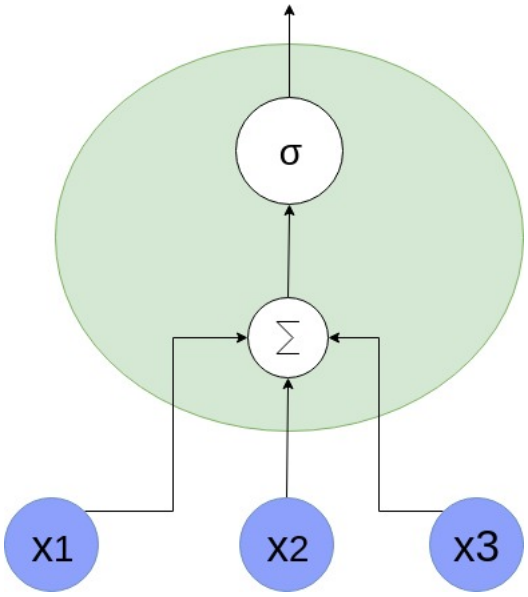

Figure 1.1: Neural Network

When the exact activation function with a sigmoid function is applied, then the output is $\sigma(a_j) = \frac{1}{1+e^{-a_j}}$. Other activation such as: $tanh$ function $\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ also become common in feedforward

neural networks. Rectified linear unit(ReLU) $l_j(z) = max(0, z)$ has been demonstrated well in DNN . For the multiclass classification task, a softmax non-linearity in an output layer of K nodes is applied

$$h_k = \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}}$$

for $k = 1$ to $k = K$. The denominator is a normalizing term consisting of the sum of the numerators, ensuring $h_1 + h_2 + ..h_K = 1$

1.1.1 Feedforward networks and backpropagation

The Order in which computation should proceed must be determined. x is the lowest layer, and each higher layer is successively computed until the output is generated at the most top layer $\check{y}$. Learning is by iterated weights, to minimize a loss function $L(h, y)$. Back-propagation, which introduced by Rumlhart et al.[8], is the most successful algorithm for training the NN. The chain rule is used to calculate the derivative of the loss function $L(h, y)$. Gradient descent is used to updates the weights. If Back-propagation will reach the global minimum is no guarantee, while the loss surface is convex.

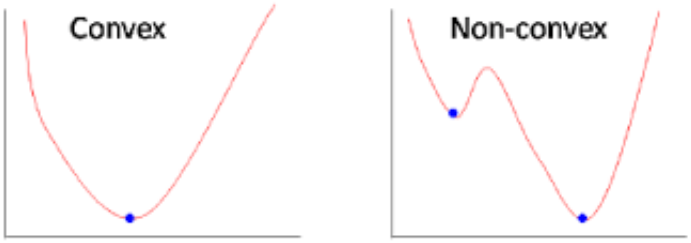

Figure 1.2: Convex and Nonconvex

Stochastic gradient descent is often used to train NN.$w \leftarrow w - \eta \nabla_w F_i$ η is the learning rate, $\nabla_w F_i$ is the gradient of the object function which respect to the parameters w. Process of the back-propagation as follows: First, an example is propagated forward through the network to produce a value v_j at each node and output $\check{y}$ at the toppest layer. Then, a loss function value $L(\check{y_k}, y_k)$ is computed at each output node k, afterwards, for each output node k, we calculate

$$\delta_k = \frac{\partial L(h_k, y_k)}{\partial h_k} \Delta l'_k(a_k)$$

Then we calculate $\delta_j = l'(a_j) \sum_k \delta_k \Delta w_{kj}$. This calculation is performed successively for each lower layer to yield δ_j. Each value δ_j represents the derivative $\partial L / \partial a_j$ of the total loss function . v_j calculated during the forward pass, and the value δ_j calculated during the backward pass. The derivative of the loss L with respect a given parameter $w_{jj'}$ is $\frac{\partial L}{\partial w_{jj'}} = \delta_j v_{j'}$.[6]

1.2 Recurrent Neural Networks(RNN)

Recurrent neural networks are feedforward neural networks increased by the inclusion of edges that span adjacent time steps, introducing a notion of time to the model.

1.2.1 Why Use RNN

In past years datasets are far larger.DNNs and CNN have demonstrated outstanding results. But DNN and CNN rely on the assumption of independence of the training dataset. If datasets are not independently generated, related in time or space, the result is unacceptable. Additionally, a standard neural network is just vectors of fixed length. RNN can model input and output consisting of sequences of elements that are not independent. A model trained using an x-length context window could never be trained to answer a simple question with an x+1 length. The modern interactive system of economic importance includes driver-less cars or robotic surgery. An explicit model of sequentially or time is necessary. For example,$x^1, x^2, x^3..x^t$ are input sequence,$y^1, y^2, y^3..y^t$ are target sequence,when a model predicted data points,predicted data are labeled h^t, A model,trained using a 3-length context window,could be able to predicted the sequence, such as "Yann climbs the mountain",where $x^1 = Yann$,$x^2 = climbs$, etc.

1.2.2 Standard Recurrent Neural Network

Recurrent neural networks are the neural networks with loops in them. And they store the information. Therefore, they are able to predict the output from the previous information. However, Long Short-Term Memory (LSTM) network is a particular type of recurrent network that works better in practice, because of its more powerful update equation and some appealing back-propagation dynamics.

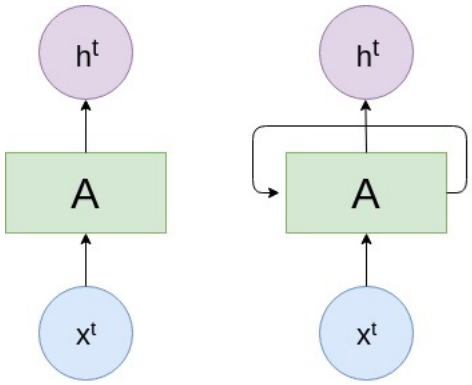

Figure 1.3: Compare between NN and RNN

This loop allows the information to be passed from one step to the next step. There is a more explicate way to demonstrate, that is, to unroll the recurrent neural network.

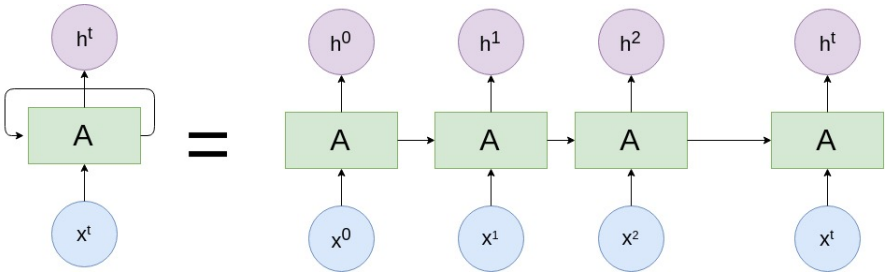

Figure 1.4: the unrolled standard RNN

The chain clearly demonstrates that recurrent neural networks store the information and take advantage from the previous information. The hidden node value is given by the equations below:

$$h^t = \sigma(W^{hx}x^t + W^{hh}h^{t-1} + b_h)$$

h^t : hidden node values of current state
h^{t-1} : hidden node values of previous state
x^t: current input date point W^{hx}: the matrix of conventional weights between the input and the hidden layer
W^{hh}: the matrix of recurrent weights between the hidden layer and itself at adjacent time steps
b_h: bias parameters which allow each node to learn an offset
$\sigma(.)$: activation function , e.g. tanh - function implements a non-linearity that squashes the activations to the range [-1 , 1]

Considering the architecture of the RNN, if the tanh function is taken as the activation function, zooming in the network structure, for standard RNN, a simple repeating module is the structure of network as below.

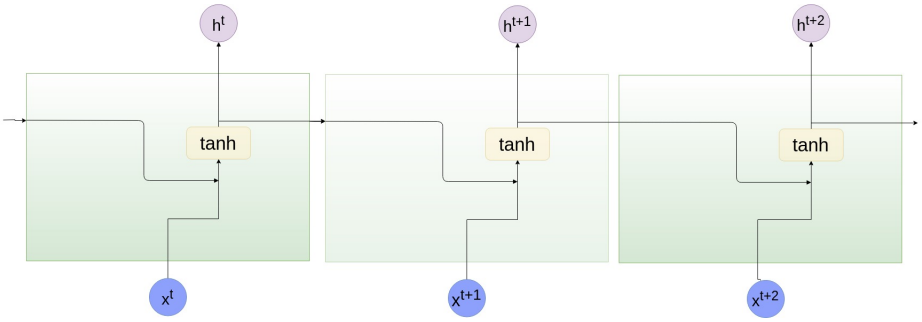

Figure 1.5: a typical RNN with tanh as the activation function

In a manner of coding, the RNN's API consists of a single step function

```
rnn = RNN()
y = rnn.step(x)
```

where x is an input datapoint, y is the RNN's output vector, below is a class step function of RNN wrote by python: [5]

```
class RNN:
  # ...
 def step(self, x):
   # update the hidden state
   self.h=np.tanh(np.dot(self.W_hh, self.h)+np.dot(self.W_xh,x))
   # compute the output vector
   y = np.dot(self.W_hy, self.h)
   return y
```

1.2.3 The Problem of Standard RNN

Recurrent Neural Network demonstrates that it in principle to connect previous information to the present step, such as using previous words to the current task, be able to predict the next word. However, Recurrent Neural Network takes too much time or do not perform well, especially when minimal time lags between inputs and previous signal are too long.[4]

In some cases, we require to look at recent information to perform the present task. For example, consider a language model is trying to predict the next word based on the previous ones. If a predicted model was implemented, trying to predict the last word in "English as a global language," the result might turn to be "language," the prediction only needs the previous few words, without the whole context. In a case like this, time lags between the current input and relevant information are relatively small, RNNs work well to predict.[7]

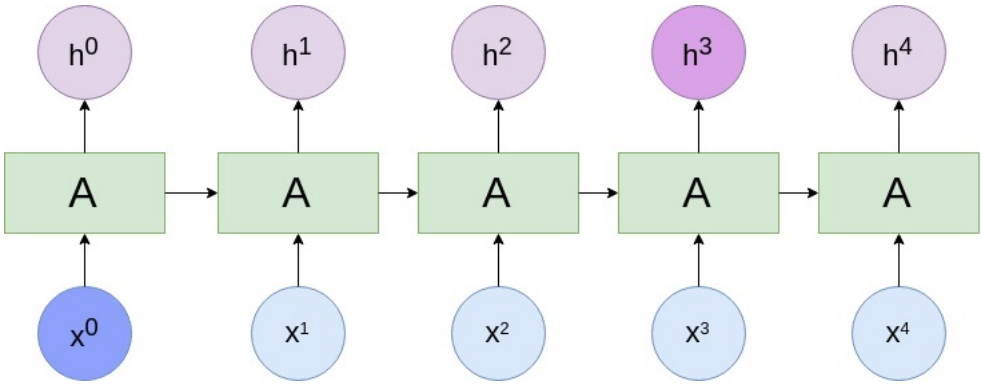

Figure 1.6: RNN with a short time lag

However, there are many examples that RNNs work not well. A typical example is that trying to predict the last word in the text, "Kylian grew up in France… Kylian speaks fluent French." From the recent information, the next word might be a kind of language, but diving into the details, the first sentence, "I grew up in France," is required. It is a relatively long time lag from the previous relevant information to the current input, which RNNs perform not very well. It indicated that as that gap grows, we need another algorithm to connect the information. These problems are theoretically revealed by the Hochreiter(1991)[3], error signals "flowing backward in time" tend to either blow up or vanish, which turn to be either oscillating weights or take an amount of time or does not work at all.

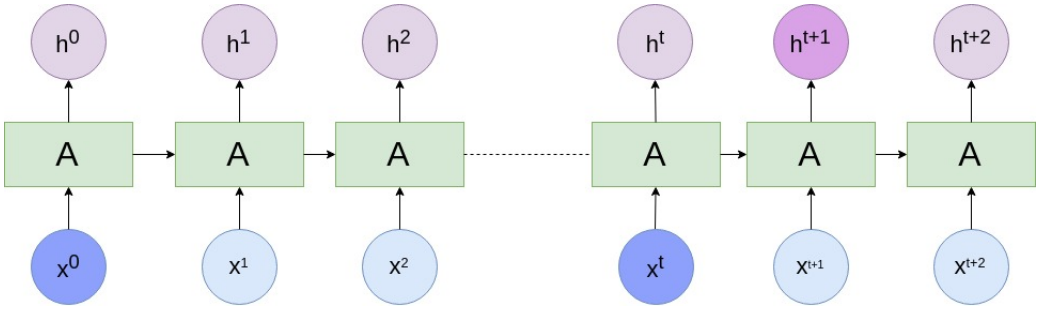

Figure 1.7: RNN with a long time lag

Chapter 2

Long short-term memory(LSTM)

2.1 Introduction of LSTM

Long Short-Term Memory networks – also known as "LSTM" – are a particular type of RNN, but they are more capable of learning long-term dependencies. LSTM was introduced by Hochreiter and Schmidhuber (1997). [4]

The LSTM architecture described uses carefully designed nodes with recurrent edges with fixed unit weight as a solution to the vanishing gradient problem. It is explicitly intended to bypass the long-term dependency problem, remembering information for long periods of time is practically their default behavior.

2.2 LSTM Cell Structure

All recurrent neural networks have the form of a chain of repeating modules of neural network. Look back to the standard RNNs, it is a repeating module with simple structure. LSTM also has chain like

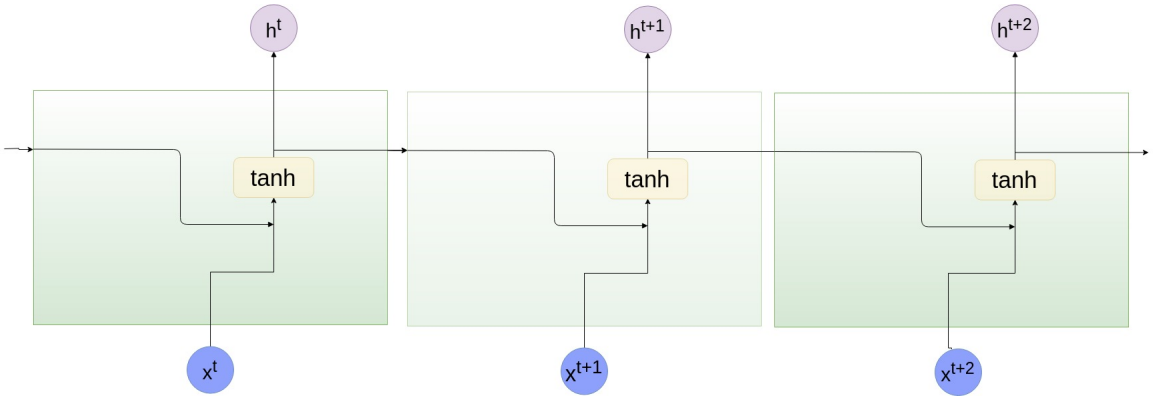

Figure 2.1: a typical RNN with tanh as the activation function

structure, but the repeating module has a different structure. Instead of a single neural network layer, there are four different and mutual layers, which are designed to interact each other. There are some node called gates that are specially defined in the LSTM, they are like filters, which allow the optional information

to pass,that if the value of gate turns to be zero, then flow from the other node is cut off. The other way around, if the value of the gate is one, all flows passed through.

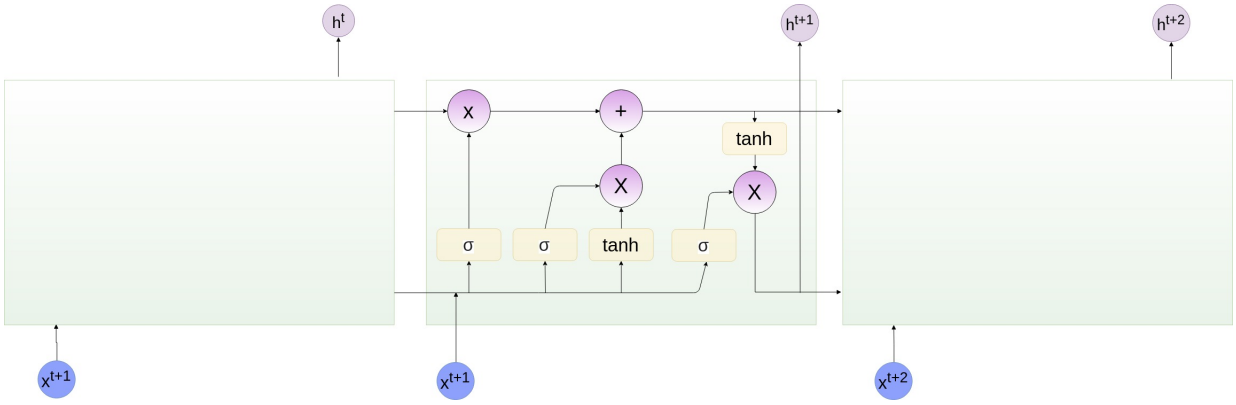

Figure 2.2: Architecture of a LSTM

- Input node: In the original LSTM paper, the activation function of input mode is a sigmoid layer, however, a tanh layer is typically used these days. This unit is labeled with $g^{(t)}$, which takes activation in the standard way from the input layer $x^{(t)}$ and at the current time step and from the hidden layer at the previous time step $h^{(t-1)}$.

$$g^{(t)} = \tanh(W^{gx}x^{(t)} + W^{gh}h^{(t-1)} + b_g)$$

- Input gate layer: A input gate layer is also a sigmoid layer, it takes activation from the current data point $x^{(t)}$ and $h^{(t-1)}$,which is from the hidden layer at the previous time step . The reason why it is called input gate is that, There is a multiplication between the value input layer and the value of the input mode .

$$i^{(t)} = \sigma(W^{ix}x^{(t)} + W^{ih}h^{(t-1)} + b_i)$$

- Cell state layer: the output of old state c^{t-1}, to be filtered , multiplied the output value of forget gate,into $f^{(t)} * c^{t-1}$, this is to forget the previous information. One last step to update the cell state, is to use $i^{(t)}$ to input the candidates $g^{(t)}$:

$$c^{(t)} = f^{(t)} * c^{(t-1)} + i^{(t)} * g^{(t)}$$

- Forget gate layer: These gates were introduced by Gers et al. [2000]. They are made by a sigmoid layer. This is especially useful in continuously running networks. In the example of "Kylian grew up in France... Kylian speaks fluent French." If a new person is introduced, with forget gates, the gender of Kylian is forgotten, which is also required. Further, the equation to calculate the value of forgetting the gate layer is as below:

$$f^{(t)} = \sigma(W^{fx}x^{(t)} + W^{fh}h^{(t-1)} + b_f)$$

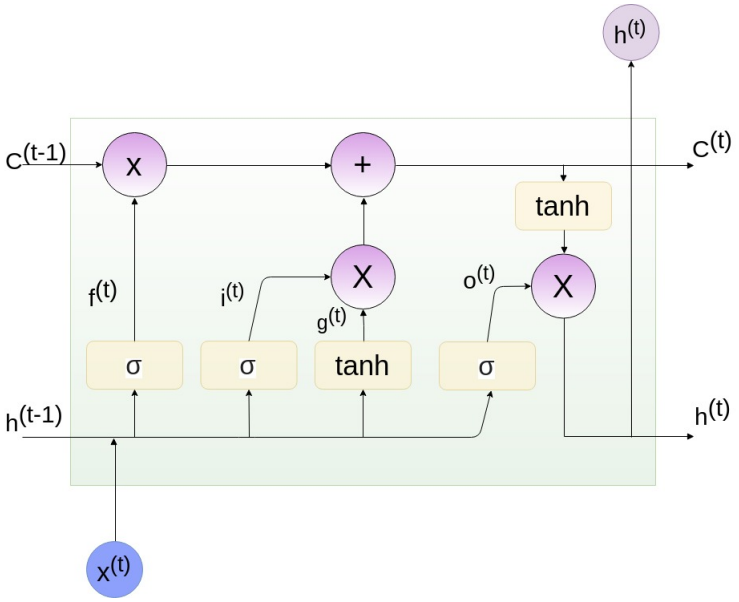

Figure 2.3: a typical LSTM Memory Cell with a forget gate

- Output gate layer: The hidden state output value is based on the cell state layer output c^t, then a tanh activation is used to squashes the value to the range [-1, 1], finally the output multiplied by $o^{(t)}$, whose filter ratio is decide by the output gate layer:

$$o^{(t)} = \sigma(W^{ox}x^{(t)} + W^{oh}h^{(t-1)} + b_o)$$

$$h^{(t)} = tanh(c^{(t)}) * o^{(t)}$$

Above is the typical LSTM with a forget gate, there are other slightly different variants, but the differences are minor.

Chapter 3

Applications of LSTM

3.1 Offline Handwriting Recognition

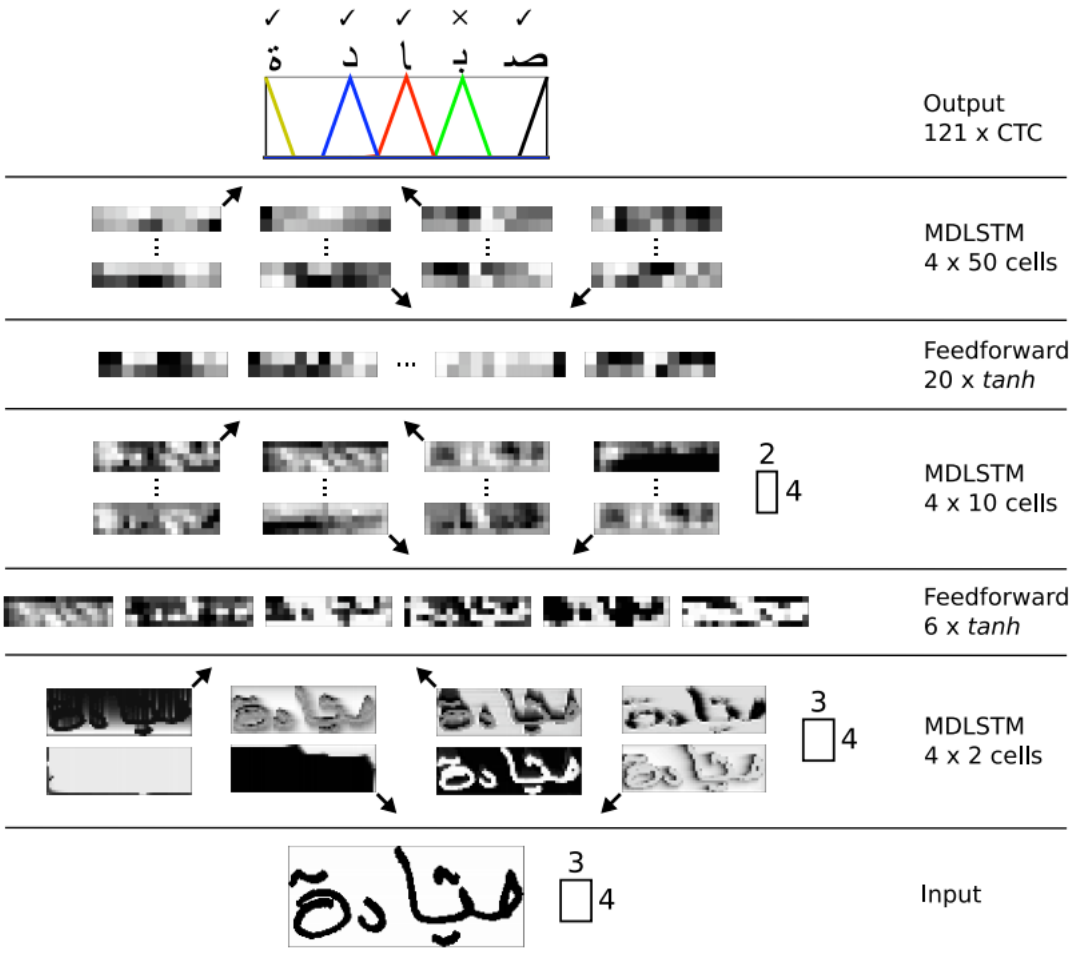

Figure 3.1: The complete recognition system

Handwriting Recognition is an interesting task which is divide into offline recognition and online recognition. In the online case, there are two features that are provided to analysis, the pen trajectory and

the image of the result. However, offline recognition is distinctly to be harder because only the image is available. Jurgen Schmidhueber, one of the Founders of LSTM, holds research in this interesting task. He combined multidimensional recurrent neural networks and connectionist temporal classification in neural network, introduced a globally trained offline handwriting recognizer that takes raw pixel data as input, with a flexibility for any language.[2]The recognizer demonstrated a outstanding performance which shined in an international Arabic recognition competition,where it outperformed all entries despite the fact that neither author understands a word of Arabic.

3.2 Exposing privacy of IoT

The IoT (Internet of Things) technology has been widely adopted in recent years and has profoundly changed people's daily lives. Researchers from the Chinese University of Hong Kong looked into the private information of the IoT devices, they proposed a traffic analysis framework based on LSTM and found the relations between packets for the attack of device identification. They used different methods to evaluate the leaking information.

config	model	average	echo dot	google home	tmall assistant	xiaomi hub	360 cam	tplink plug	orvibo plug	mitu story	xiaobai camera	broadlink plug
NAPT	RF	92.2	89.0	85.9	86.9	89.6	99.0	**99.9**	**99.9**	93.3	98.5	99.3
	LSTM	97.3	**98.5**	91.6	93.9	98.6	**99.9**	**99.9**	**99.9**	98.7	**99.9**	**99.9**
	BLSTM	**99.2**	97.0	**99.2**	**99.8**	**99.9**	**99.9**	**99.9**	**99.9**	**99.3**	**99.9**	**99.9**
VPN	RF	83.2	76.1	81.2	74.7	94.0	83.2	89.1	93.1	87.5	90.5	**99.0**
	LSTM	92.4	89.7	89.7	75.4	96.1	95.9	92.2	95.5	96.8	94.7	95.7
	BLSTM	**97.7**	**96.6**	**96.8**	**94.7**	**99.4**	**98.5**	**98.0**	**99.5**	**98.9**	**99.7**	96.7

Table 3.1: Accuracy of baseline model under pure-IoT scenario

By researching the relations between packets through models like LSTM, they showed it is possible to achieve high accuracy in device identification, even under the complex network environment. their result suggests the network communications of IoT devices do have serious privacy implications, even under encryption and traffic fusion.[1]

Chapter 4

Summary

Considering the traditional recurrent neural network, the problems of vanishing and exploding gradients occur when back-propagating errors across many time steps. It makes RNN especially challenging due to the difficulty of learning long-range dependencies. Therefore, as proposed by Hochreiter and Schmidhuber,[4] the LSTM model was introduced to overcome the problem of vanishing and exploding gradient. It resembles the input gate, output gate, forget gate and cell state, to make LSTM more capable of learning long-term dependencies.

Bibliography

[1] S. Dong, Z. Li, D. Tang, J. Chen, M. Sun, and K. Zhang. Your smart home can't keep a secret: Towards automated fingerprinting of iot traffic with neural networks, 2019.

[2] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc., 2009.

[3] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. 1991.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[5] A. Karpathy. The unreasonable effectiveness of recurrent neural networks. URL `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

[6] Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[7] C. Olah. Understanding lstm networks. URL `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[8] H. G. . W. R. Rumelhart, D. Learning representations by back-propagating errors, 1986.