INTRODUCTION TO RECURRENT NEURAL NETWORK

한재근 과장 | jahan@nvidia.com

AGENDA

Introduction to Recurrent Neural Network

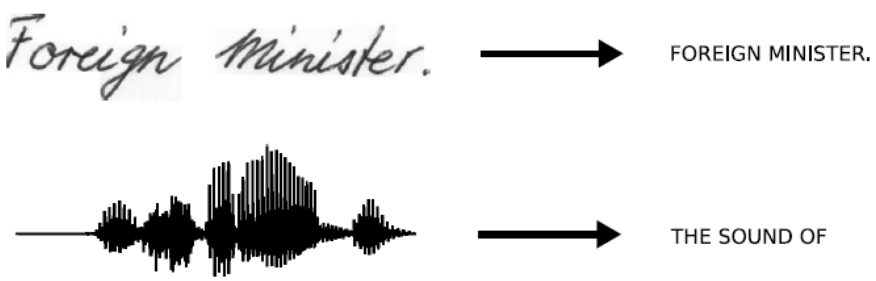
Why RNN is difficult to training

Natural Language Processing with RNN

INTRODUCTION TO RNN

SEQUENCE CLASSIFICATION AND LABELING

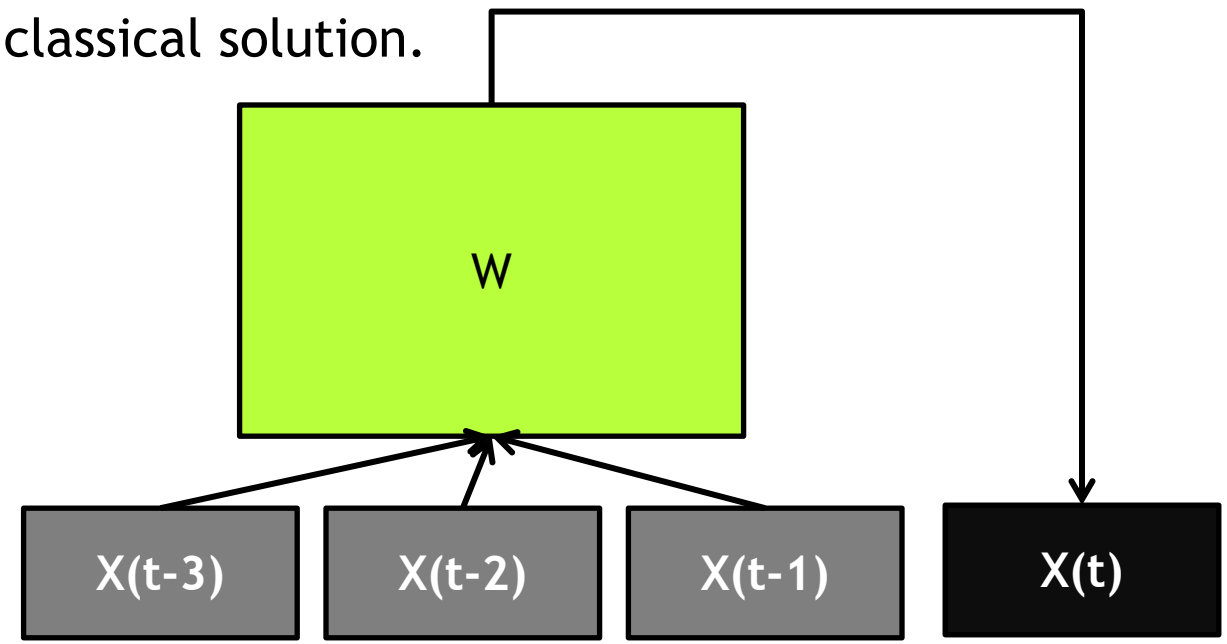
- The sequence labeling:
  - Handwriting Recognition
  - Speech Recognition
  - Language Translation



- Special case of pattern classification: individual data points cannot be assumed to be independent. Both the inputs and the labels are strongly correlated.

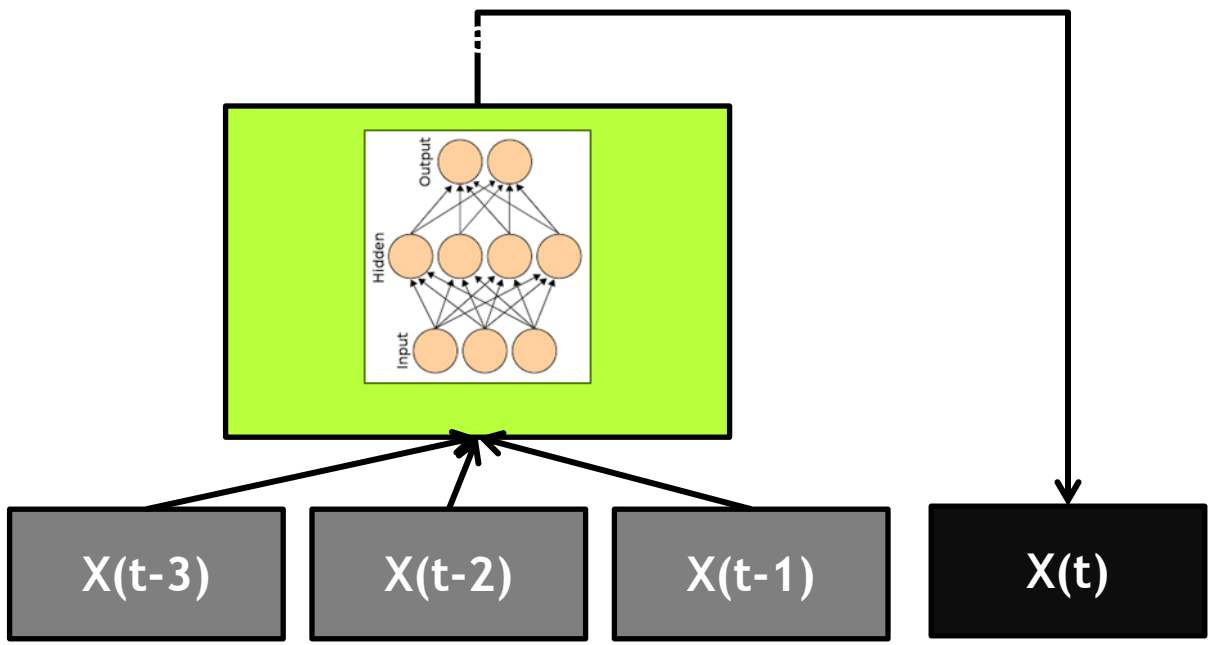
EXAMPLE

- **Problem:** Find linear function which predicts the next term x(t+1) in the input s equence based on N previous elements { x(t-N).. X(t-1) }
- Auto-regression is classical solution.



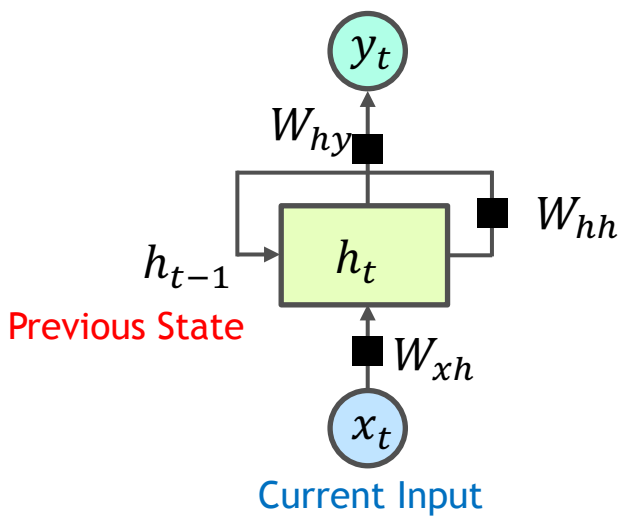
FEED FORWARD TIME-DELAY NN

- Feed-forward NN (e.g. Conv NN) is natural extension of Auto-regression:
  we use network (non-linear function) with input composed from fixed number of s
  equence elements



NEURAL NETWORK WITH MEMORY

- Previous input affects current output
- → We use this for future prediction



Current State — Previous State

$$h_t = f_W(h_{t-1}, x_t)$$

Activation — Current Input

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

AN UNROLLED RECURRENT NEURAL NETWORK

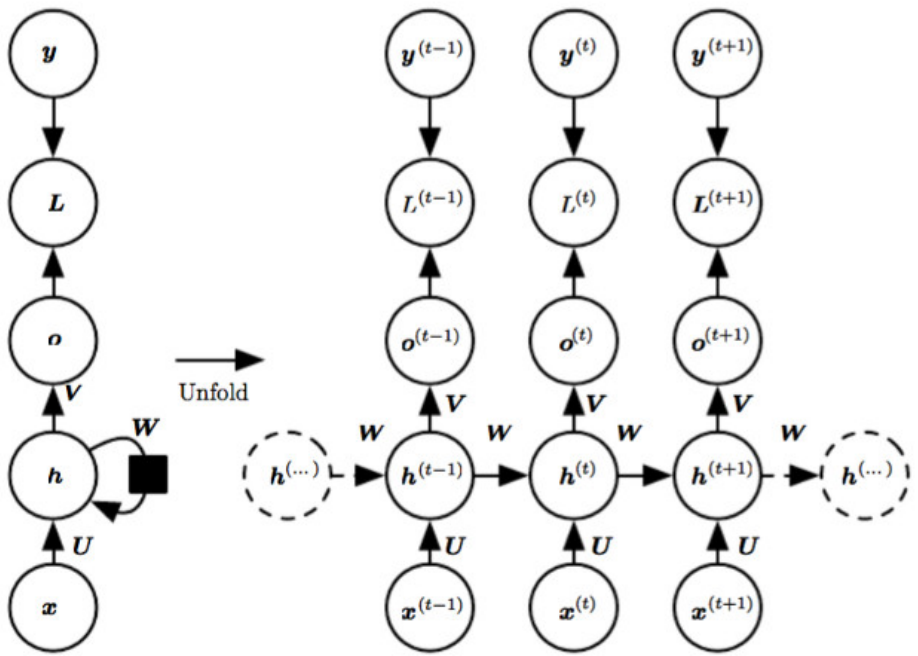


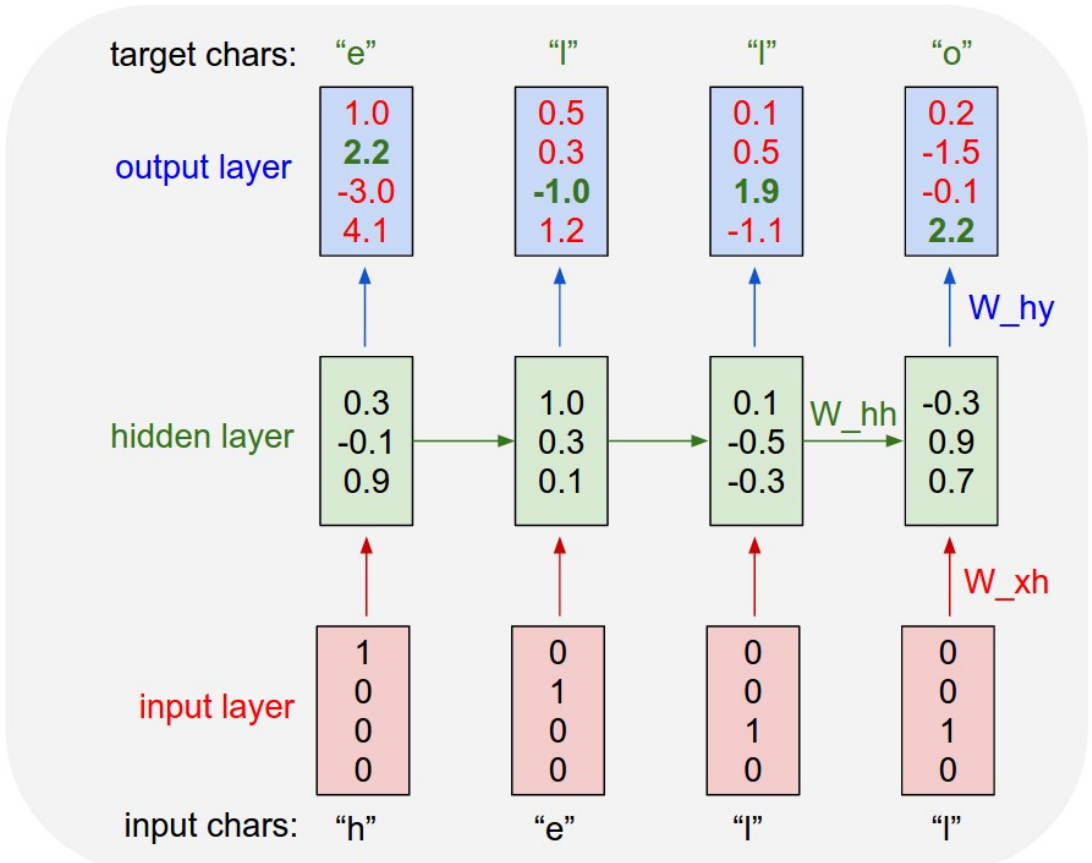
Figure 10.3: The computational graph to compute the training loss of a recurrent network that maps an input sequence of $\boldsymbol{x}$ values to a corresponding sequence of output $\boldsymbol{o}$ values.

Source: deeplearningbook.org

CHARACTER LEVEL EXAMPLE

- Vocabulary:
- [h,e,l,o]

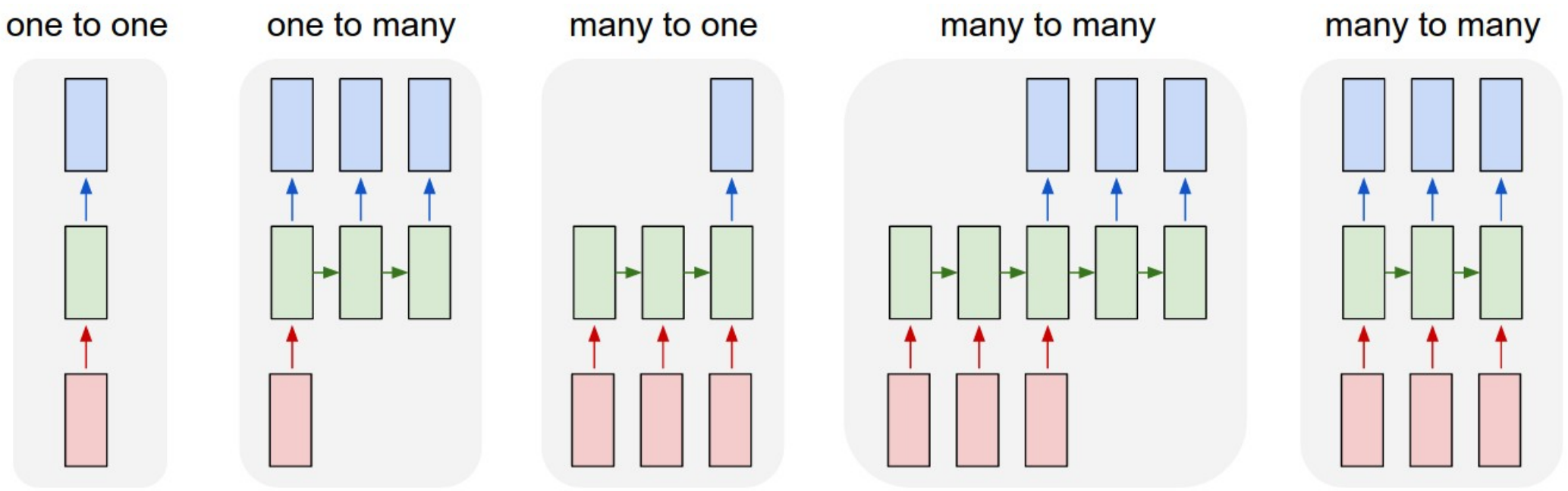
- Example training
- sequence:
- **“hello**



source: cs231n

FLEXIBLE RNNS

Examples



Vanilla NN

Image Captioning

Sentiment Classification

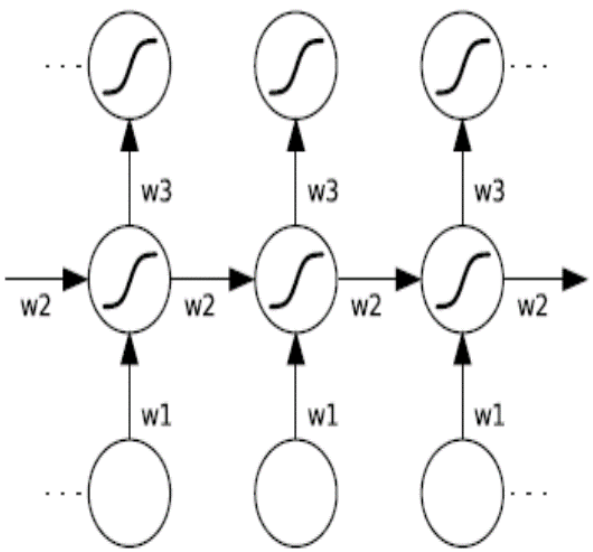
Machine Translation

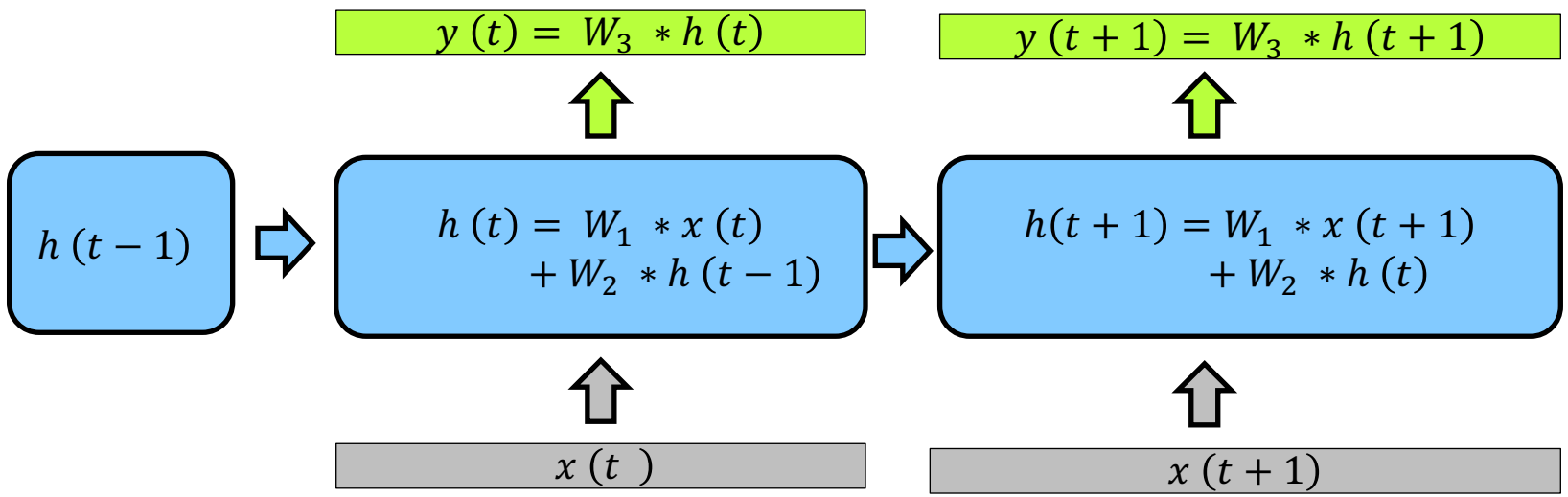
Video Classification

source: cs231n

RECURRENT NN

Let’s unfold RNN in time:

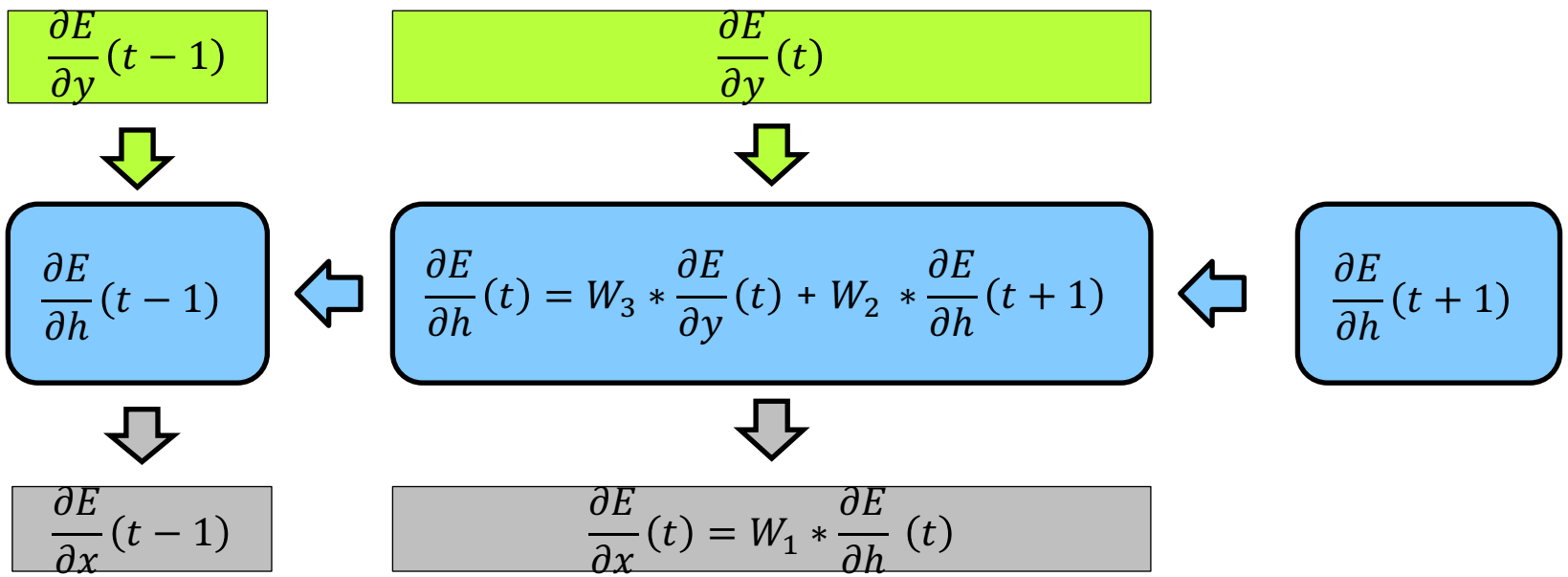


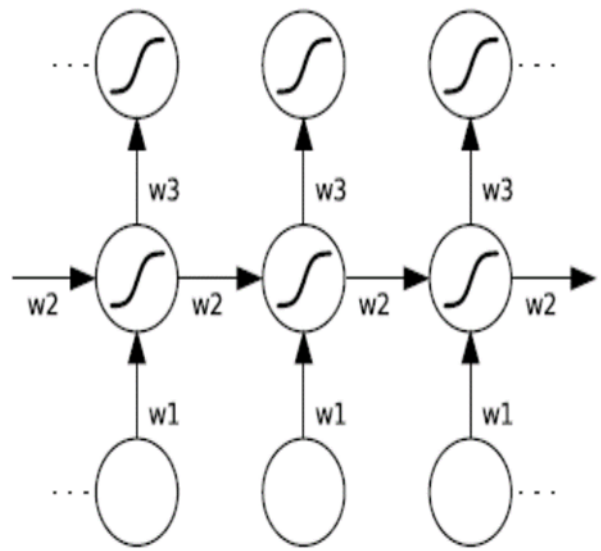


Warning: we skip bias and non-linear function for simplicity!

RECURRENT NN: TRAINING

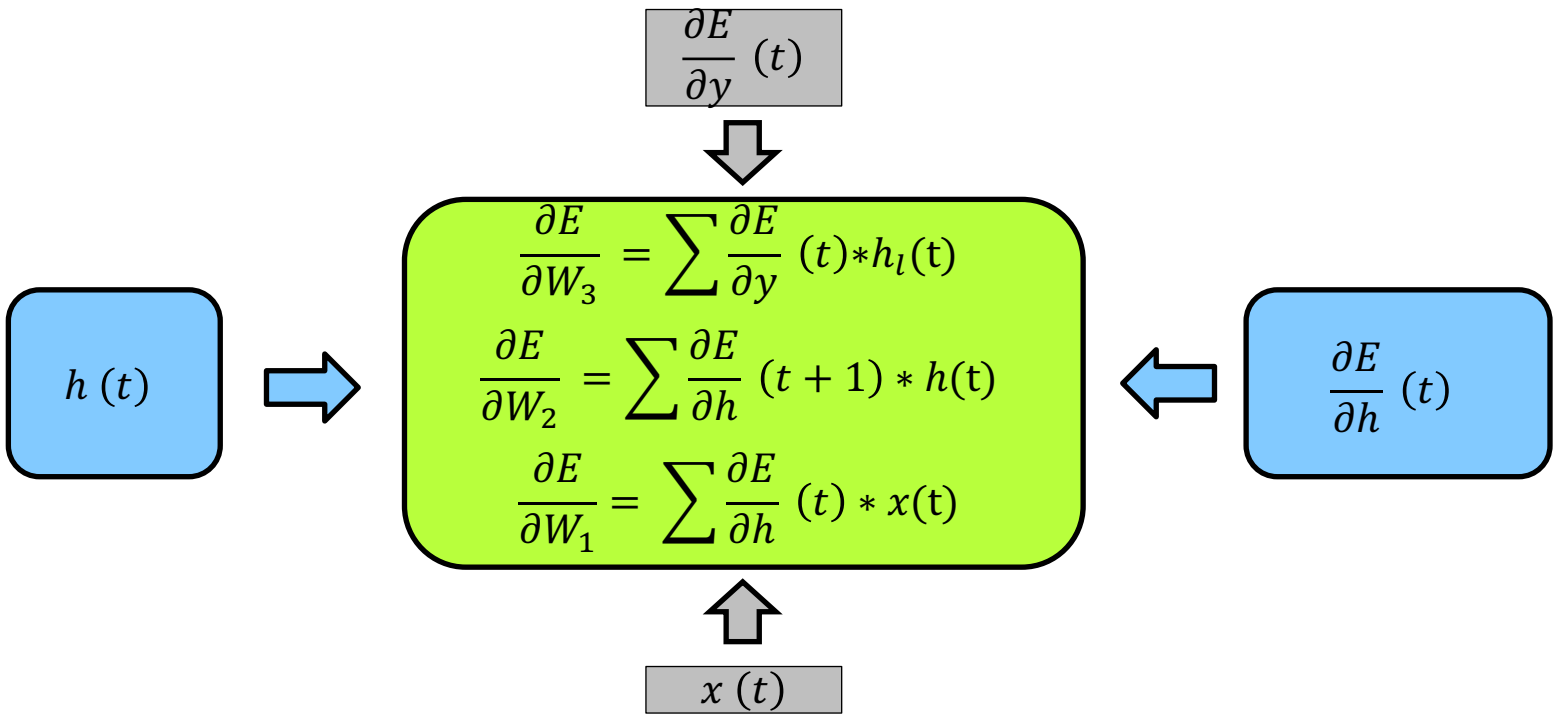
RNN training is done by using gradient back-propagation recursively back in time from T to 1. First let propagate $\frac{\partial E}{\partial y}(t)$

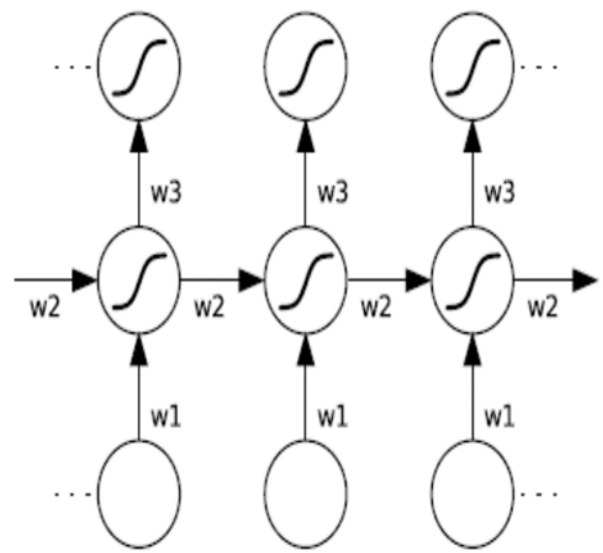




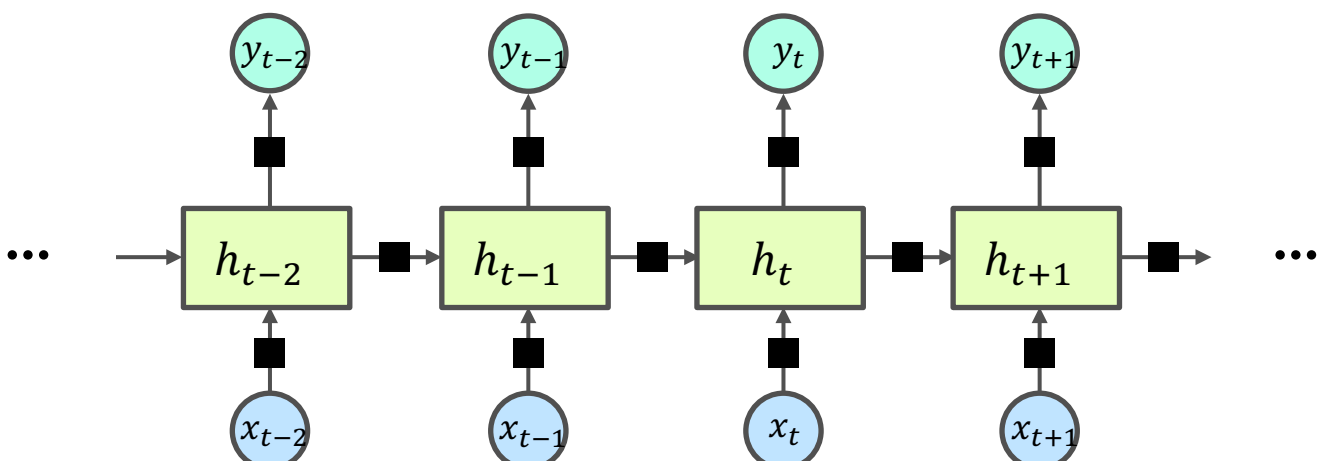
RECURRENT NN: TRAINING

After we have $\frac{\partial E}{\partial h}(t)$ we compute $\frac{\partial E}{\partial w}(t)$: we sum the "instant" derivatives with respect to the weights over all t=1..T

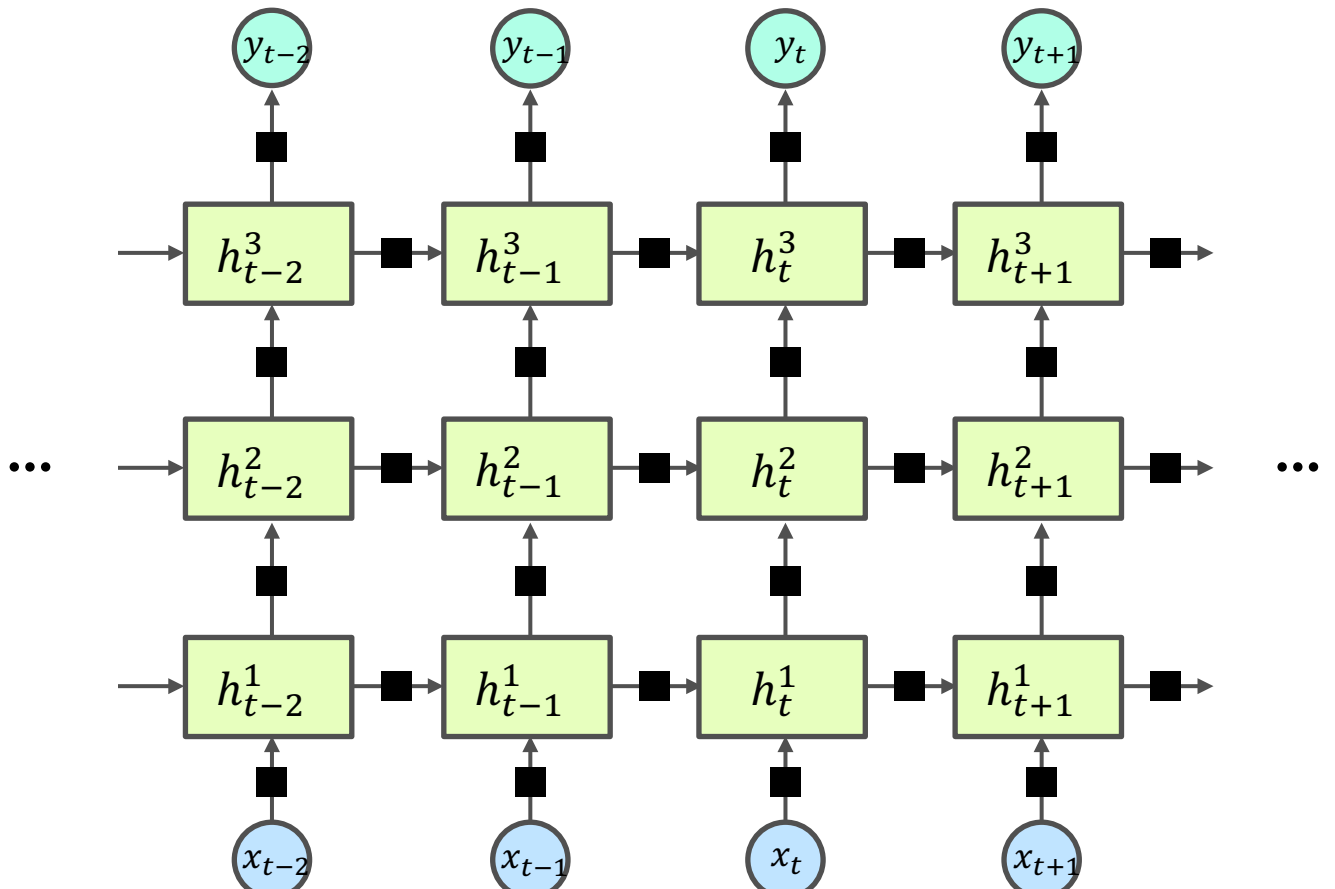




DEEP RECURRENT NEURAL NETWORK



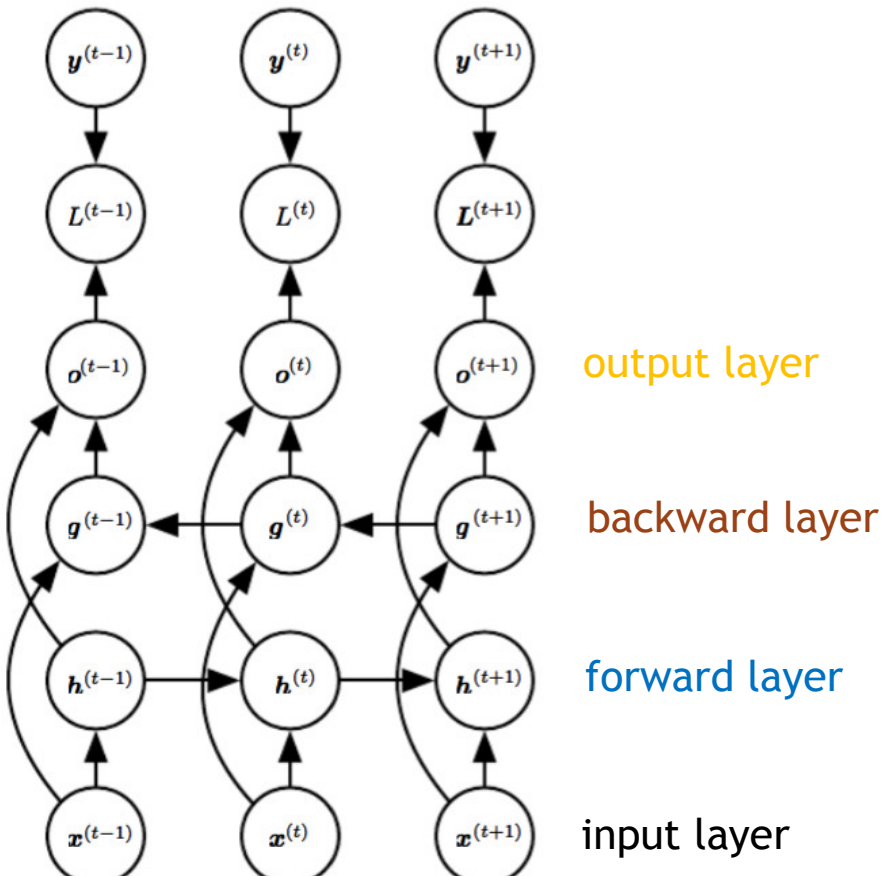
DEEP RECURRENT NEURAL NETWORK



BI-DIRECTIONAL RNN

For many problems (NLP, handwriting, speech rec ognition, protein structure prediction,…) it is ben eficial to have access to future as well as to past context.

Bidirectional RNN: split recurrent layers into two separate recurrent layers both of which are conn ected to the same output layer. There is no infor mation flows between the forward and backward hidden layers to make sure that the unfolded gra ph is acyclic



MACHINE TRANSLATION

Sequence to Sequence

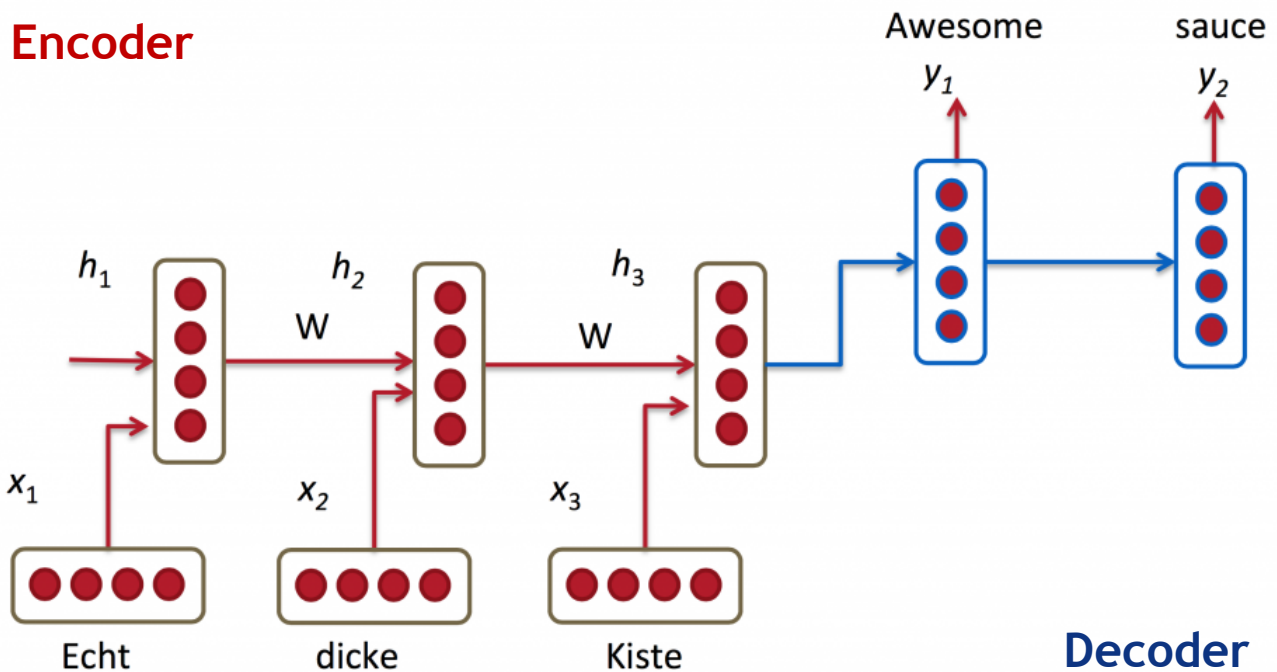
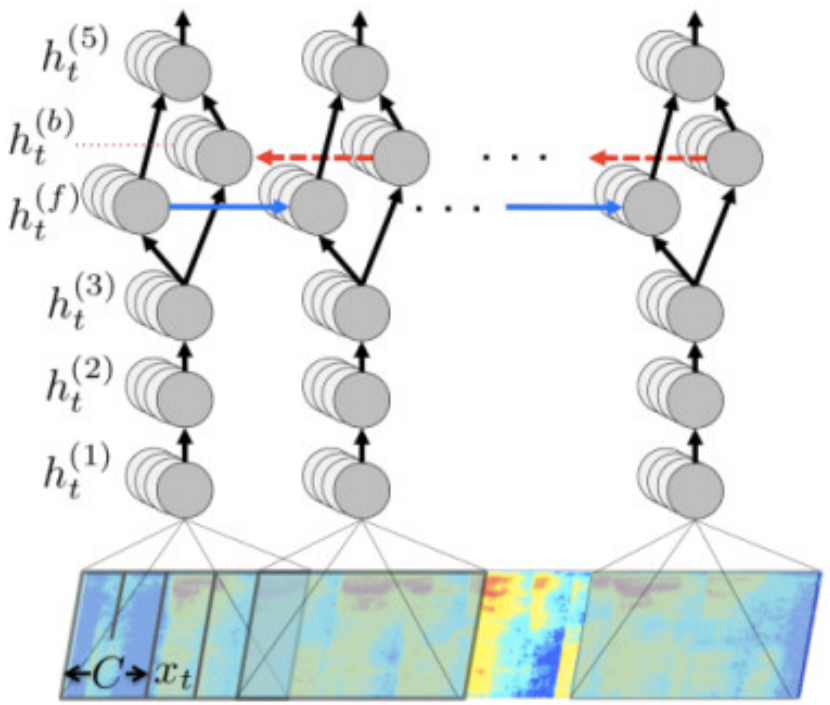


Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

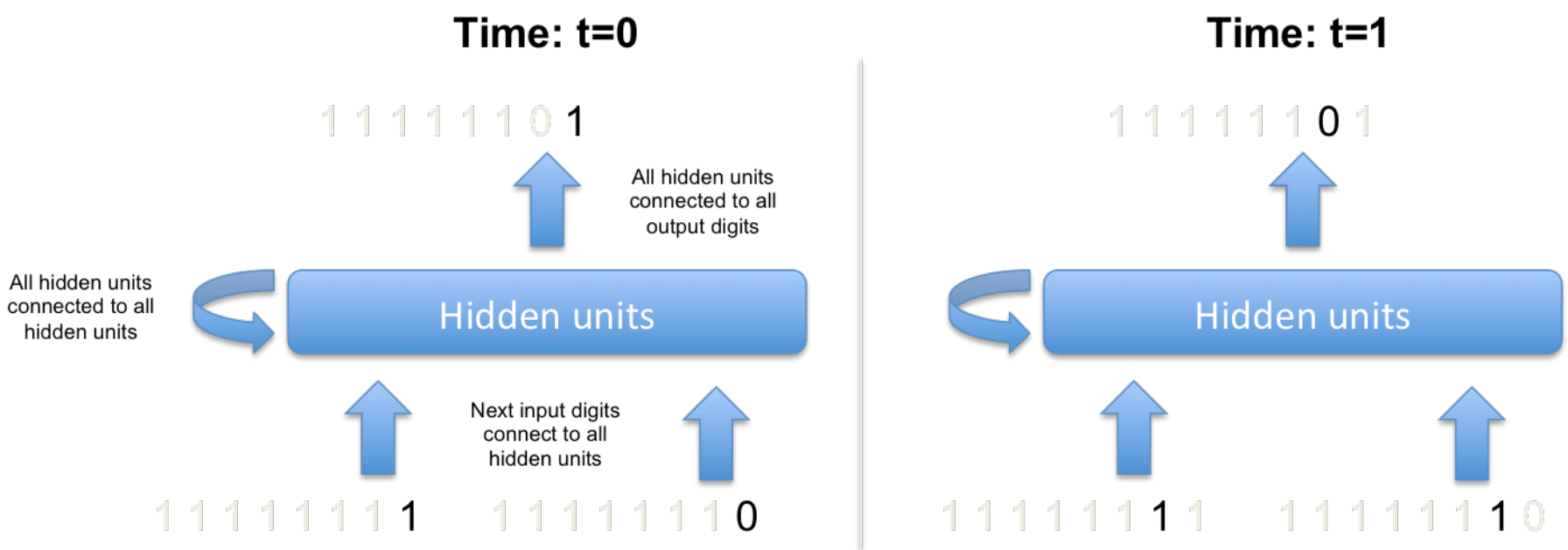
BAIDU SPEECH RECOGNITION



http://research.baidu.com/deep-speech-accurate-speech-recognition-with-gpu-accelerated-deep-learning/

EXAMPLE 1

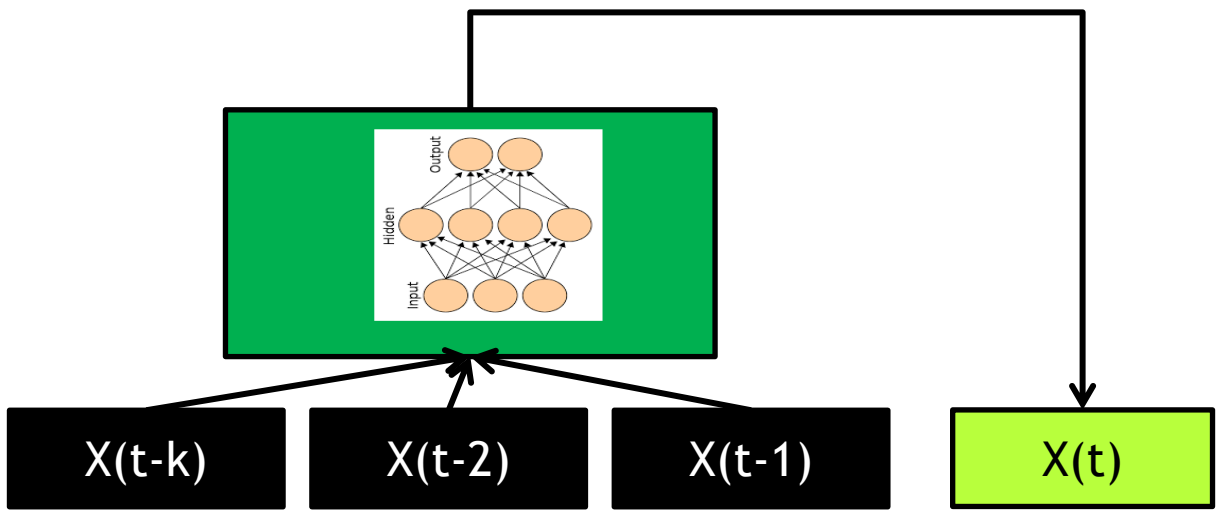
Binary addition



Natural Language Processing with RNN

FEED FORWARD NN-BASED LANGUAGE MODEL

- Feed-forward NN (e.g. Conv. NN) can be used to predict next word from *k* previous words



Dictionary= 100,000 ➔ net has 100,000 outputs. Can we scale NN to 100,000 outputs? Is there a better way to deal with such a large number of outputs?

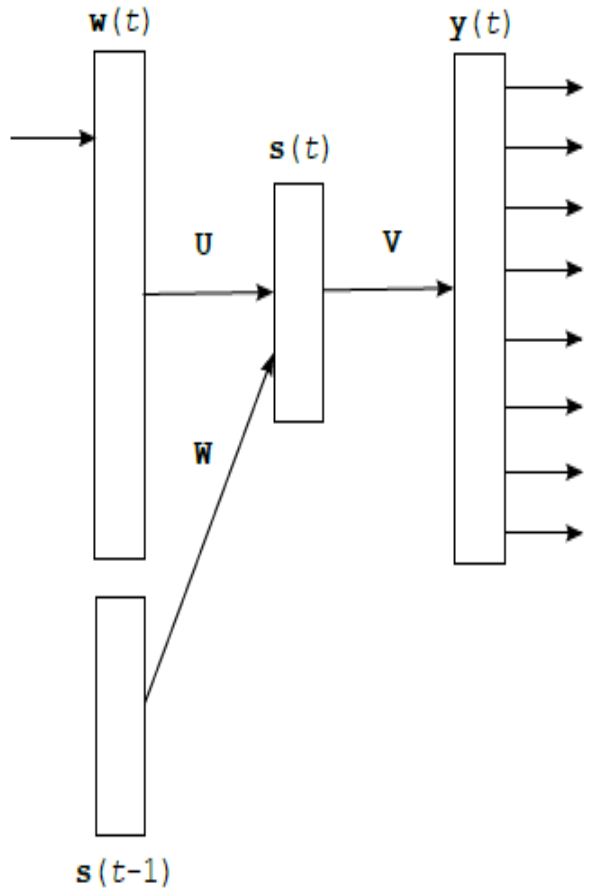
RNN - BASED LANGUAGE MODEL

- RNN allows to extend fixed input window used by TD-NN
- Input layer w and output layer y have the same dimensionality as the v
- Hidden layer s is small (50 - 1000 neurons)

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right),$$

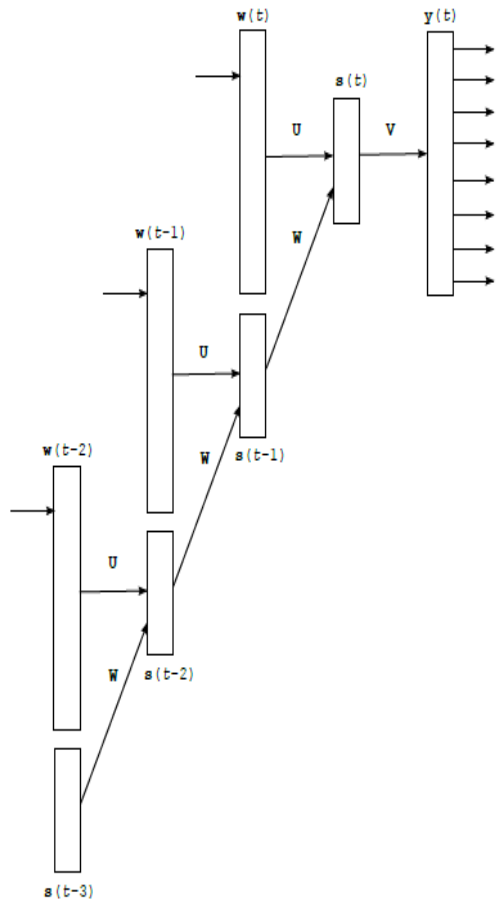
$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$



Network parameters:
U - weights between input and hidden layer
V - weights between hidden and output layer
W – recurrent weights

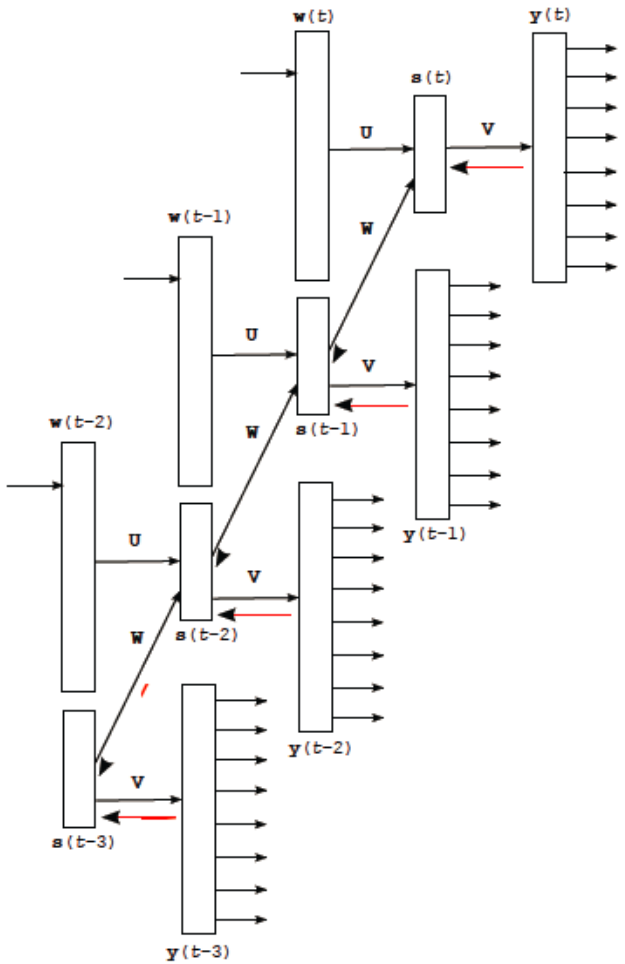
TRAINING OF RNN LANGUAGE MODEL

- RNN is trained by unfolding in time and training as
- Gradients are propagated back through time



TRAINING OF RNN LANGUAGE MODEL

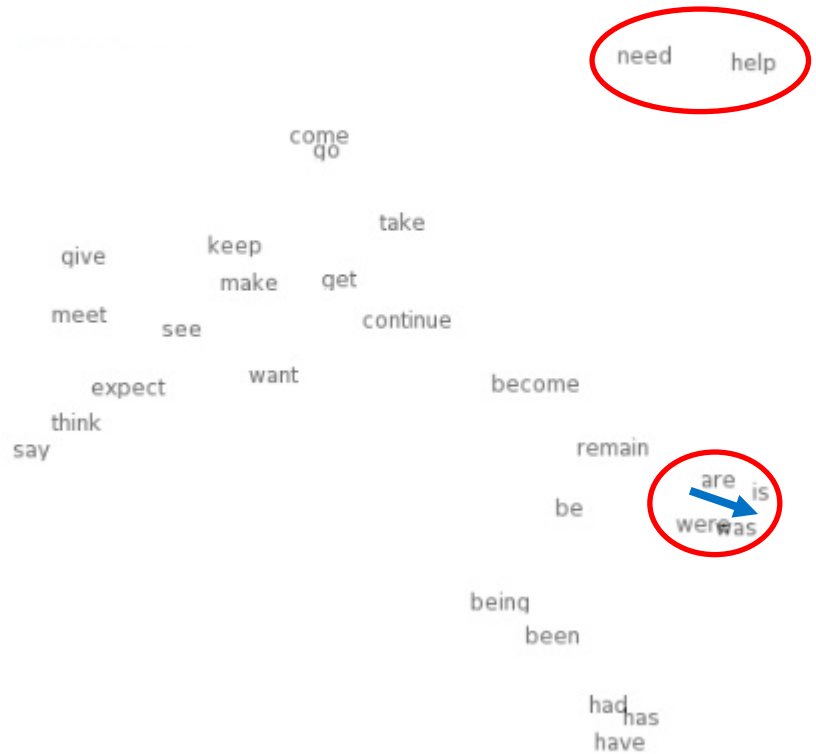
- Example: Batch training on input sequence



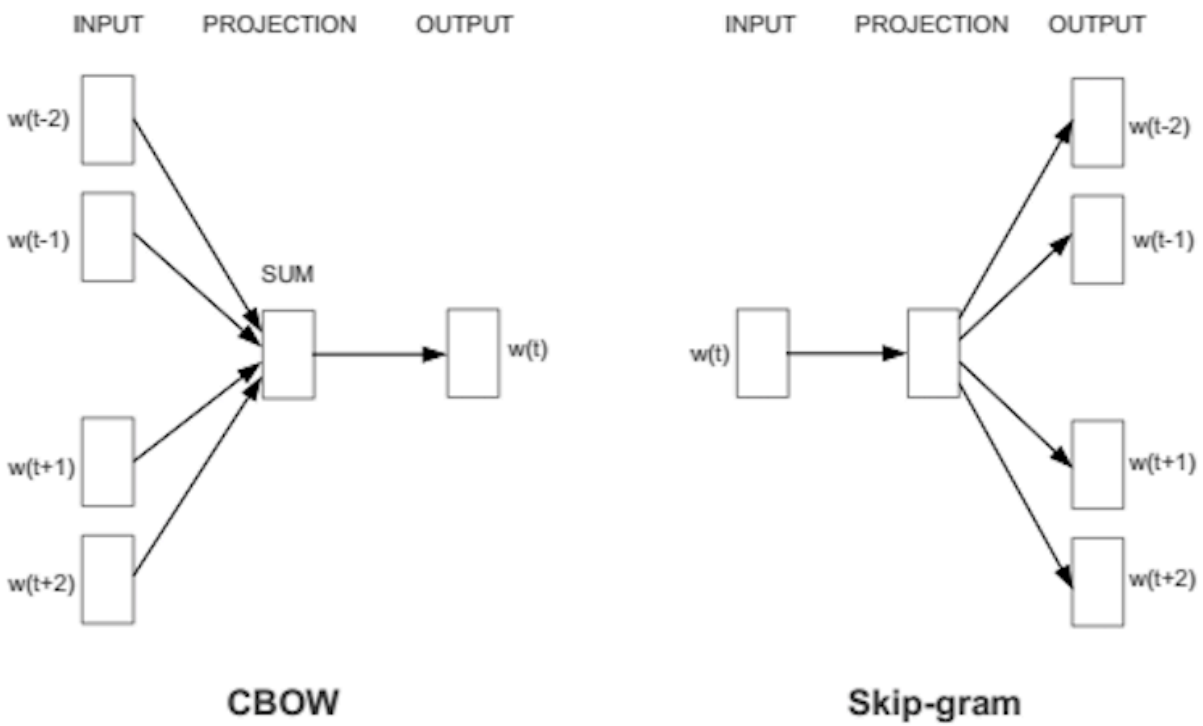
WORD EMBEDDING

- The classical NLP regards words as atomic symbols. Each word is represented as *one-hot* vector [0000010000]
  - motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] **AND**
    hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0
- Word embedding - a word is represented as a dense vector

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



WORD2VEC

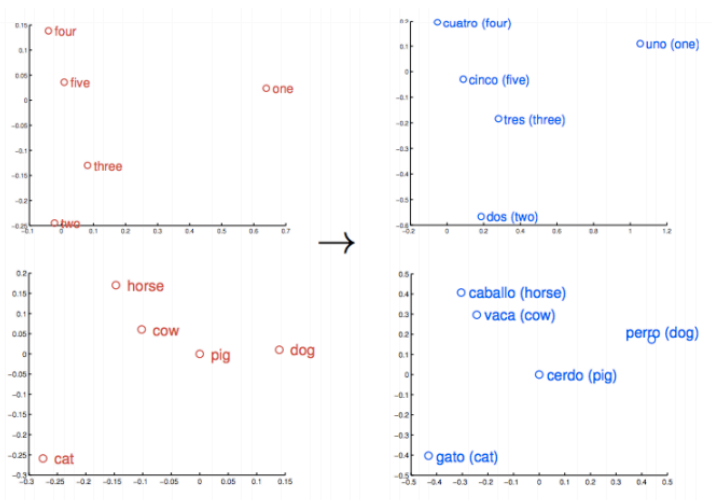

INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

WORD EMBEDDING

Syntactic analysis using word2vec

Closed to Sweden

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408



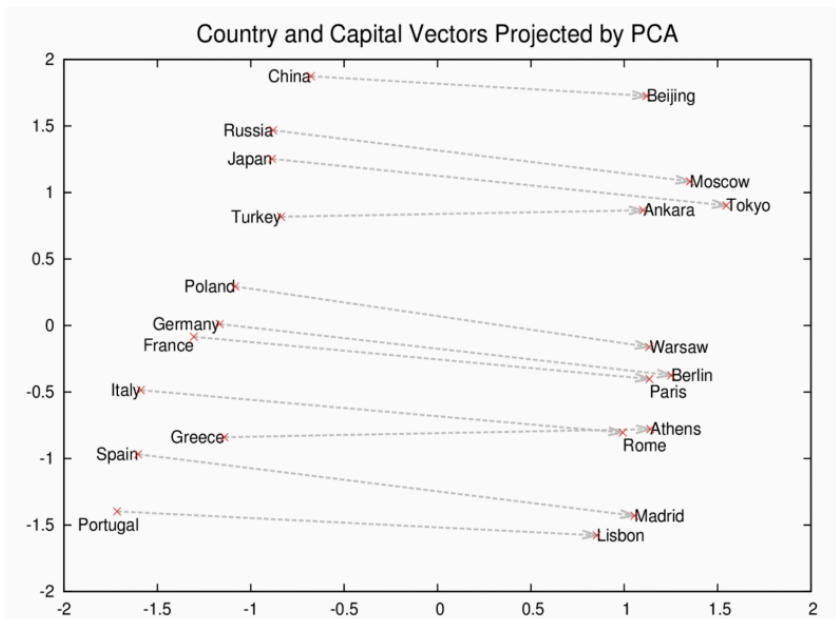


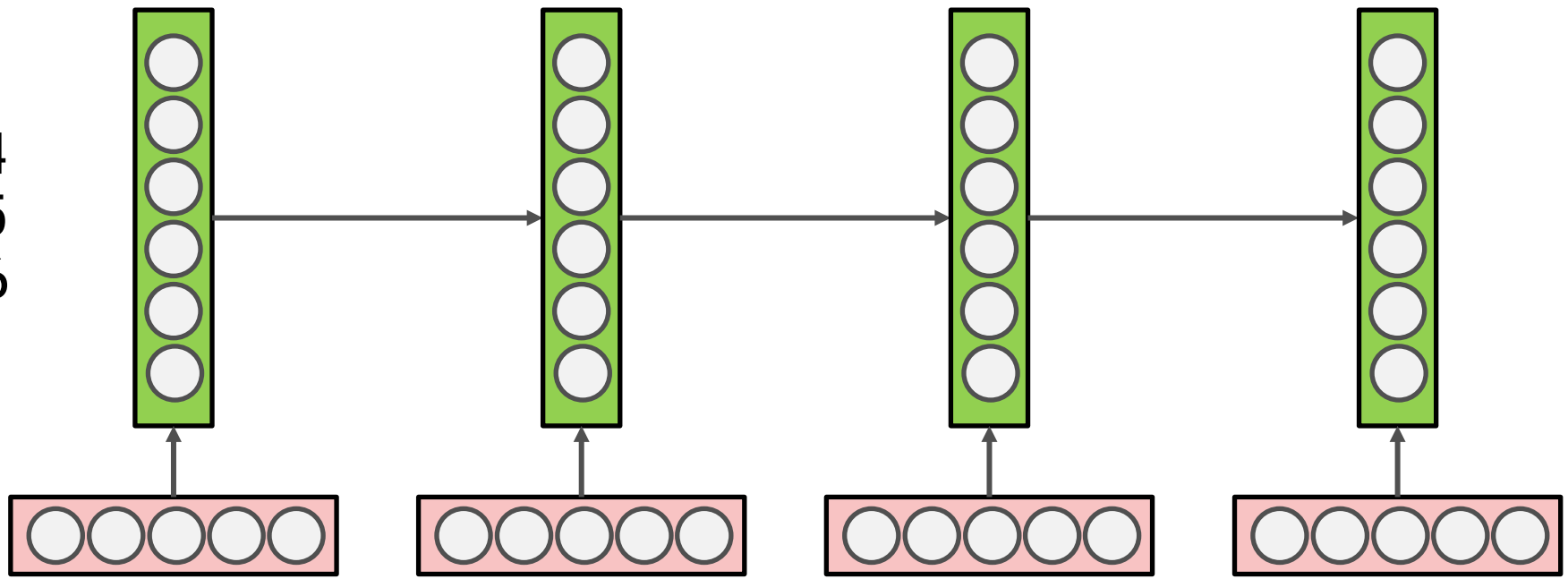
Image Source: DL4J's word2vec guide

EXAMPLE 2

Character level text generation

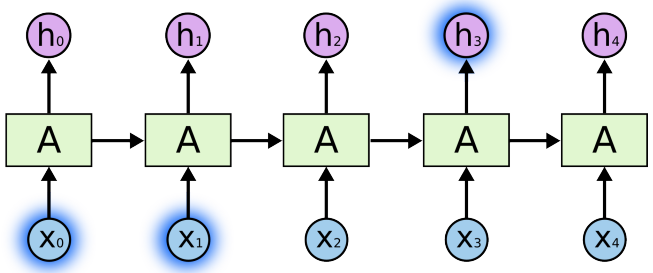
RNN EXAMPLE

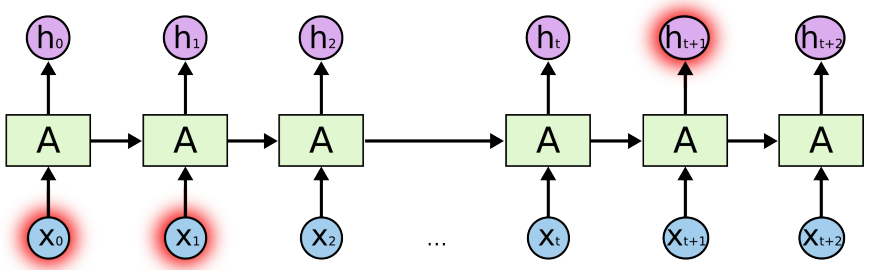
T = 4
D = 5
H = 6



WHY RNN IS DIFFICULT TRAINING

PROBLEM OF LONG TERM DEPENDENCIES





RNN has *short memory*: the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially:

$$h(1) = W_1 * \boldsymbol{x(1)} + W_2 * h(0)$$

$$h(2) = W_1 * x(2) + W_2 * h(1)$$

$$= W_1 * x(2) + W_2 * (W_1 * x(1) + W_2 * h(0))$$

$$= W_1 * x(2) + \boldsymbol{W_2 * W_1 * x(1)} + W_2^2 * h(0)$$

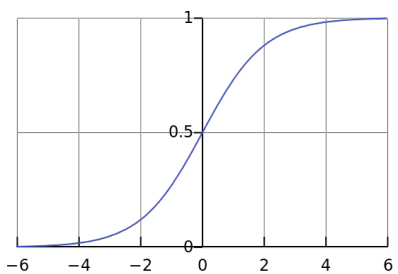
$$h(k) = W_1 * x(k) + W_2 * W_1 * x(k-1) + \ldots$$

$$+ \boldsymbol{W_2^{k-1} * W_1 * x(1)} + W_2^k * h(0)$$

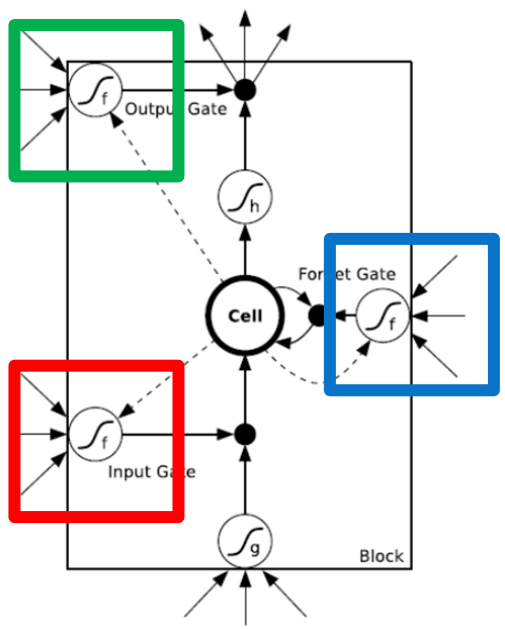
LONG SHORT-TERM MEMORY (LSTM)

- **Long** Short-Term Memory (LSTM) architecture was introduced to address RNN shor t memory. It has a n new basic element - *memory cell*

All gates use logistic sigmoid:
0 - gate closed,
1 - gate open



Output gate **"switch-off"** output of the cell

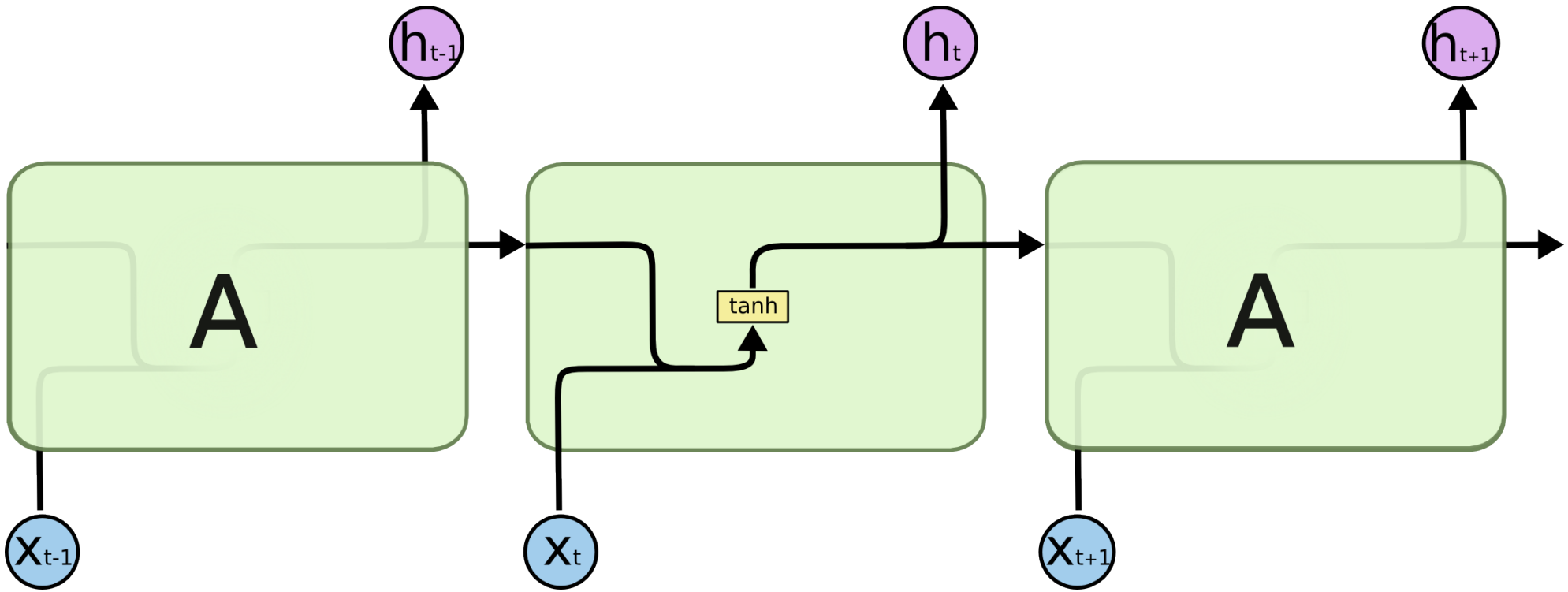


Forget gate reset the cell's state

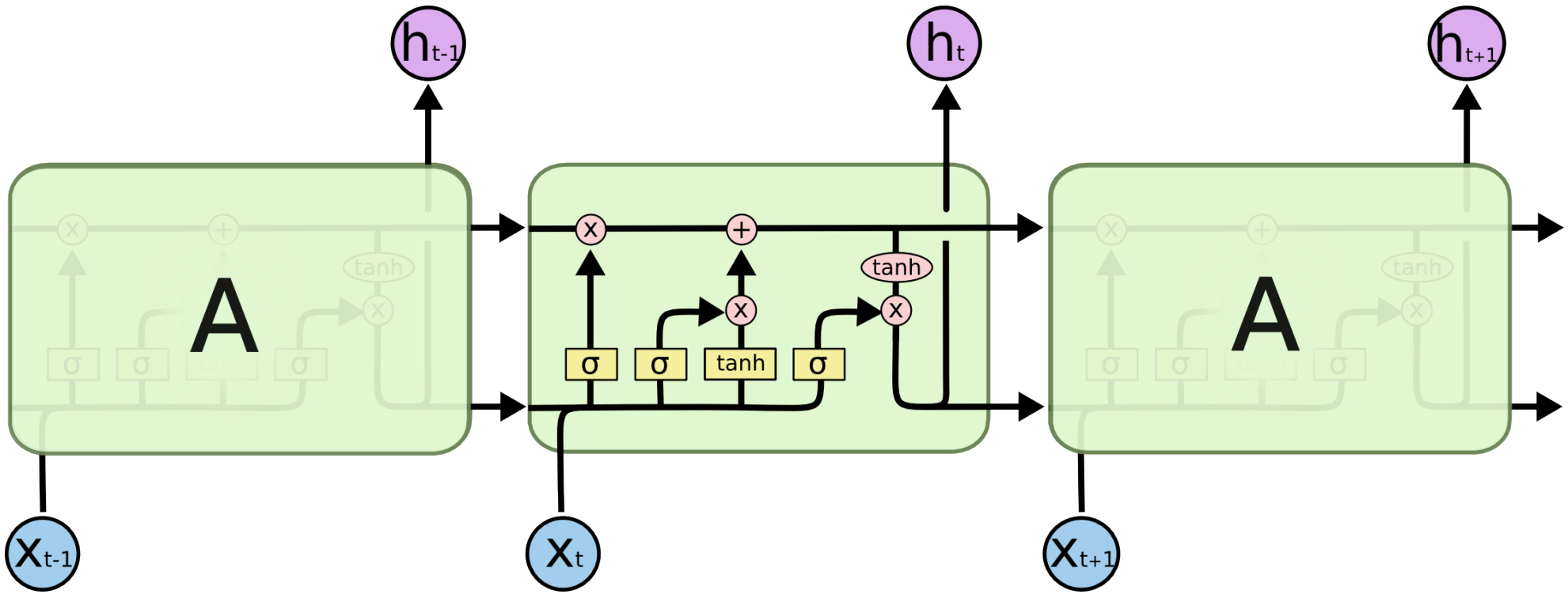
Input gate cut-off the input of the cell

Hochreiter and Schmidhuber,1997.

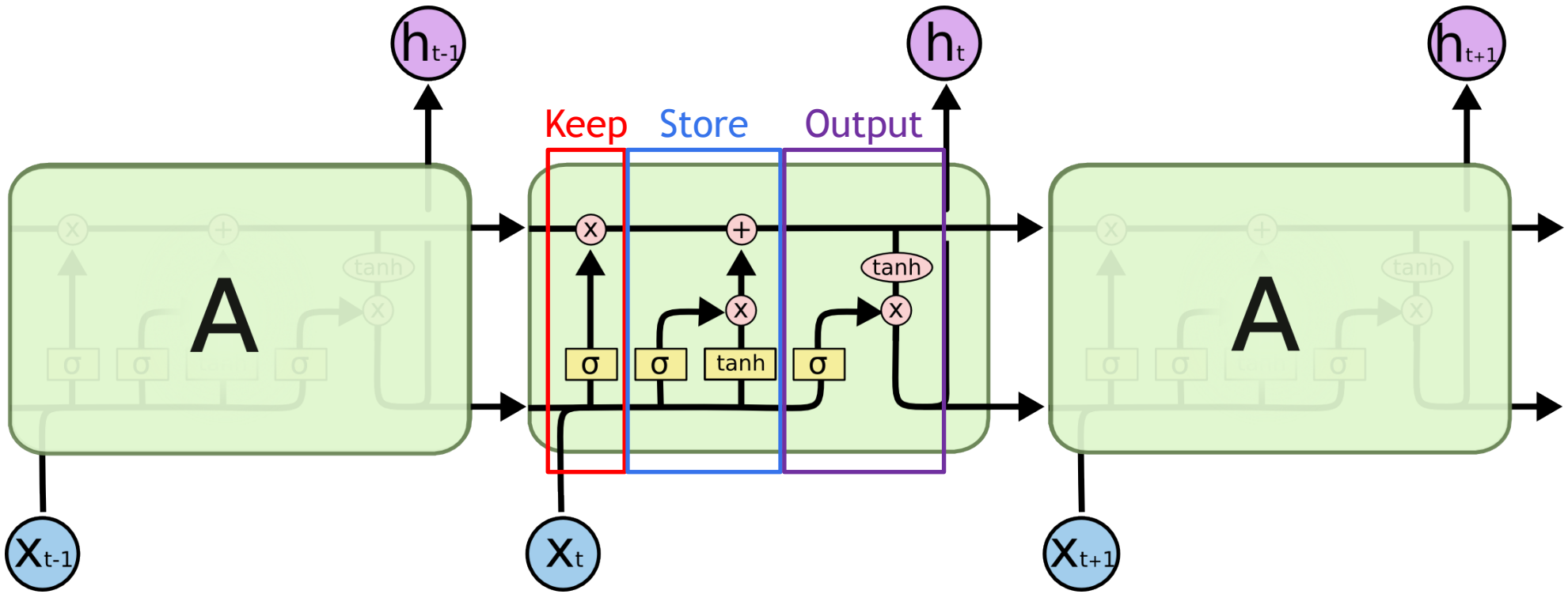
RNN SEQUENCE REPRESENTATION



LSTM REPRESENTATION

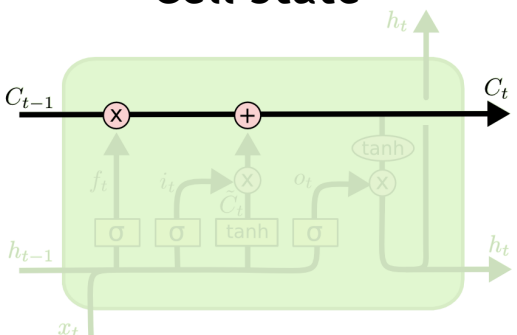


LSTM REPRESENTATION

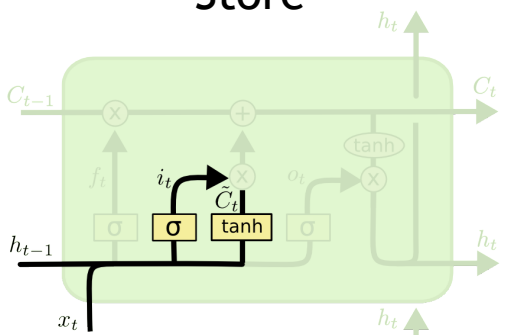


LSTM OPERATION

Cell State

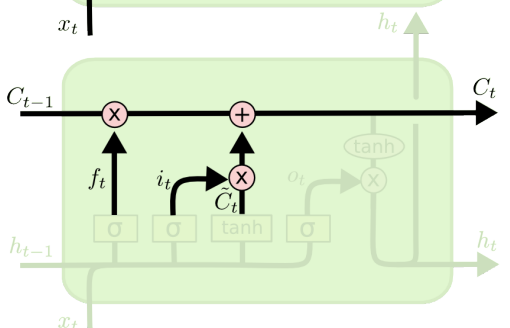


Store



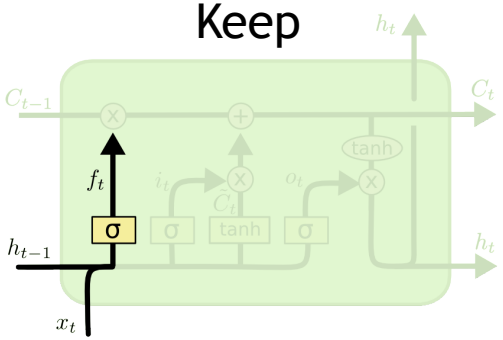
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



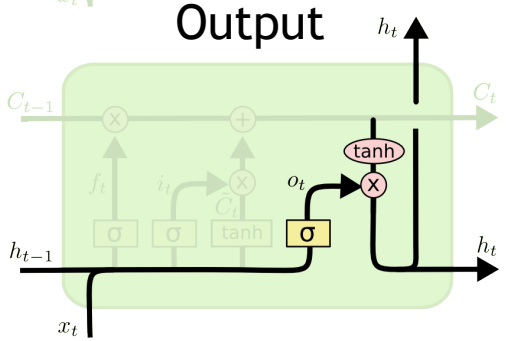
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Keep



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

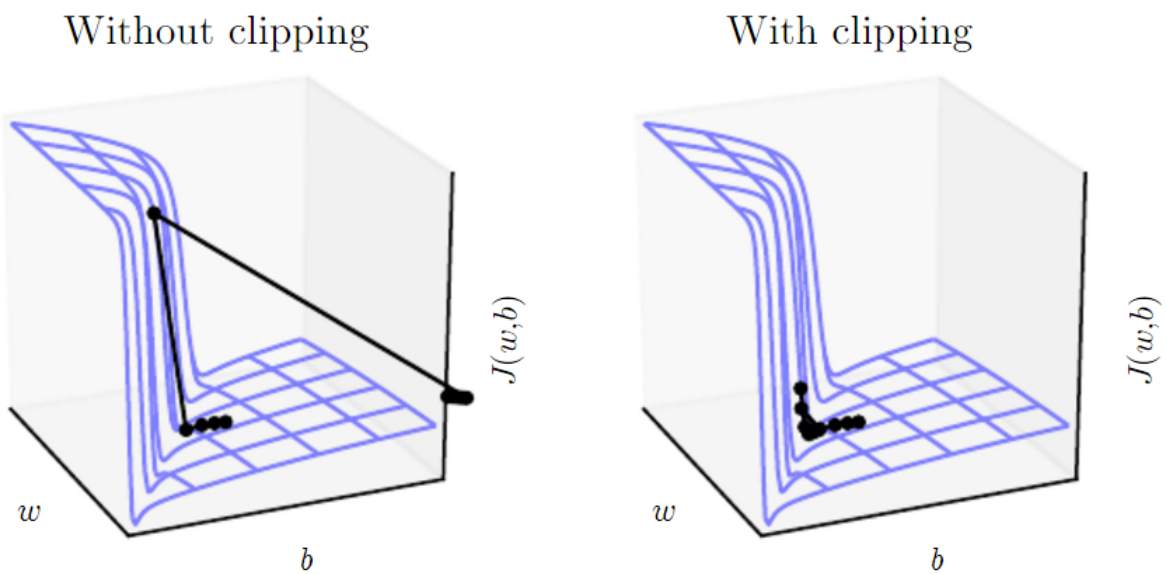
Output



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

CLIPPING GRADIENT



strongly nonlinear functions such as those computedby a recurrent net over many time steps tend to have derivatives that can be either very large or very small in magnitude.

Source:deeplearingbook

CHALLENGE

Change the model type:

```
opt.model_type = 'lstm'
```

Change regularization factors:

```
opt.batch_norm = 1
opt.dropout = 0.2
```

Change other hyper-parameters:

```
rnn_size = 128
num_layers = 2

vocab_size = 256
wordvec_dim = 64
```

```
opt.model_type = 'rnn'
opt.wordvec_size = 64
opt.rnn_size = 128
opt.num_layers = 2
opt.dropout = 0
opt.batchnorm = 0
opt.cudnn = 1
```

RELATED TOPICS

Andrej Karpathy blog - https://github.com/karpathy/char-rnn

Neural Talk

RNN(Recurrent Neural Network)과 Torch로 발라드곡 작사하기
https://github.com/socurites/char-rnn-korean

자동 Jazz 악보 생성 / 메탈리카 음악 생성

주가 예측?? – stocksneural – 어렵지 않을까...

awesome-rnn

NVIDIA | DEEP LEARNING INSTITUTE

www.nvidia.com/dli