Deep Learning in a nutshell: Sequence Learning

Xu Wang

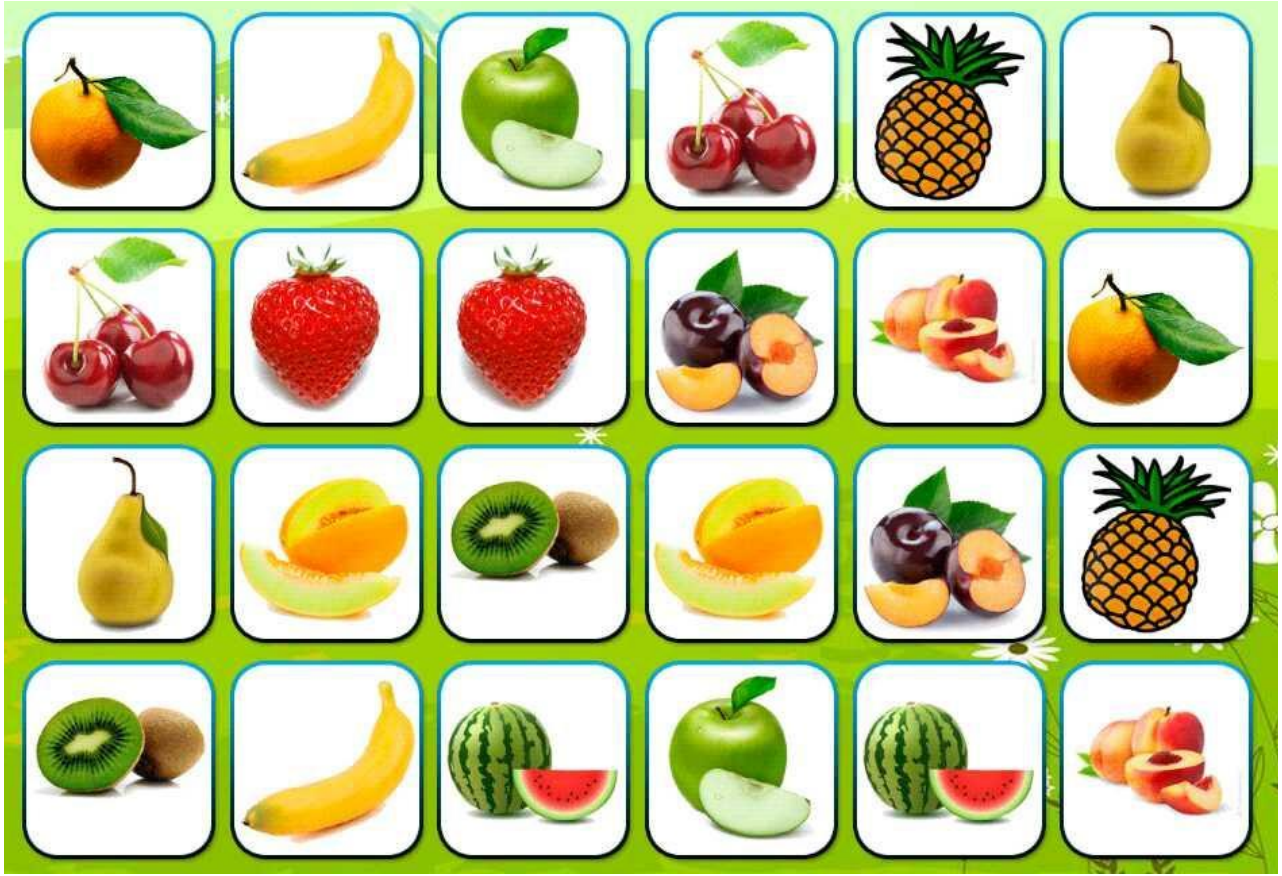
What is Sequence Data?

- Time series
- Video
- language
- ...

How to Process Sequence Data?

- ARIMA
- Markov chain
- ...
- Deep Learning

General idea

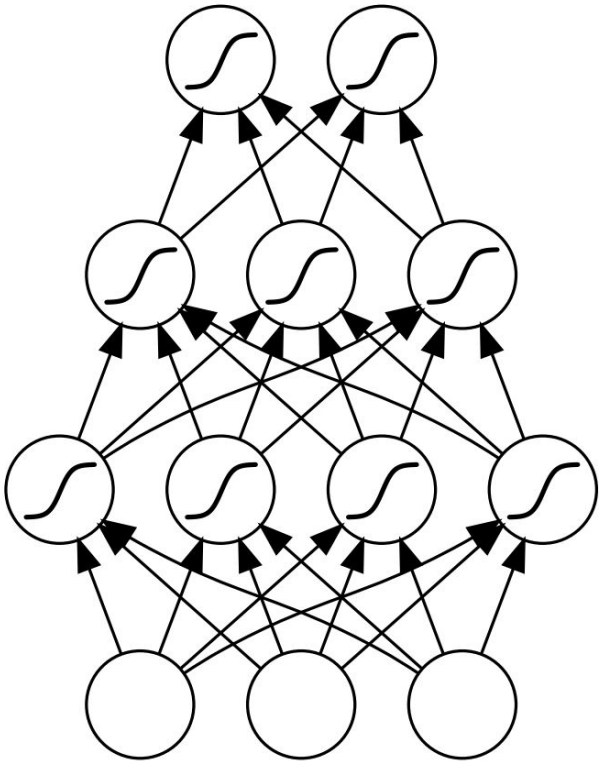


ConvNet
(or CNN)

General idea

ConvNet
(or CNN)

Multilayer Perceptron



Output Layer

Hidden Layers

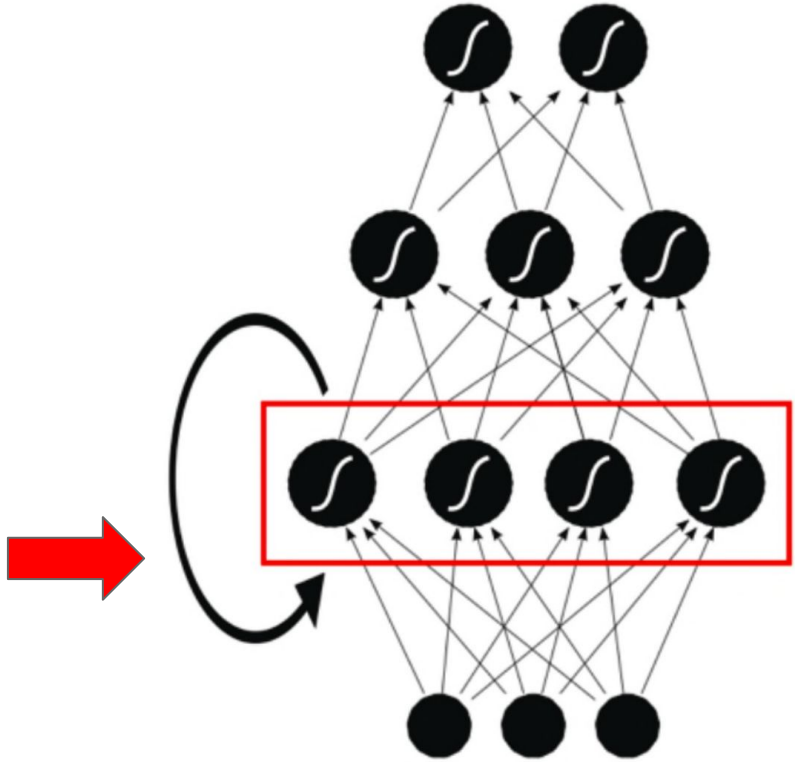
Input Layer

The output depends ONLY on the current input.

Alex Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”

Recurrent Neural Network (RNN)

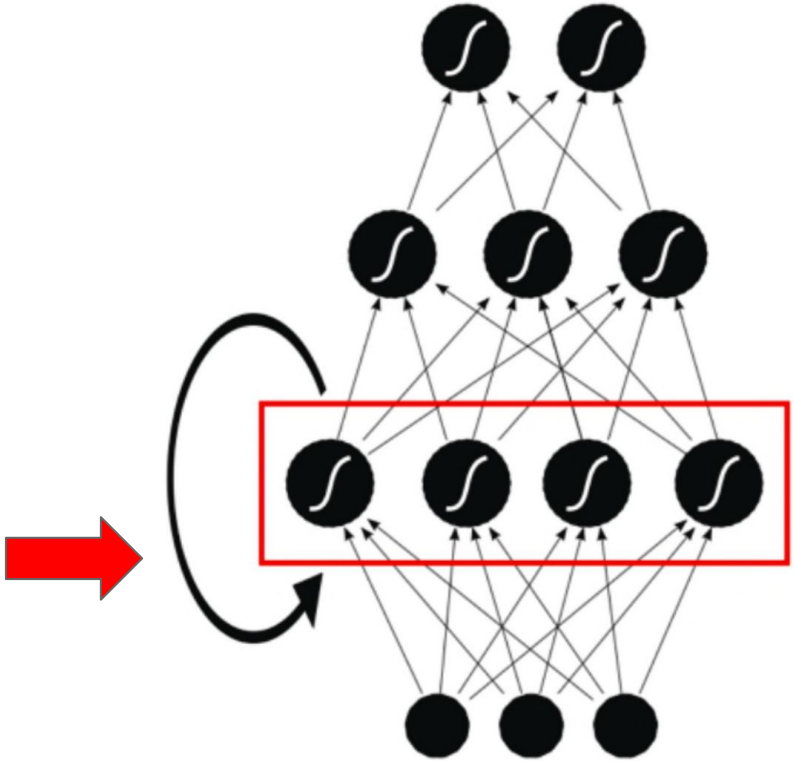
The hidden layers and the output depend from previous states of the hidden layers



Alex Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”

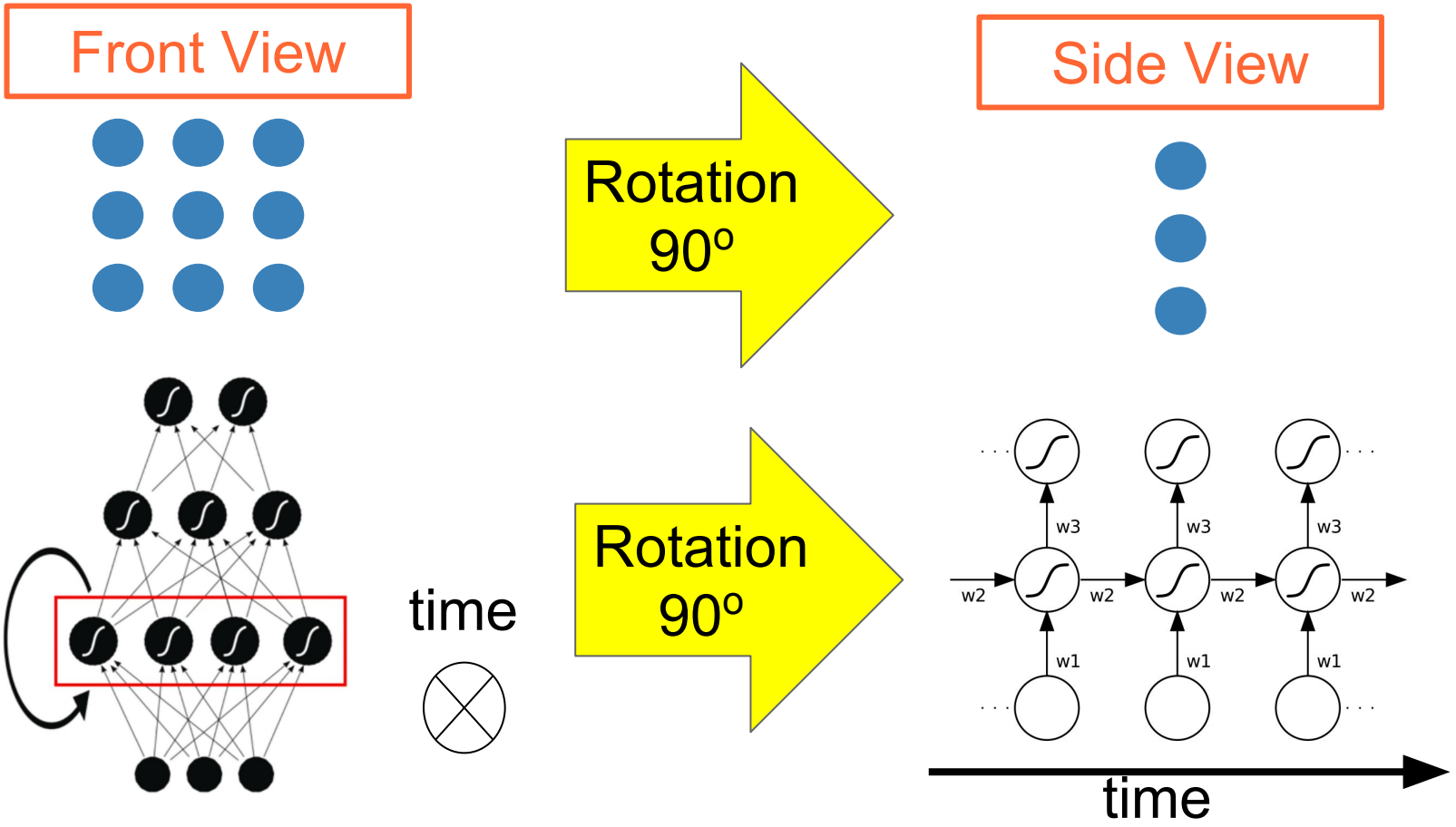
Recurrent Neural Network (RNN)

The hidden layers and the output depend from previous states of the hidden layers

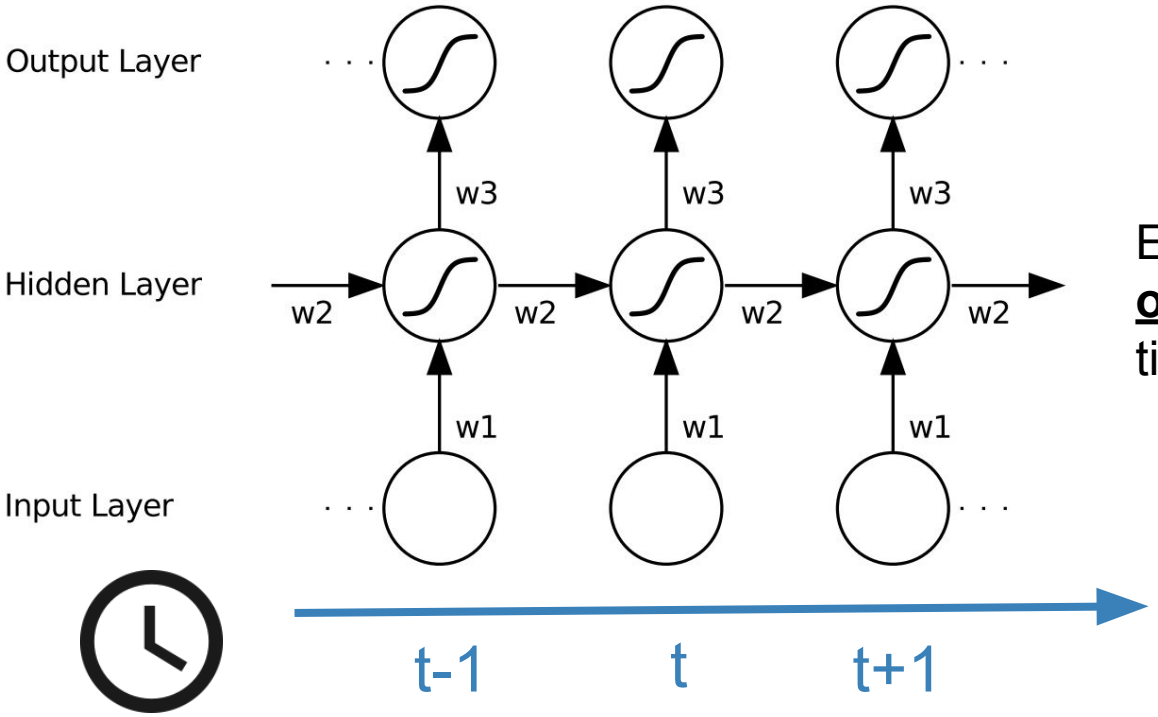


Alex Graves, [“Supervised Sequence Labelling with Recurrent Neural Networks”]

Recurrent Neural Network (RNN)



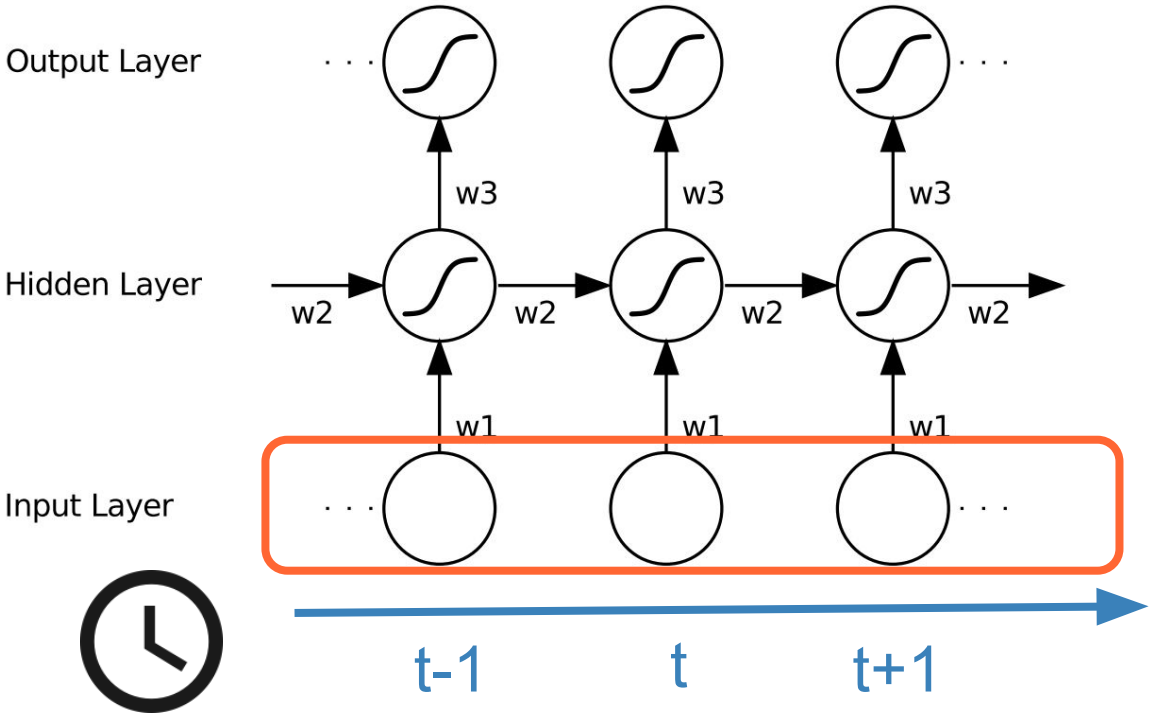
Recurrent Neural Networks (RNN)



Each node represents **a layer of neurons** at a single timestep.

Alex Graves, "Supervised Sequence Labelling with Recurrent Neural Networks"

Recurrent Neural Networks (RNN)

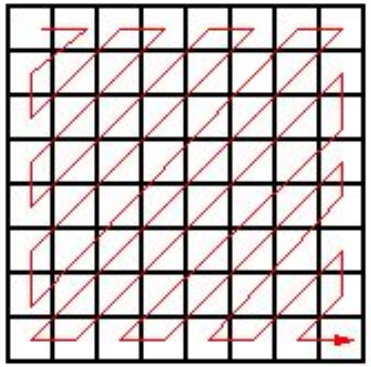


The input is a **SEQUENCE x(t)** of any length.

Alex Graves, "Supervised Sequence Labelling with Recurrent Neural Networks"

Recurrent Neural Networks (RNN)

Common visual sequences:



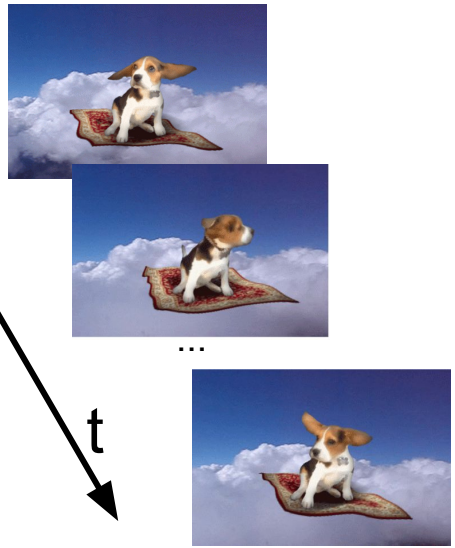
The input is a **SEQUENCE** **x(t)** of any length.

Still image + Spatial scan (zigzag, snake)

Recurrent Neural Networks (RNN)

Common visual sequences:

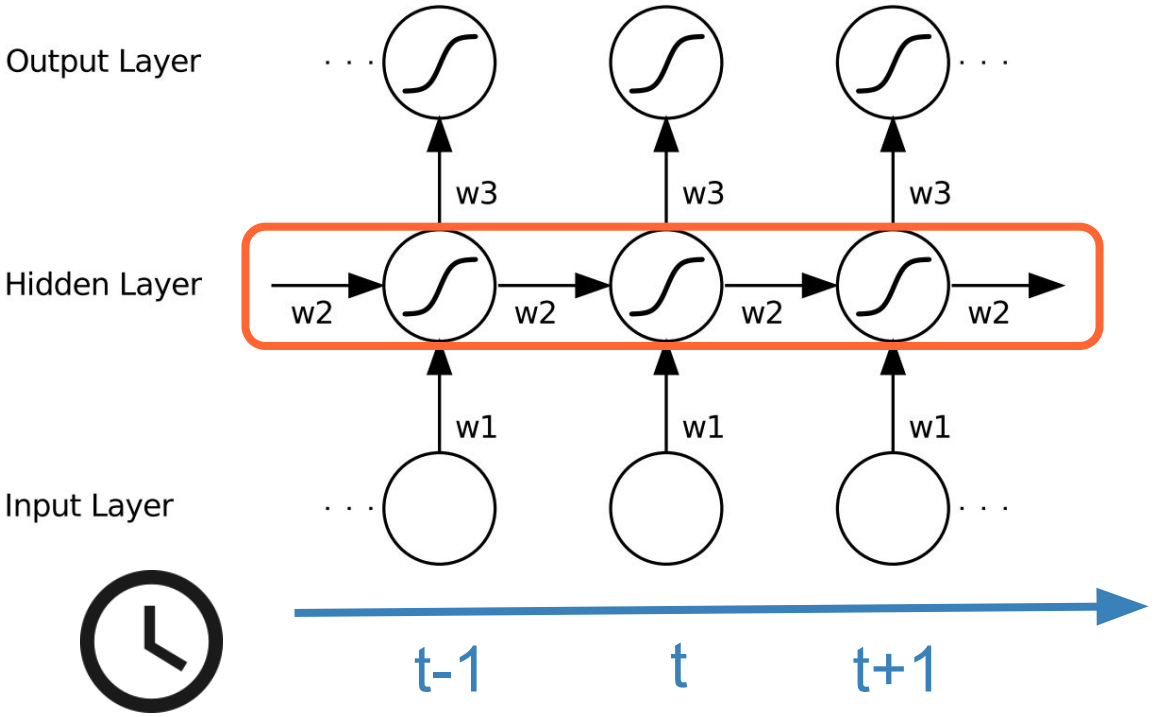
Video



Temporal sampling

The input is a **SEQUENCE x(t)** of any length.

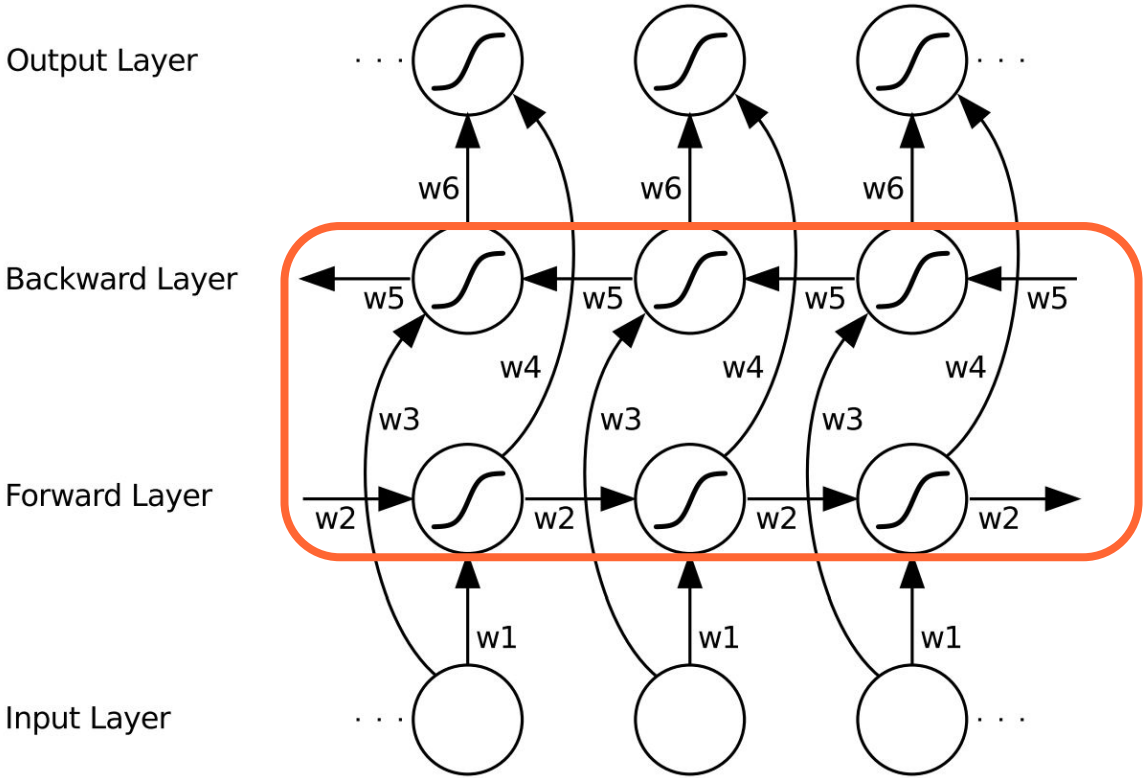
Recurrent Neural Networks (RNN)



Must learn temporally shared weights w2; in addition to w1 & w3.

Alex Graves, "Supervised Sequence Labelling with Recurrent Neural Networks"

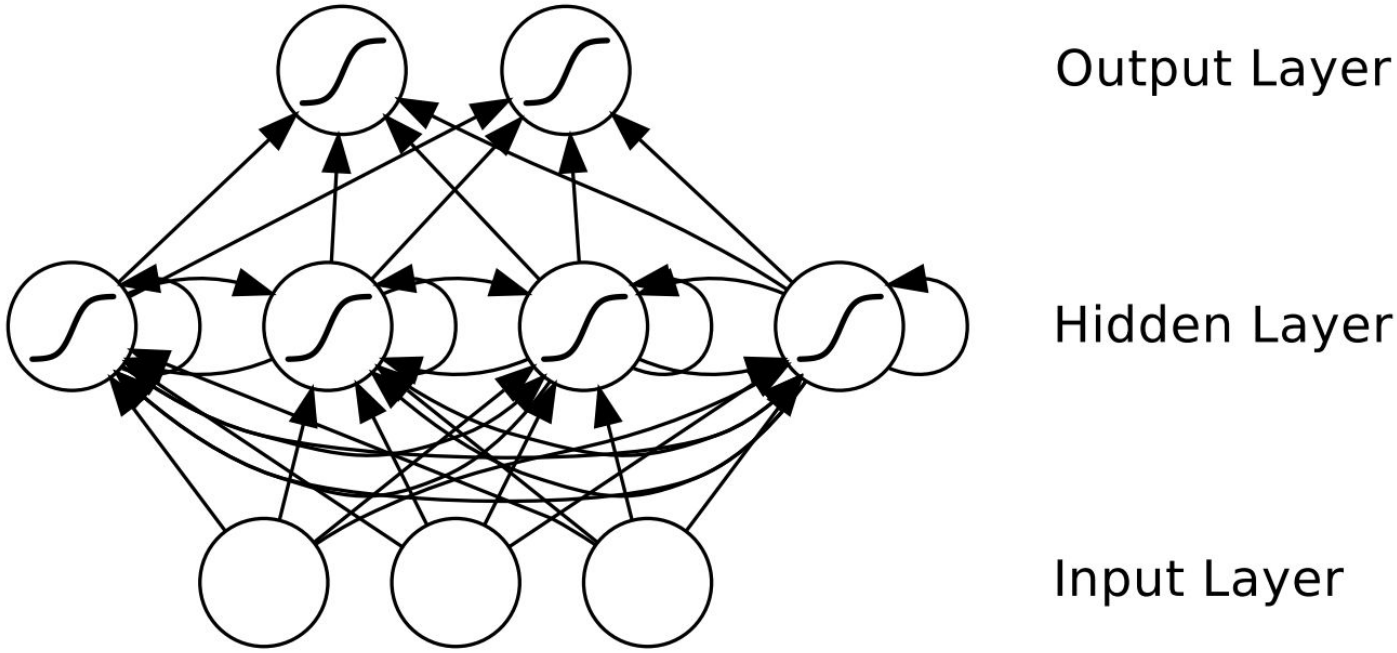
Bidirectional RNN (BRNN)



Must learn weights w2, w3, w4 & w5; in addition to w1 & w6.

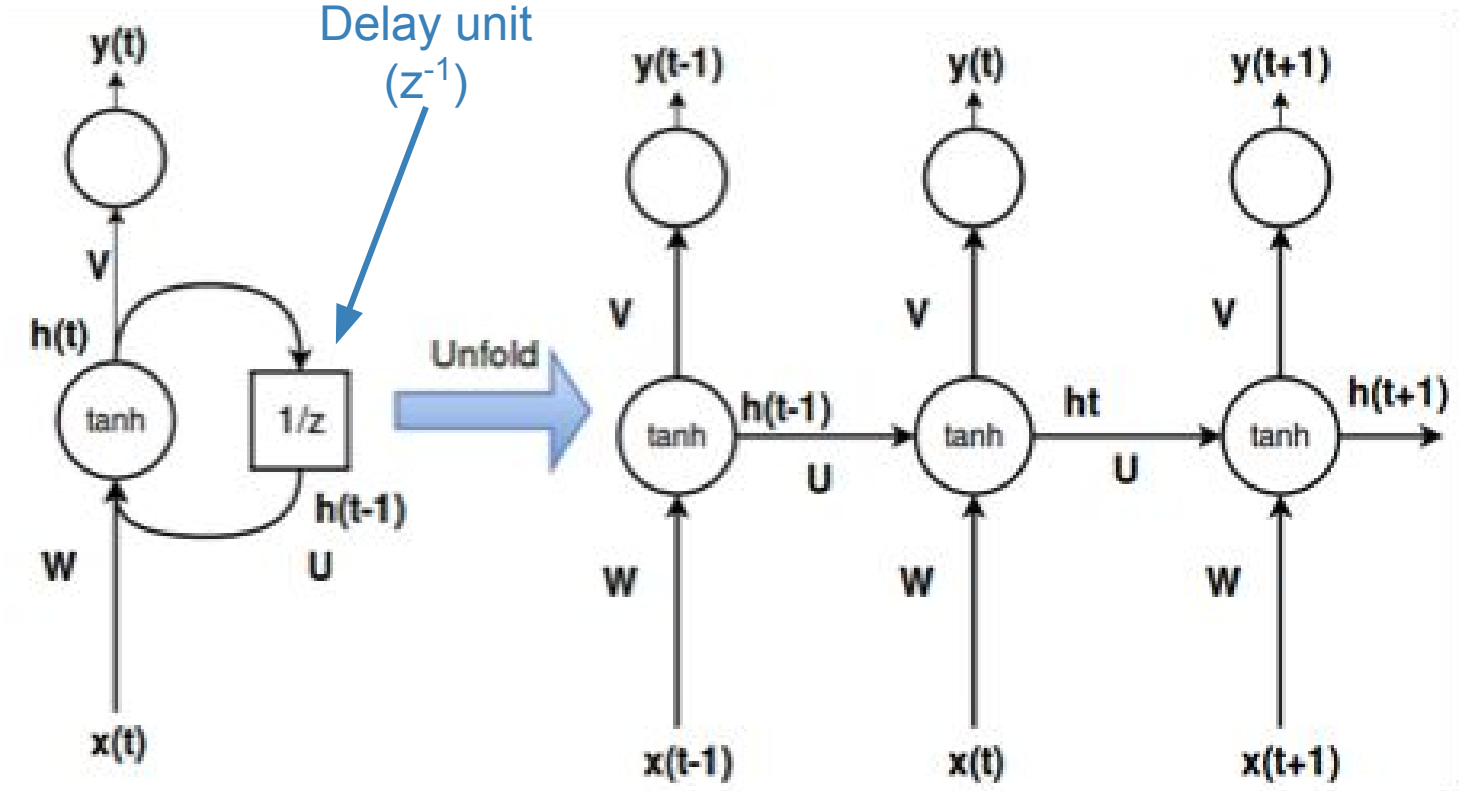
Alex Graves, [“Supervised Sequence Labelling with Recurrent Neural Networks”]

Bidirectional RNN (BRNN)



Alex Graves, [“Supervised Sequence Labelling with Recurrent Neural Networks”]

Formulation: One hidden layer



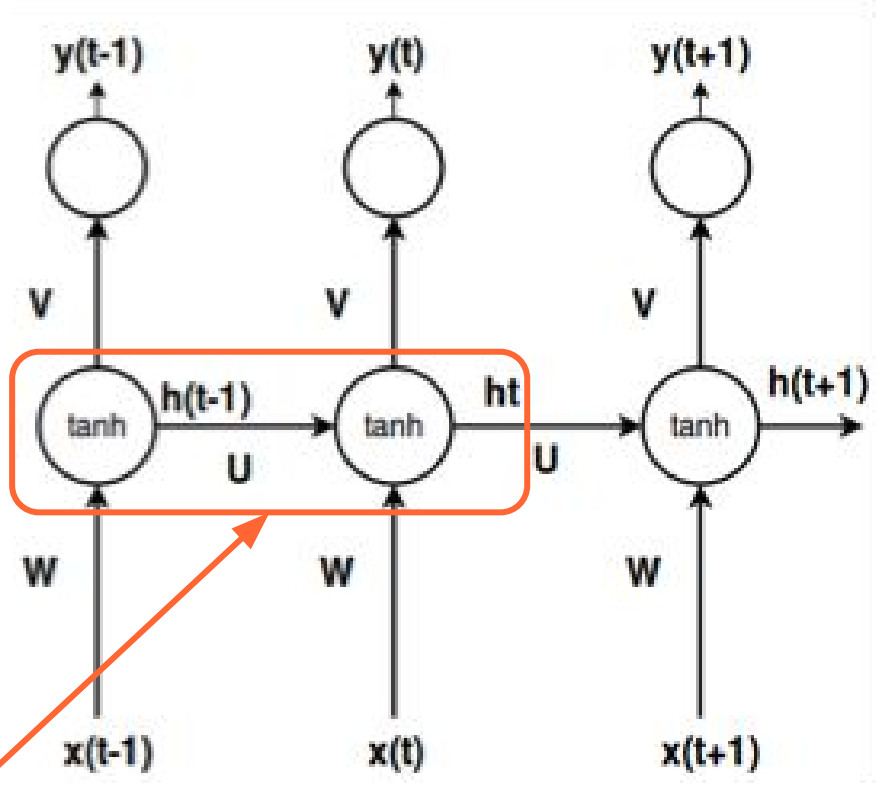
Slide: Santi Pascual

Formulation: Single recurrence

$$\mathbf{y}_t = f(\mathbf{V} \cdot \mathbf{h}_t + \mathbf{b}_v)$$

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{x}_t \in \mathbb{R}^n$$



One-time Recurrence

Slide: Santi Pascual

Formulation: Multiple recurrences

One time-step recurrence

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

Recurrence

T time steps recurrences

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot g(\cdots g(\mathbf{W} \cdot \mathbf{x}_{t-T} + \mathbf{U} \cdot \mathbf{h}_{t-T} + \mathbf{b}_h) \cdots) + \mathbf{b}_h)$$

Slide: Santi Pascual

RNN problems

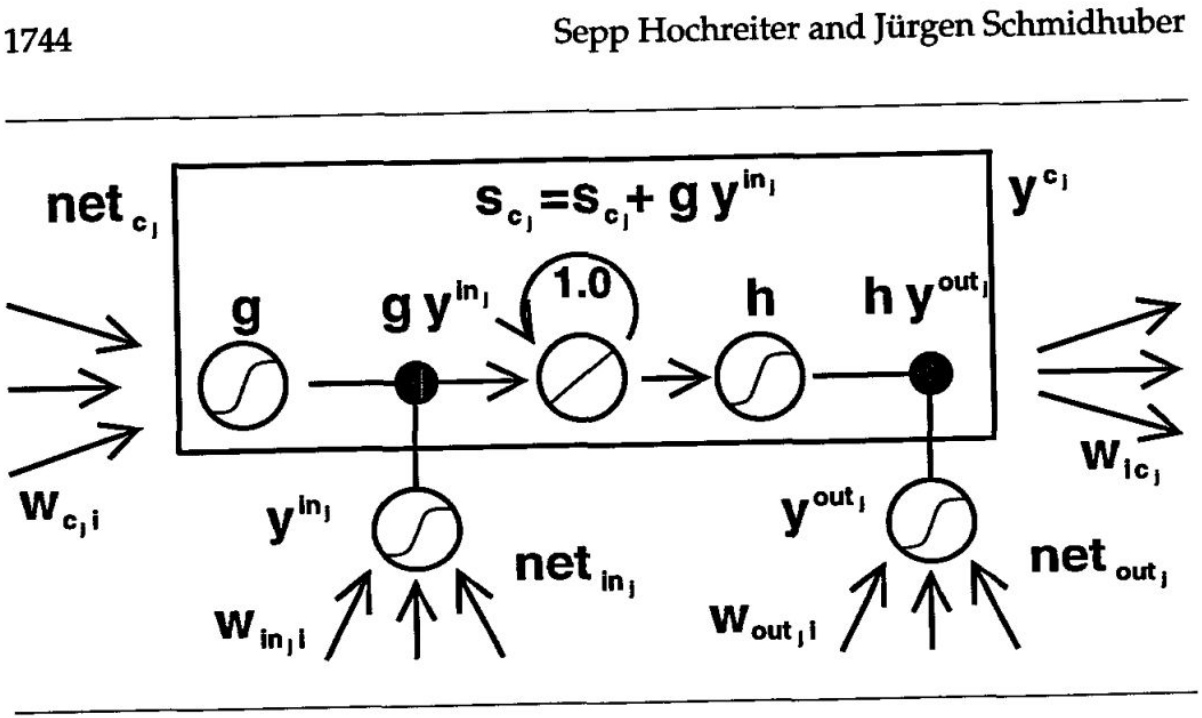
Long term memory vanishes because of the T nested multiplications by U.

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot g(\cdots g(\mathbf{W} \cdot \mathbf{x}_{t-T} + \mathbf{U} \cdot \mathbf{h}_{t-T} + \mathbf{b}_h) \cdots) + \mathbf{b}_h)$$

During training, gradients may explode or vanish because of temporal depth.

Slide: Santi Pascual

Long Short-Term Memory (LSTM)



Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

Long Short-Term Memory (LSTM)

Based on a standard RNN whose neuron activates with *tanh*...

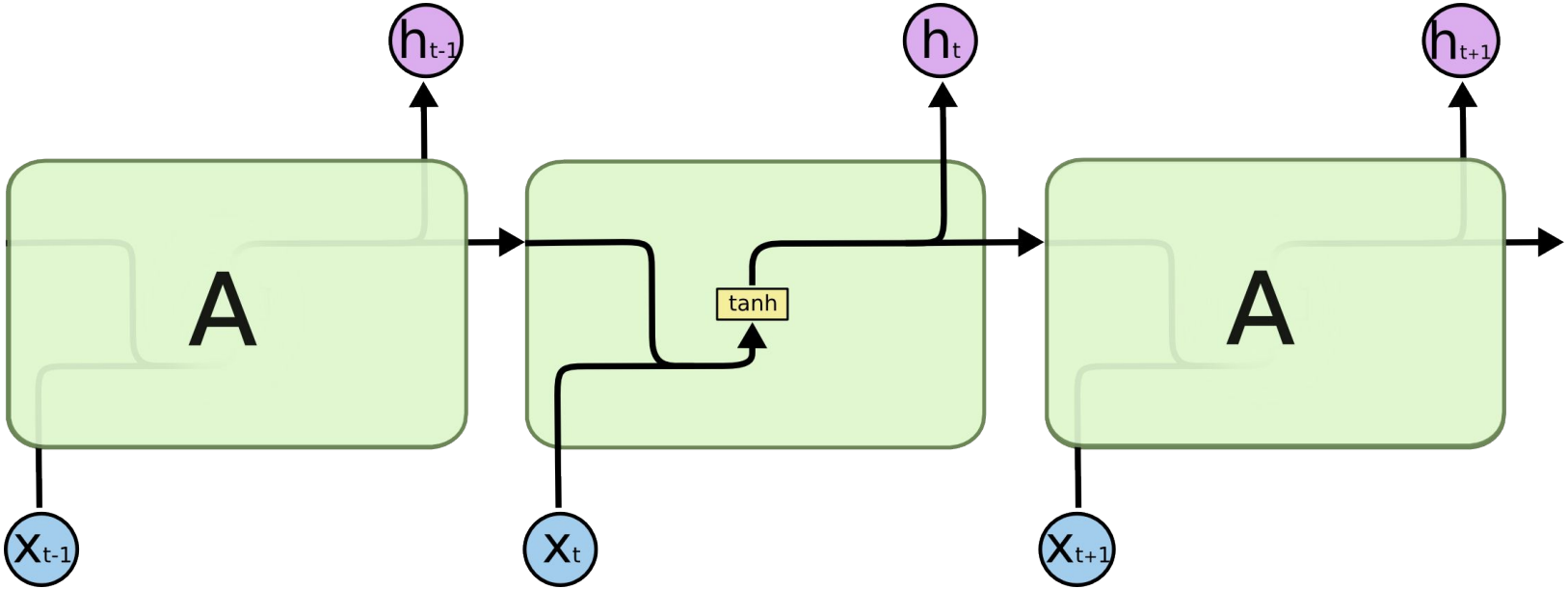


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long Short-Term Memory (LSTM)

$\mathbf{C_t}$ is the cell state, which flows through the entire chain...

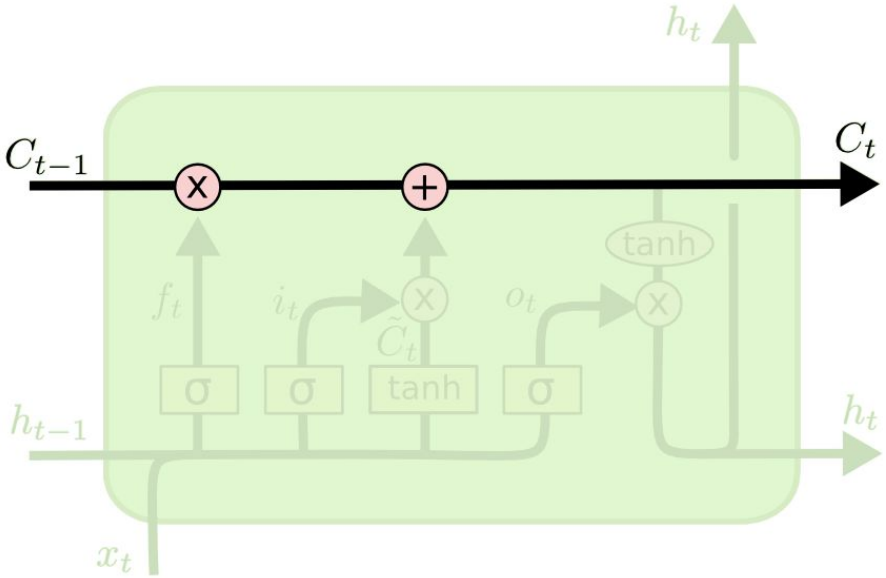


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long Short-Term Memory (LSTM)

...and is updated with a **sum** instead of a product. This avoid memory vanishing and exploding/vanishing backprop gradients.

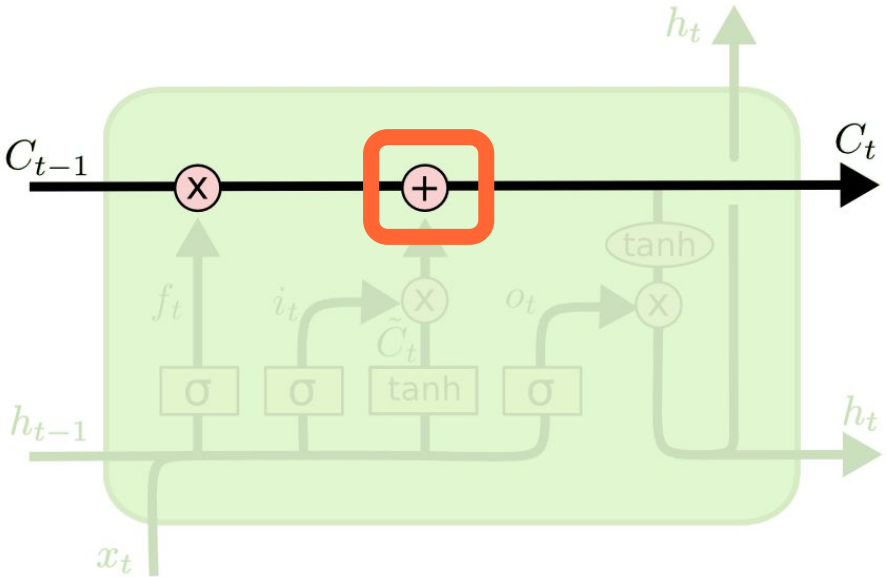


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long Short-Term Memory (LSTM)

Three **gates** are governed by *sigmoid* units (btw [0,1]) define the control of in & out information..

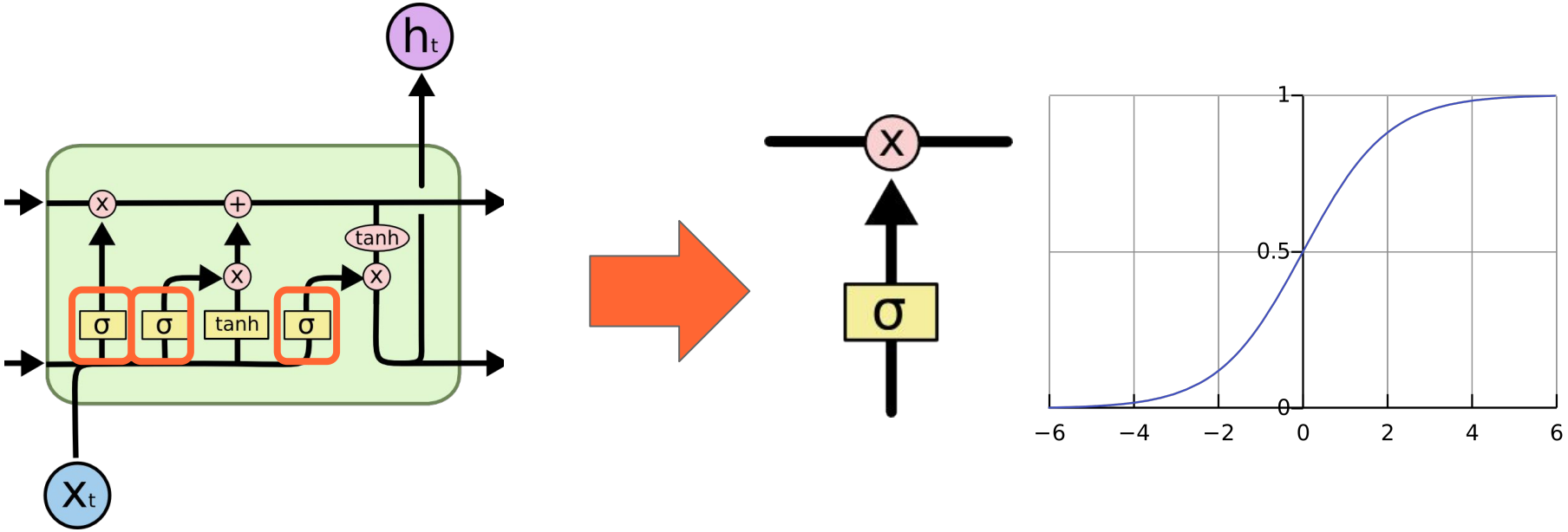
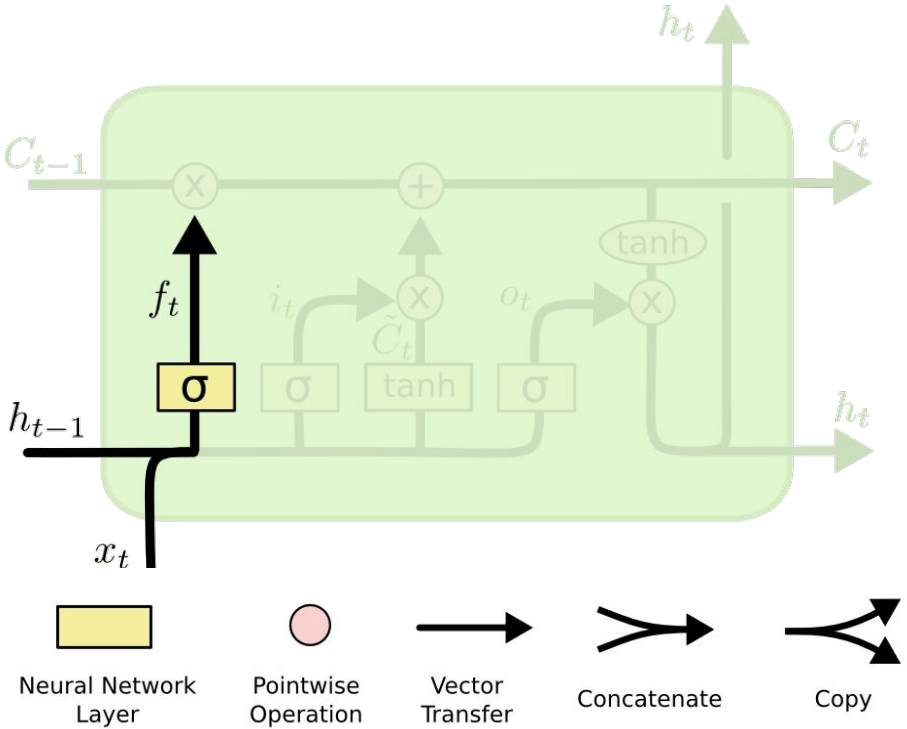


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long Short-Term Memory (LSTM)



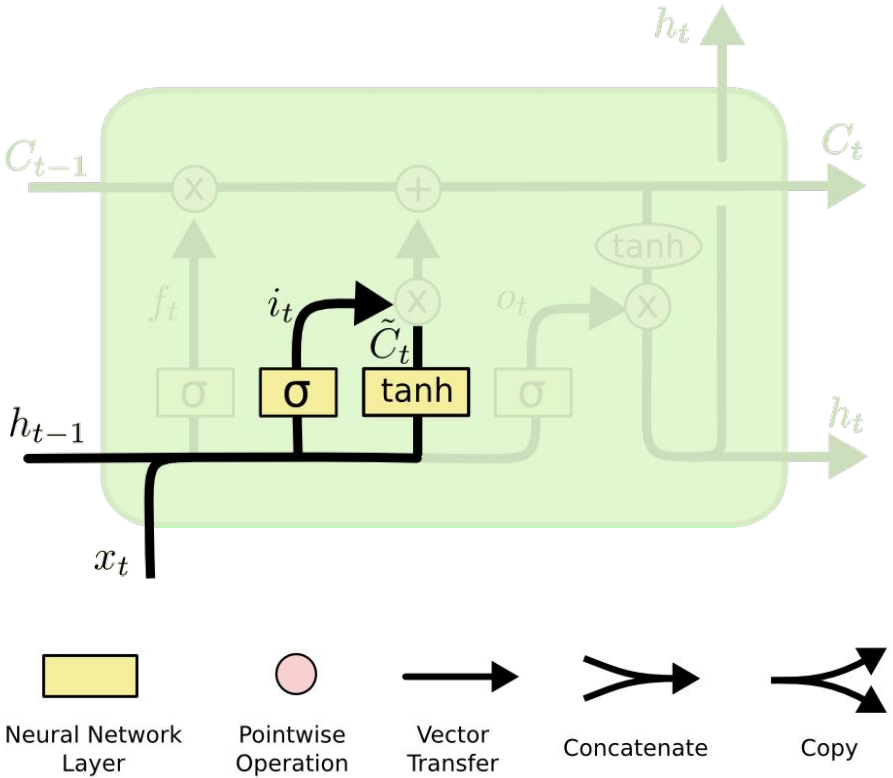
Forget Gate:

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

Concatenate

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)



Input Gate Layer

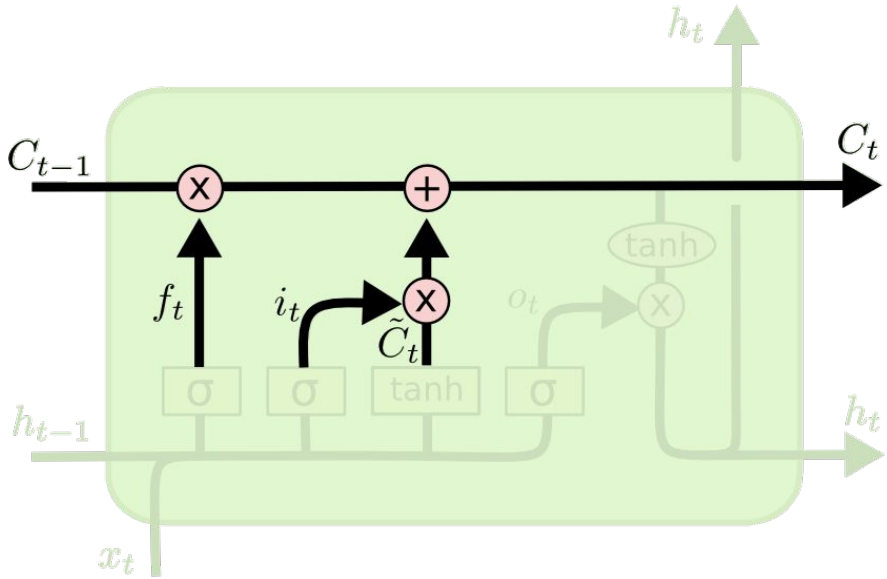
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

New contribution to cell state

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)

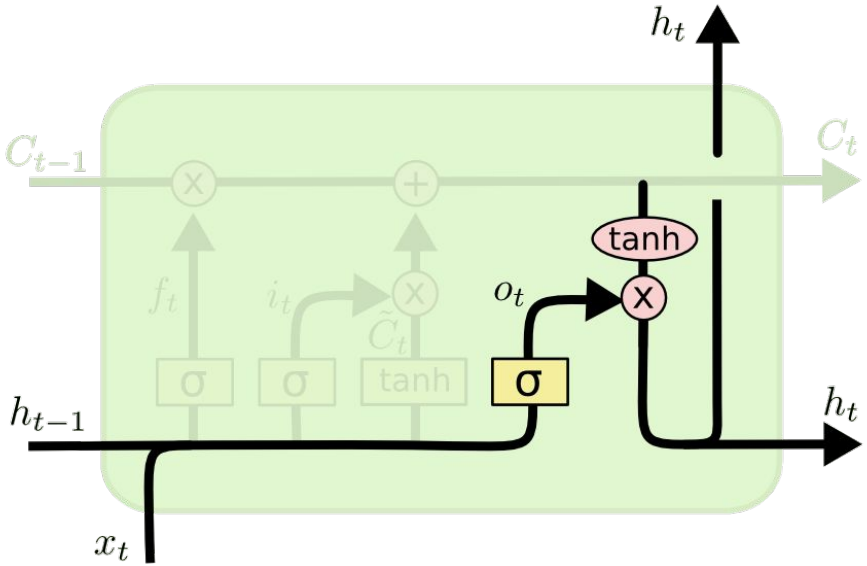


Update Cell State (memory):

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)



Output Gate Layer

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

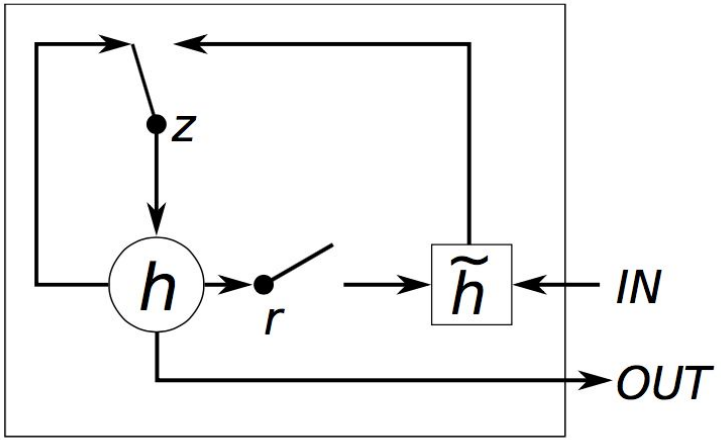
Output to next layer

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Gated Recurrent Unit (GRU)

Similar performance as LSTM with less computation.



$$u_i = \sigma\left(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}\right) \quad (1)$$

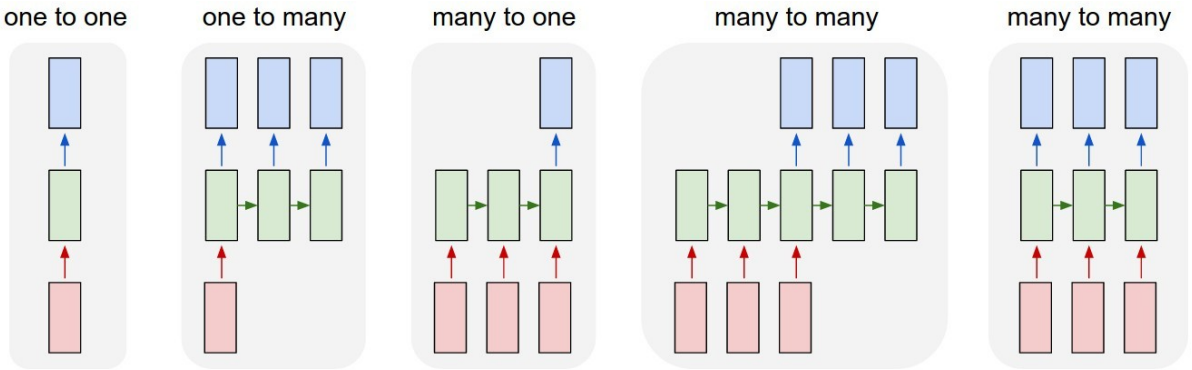
$$r_i = \sigma\left(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}\right) \quad (2)$$

$$\tilde{h}_i = \tanh\left(Wx_i + r_i \circ Uh_{i-1} + b^{(h)}\right) \quad (3)$$

$$h_i = u_i \circ \tilde{h}_i + (1 - u_i) \circ h_{i-1} \quad (4)$$

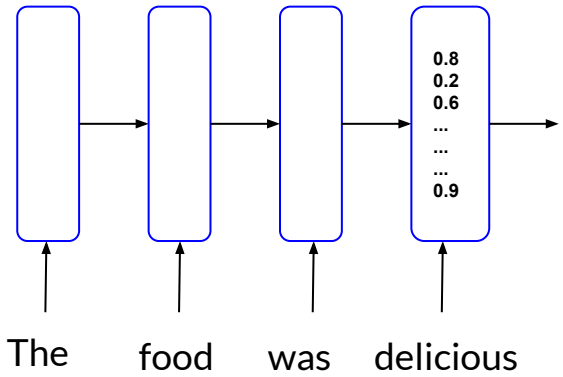
Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

Sequence to Sequence Learning



RNN to Encode a Sequence to a Fixed-sized Vector

- Encode a sequence of word vectors into a fixed-sized context dependent vector
- Application: Sentiment Analysis

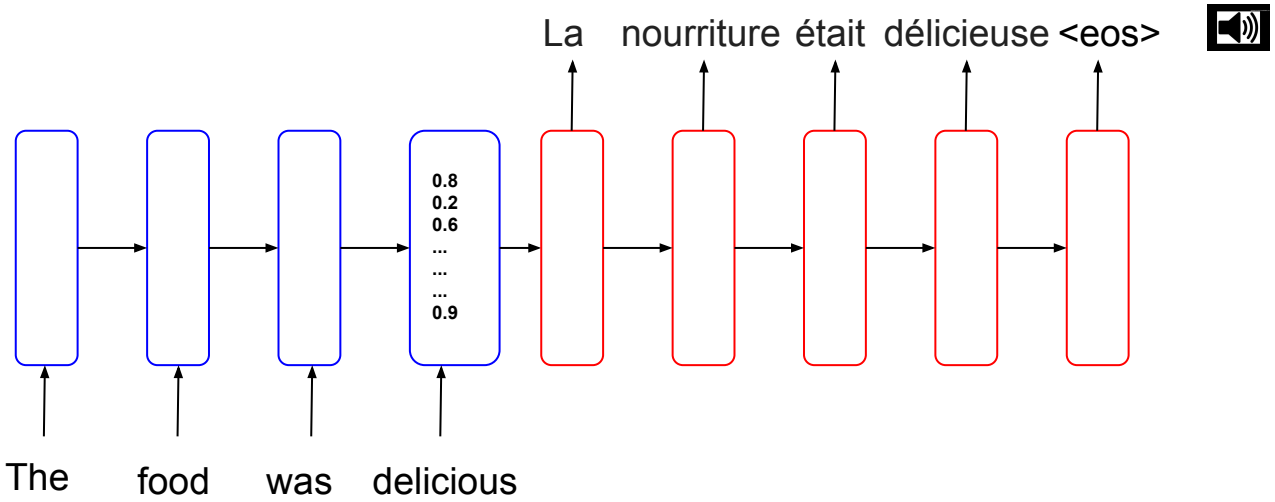


P(y = positive | x = The food was delicious)

Encoder - Decoder (Seq2Seq) Model

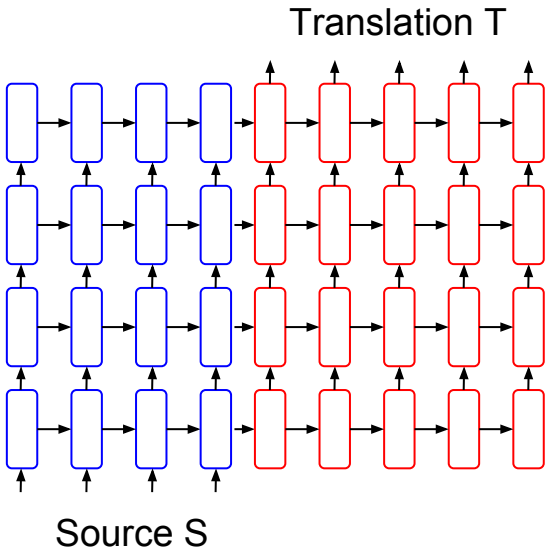
Encoder - Decoder LSTM Model

- Encoder LSTM encodes a sequence of word vector into a fixed sized context vector
- Use that vector as initial state to produce a sentence using the decoder LSTM



Experiments

- Dataset: WMT' 14 English to French dataset
- Model: 4 LSTM layers in both encoder and decoder
- Objective: Maximize the log probability of a correct translation T given a source sentence S
- Classification (Softmax over 80000 words) in every decoder's output time step

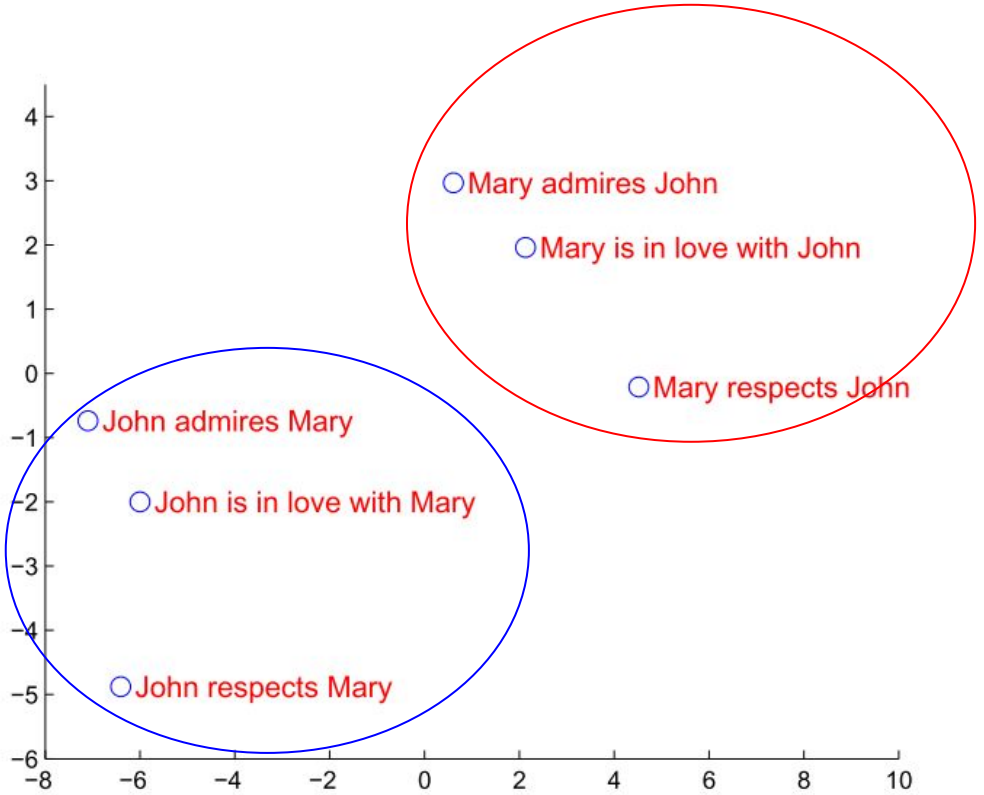


$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

Model Analysis

- Visualized PCA1 and PCA2 of the hidden vector encoded by LSTM encoder

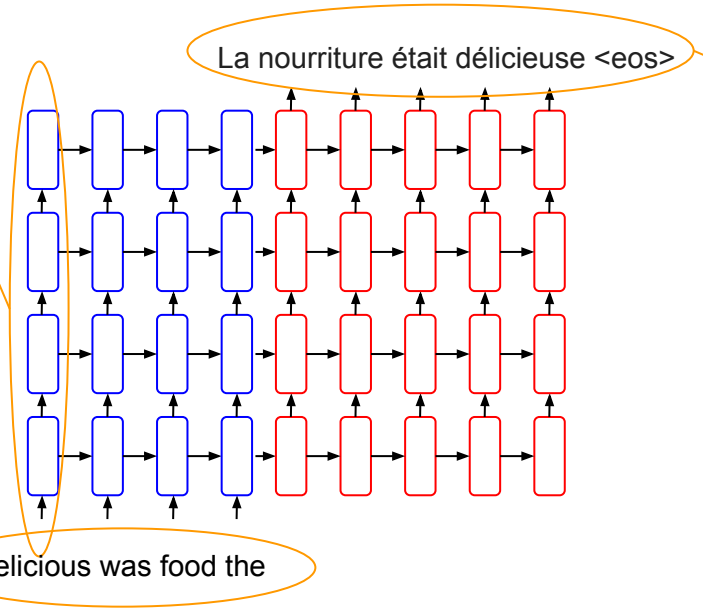
- The encoded context vector capture the semantics of the sentence



Pro 1: Lots of Tricks to Improve Performance

Deep LSTMs (Each additional layer reduced perplexity by nearly 10%)

Reversing the source sentence (Test Perplexity: 5.8 -> 4.7, Test BLEU: 25.9 -> 30.6)



Beam search decoder

Ensemble (34.81 BLEU score)

Pro 3: Applicable for Various Sequence to Sequence Tasks

Task	Source	Target
Machine translation	Source language	Translated language
Chatbot	Source conversation	Response conversation
Text Summarization	Article	Summary
Machine Reading Comprehension	Embedded Passage-Question vector	Answer

Pro 4: End-to-end Neural Network Model

- Train word embeddings along with encoder-decoder LSTMs

Source sequence → 

→ Target sequence

Con 1: Need better results comparison

	Seq2Seq (Le 2014)	RNN Encoder–Decoder (Cho 2014)
Encoder	LSTM	RNN
Decoder	LSTM	RNN
Vocabulary	160000 (10x more)	15000
Word Embeddings Dimension	1000 (10x more)	100
Training Time	10 days (3x more)	3 days
BLEU (ntst2014)	36.5	34.54

Handle Spatial Temporal Data?

- Convolution + LSTM

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

LSTM

Input & state are 1D vectors

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

ConvLSTM

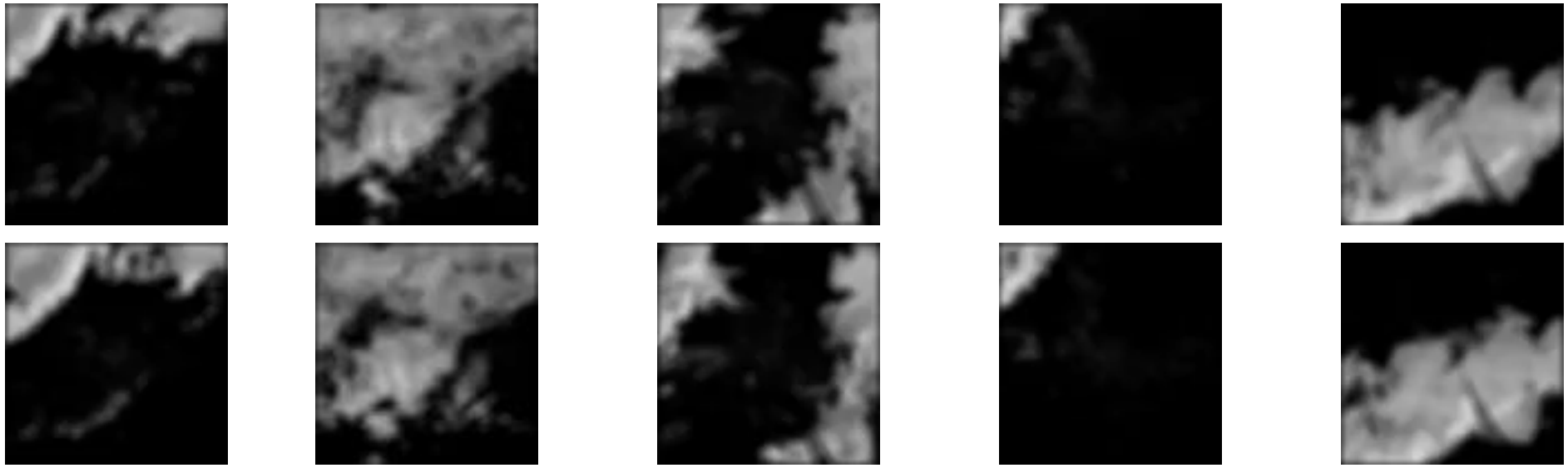
Input &state are 3D tensors. Convolution is used for both input-to-state and state-to-state connection.
Use Hadamard product to keep the constant error carousel property of cells.

Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting
卷积LSTM网络:利用机器学习预测短期降雨

施行健
香港科技大学
VALSE 2016/03/23

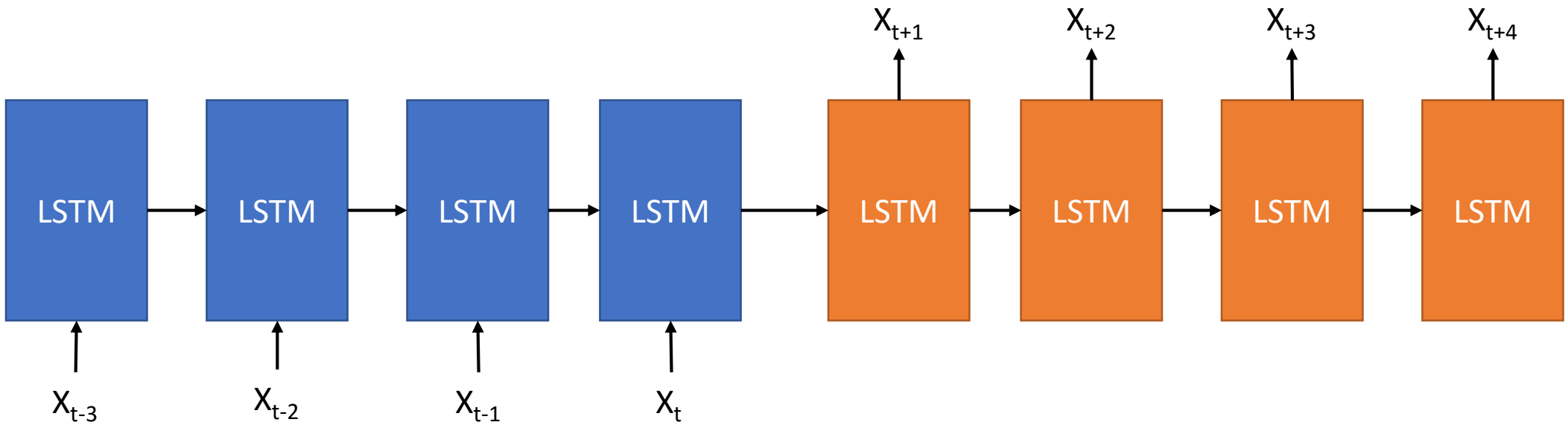
Goal of Precipitation Nowcasting

- Give precise and timely prediction of rainfall intensity in a local region over a relatively short period of time (e.g., 0-6 hours)
  - High resolution & Accurate timing
  - High dimensional spatiotemporal data



How to deal with spatiotemporal data?

- A pure Encoding-Forecasting structure is not enough
- We are dealing with spatiotemporal data!



NOT ENOUGH!!!

How to deal with spatiotemporal data?

- We need to design a specific network structure for spatiotemporal data
- What's the characteristics of the spatiotemporal data we are dealing with?
  - Strong correlation between local neighbors, i.e, neighbors tend to act similarly
- Fully-connected LSTM (FC-LSTM) → Convolutional LSTM (ConvLSTM)
  - Regularize the network by specifying the structure
  - Use convolution instead of fully-connection in state-to-state transition!

Comparison between FC-LSTM & ConvLSTM

FC-LSTM

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

Input & state at a timestamp are 1D vectors. Dimensions of the state can be permuted without affecting the overall structure.

ConvLSTM

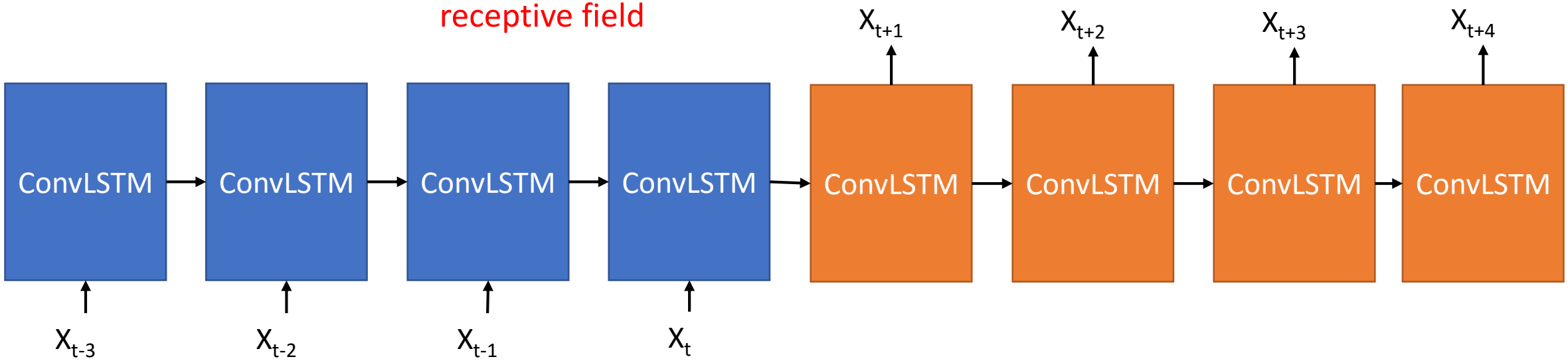
$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

Input & state at a timestamp are 3D tensors. Convolution is used for both input-to-state and state-to-state connection.

Use Hadamard product to keep the constant error carousel (CEC) property of cells

Convolutional LSTM

With the help of convolutional recurrence, the final state has large receptive field



Final Structure

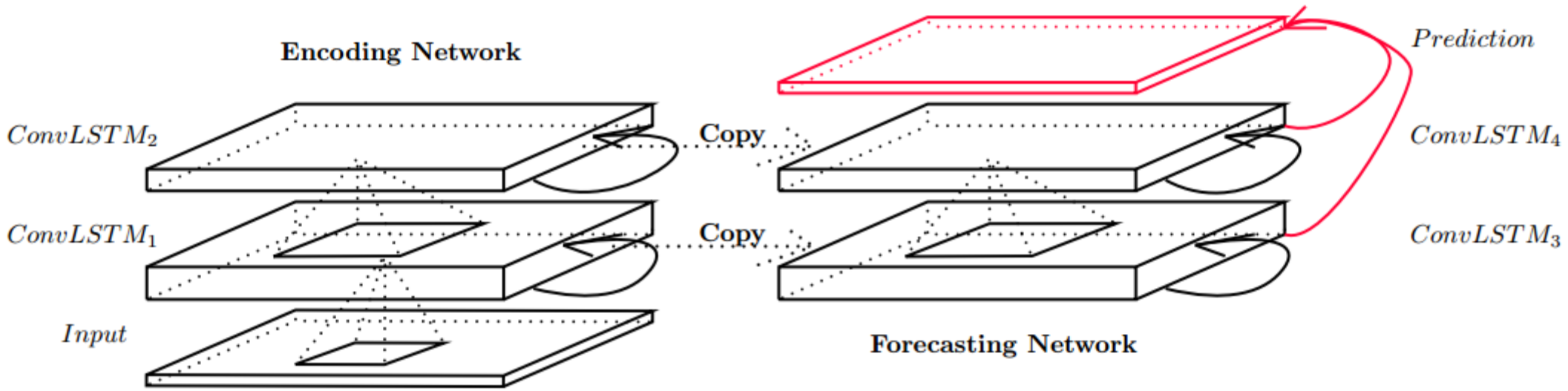


Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting

Cross Entropy Loss + BPTT + RMSProp + Early-stopping

Thanks !
Q&A