Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

AI Training Course Series

Introduction to NLP, RNN and LSTM

Lecture 5

Architecture
Development &
Algorithm
Refinement for
Intelligent Computing

Student: Yu-Chen Tsai (蔡雨蓁)
Advisor: Juinn-Dar Huang, Ph.D.

July 22, 2024

Outline

- Introduction to NLP
- Tasks
- Evaluation
- Preprocessing
- Introduction to RNN
- Structures
- Applications
- Problems and Limitations
- References
- Homework

Introduction

Natural Language Processing (NLP)

- The understanding and utilization of human language through the combination of mathematical theory and linguistic study
- Research started in 1950s, along with information theory
- Computing machine and cryptography were both developed rapidly during WW2
- Machine learning became an important tool since 1990s
- Many subtasks are included, such as language model, machine translation, question answering…

Development History (1/7)

- Key Technologies
  - 1997: Long-Short Term Memory
  - 2017: Transformer
  - 2020: Large Language Model

Development History (2/7)

- Key Technologies
  - Self-supervised learning
  - Transfer Learning
  - Word Vectorization
  - Improvement of GPU and computation power
  - Reduction of storage cost
  - Digitization of documents and its rapid distribution through the Internet

Development History (3/7)

- Paradigm Shift in NLP
  - Before 2018: Expert system, task-oriented, RNN and LSTM structure
  - After GPT-1 in 2018: General system, pretrain-finetune, Transformer structure

Development History (4/7)

- Self-Supervised Learning (SSL)
  - Pretrain
  - Well-annotated datasets are limited and time-consuming to create, yet raw texts are abundant and unused
  - Model can learn its parameters on raw text following some simple and automated goals
  - This goal should allow model training to converge on a meaningful form

Development History (5/7)

- Transfer Learning
  - Finetune
  - After pretraining, model should contain some general representation and understanding of its task
  - If the following task is similar to the pretraining task, then continuing training from pretrained model on the downstream task can be better and faster

Development History (6/7)

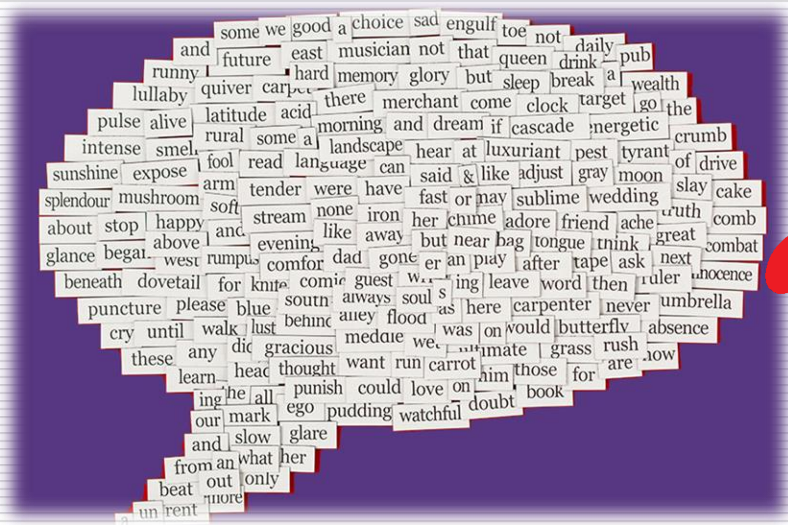
- Classification of Language Model (LM)
  - Causal Language Model (CLM)
    - Autoregressive (AR)
    - Used by GPT
  - Masked Language Model (MLM)
    - Autoencoder (AE)
    - Used by BERT

Development History (7/7)

- Word Vectorization
  - Also called embedding, is the representation of words using high-dimensional vectors
- Word2Vec (2013)
  - Published by Mikolov et al. at Google
- GloVe (2014)
  - Published by Pennington et al. at Stanford

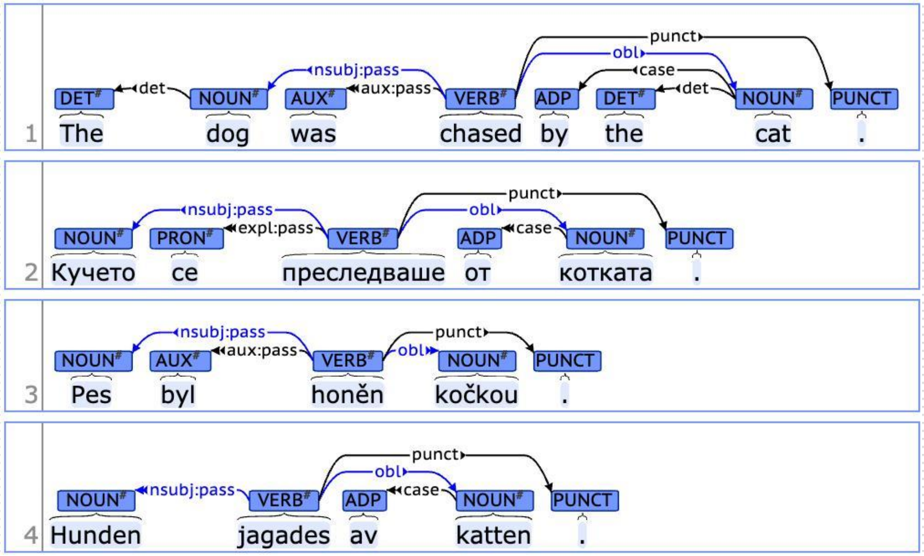
Word Representation

- How to represent words with numeric values?



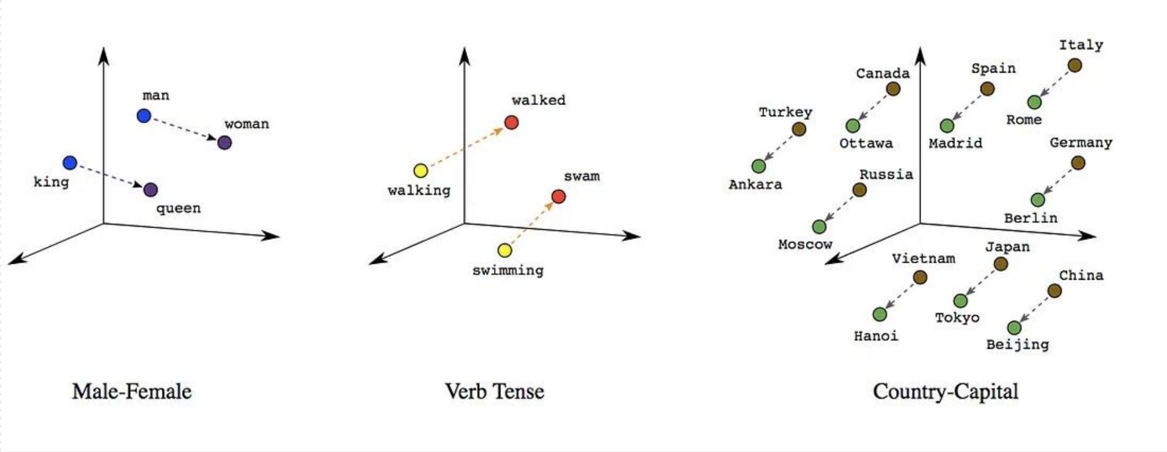
Word Representation (1/2)

- Structural representation
  - The linguistic approach
  - Words are labeled with grammatical properties
  - Sentences are annotated with semantic relationships, sometimes using tree graph
  - Time-consuming, cumbersome



Word Representation (2/2)

- **Vector** representation
  - The mathematical approach
  - Words and their meanings are “assumed” to exist in a high-dimensional numerical space
  - Words can be transformed into a numerical vector, this process is usually called “embedding”
  - These vectors are learnable by neural network



Tasks

Task Categories (1/2)

- Language modeling
- Grammatical acceptability
- Sentiment analysis
- Natural language inference (NLI)
- Question answering (QA)

Task Categories (2/2)

- Semantic similarity
- Summarization
- Translation
- …
- …
- …Emergent abilities

Evaluations

Common Evaluations

- General Language Understanding Evaluation (GLUE)
- Massive Multitask Language Understanding (MMLU)
- Beyond the Imitation Game (BIG-bench)

GLUE (1/3)

- *GLUE: A Multi-Task Benchmark And Analysis Platform For Natural Language Understanding*
- Published by Wang et al. on ICLR 2019
- Consists of 9 subtask and combined scoring for generality
- Public online score leaderboard

GLUE (2/3)

- Grammar
  - Corpus of Linguistic Acceptability (CoLA)
- Sentiment
  - Stanford Sentiment Treebank (SST-2)
- Semantic
  - Microsoft Research Paraphrase Corpus (MRPC)
  - Semantic Textual Similarity Benchmark (STS-B)
  - Quora Question Pair (QQP)

GLUE (3/3)

- Natural Language Inference (NLI)
  - Multiple-Genre Natural Language Inference (MNLI)
  - Recognizing Textual Entailment (RTE)
  - Winograd Schema Challenge (WNLI)
- Question Answering (QA)
  - Question Natural Language Inference (QNLI) (Partial NLI)

SuperGLUE (1/3)

- *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*
- Published by Wang et al. on NIPS 2019
- Developed 1 year after GLUE, after reported model accuracy surpassed average human performance
- Consists of 8 subtasks, more difficult than before
- Public online score leaderboard

SuperGLUE (2/3)

- Question Answering
  - Boolean Questions (BoolQ)
  - Choice of Possible Alternatives (CoPA)
  - Multi-Sentence Reading Comprehension (MultiRC)
  - Reading Comprehension with Commonsense Reasoning Dataset (ReCoRD)

SuperGLUE (3/3)

- Natural Language Inference
  - CommitmentBank (CB)
  - Recognizing Textual Entailment (RTE)
  - Winograd Schema Challenge (WSC)
- Word Sense Disambiguation
  - Word-in-Context (WiC)

MMLU (1/2)

- *Measuring Massive Multitask Language Understanding*
- Published by Hendrycks et al. on ICLR 2021
- 57 QA multiple-choice subset
  - Humanities: 13
  - Social Sciences: 12
  - STEM: 19
  - Other: 13
- Public online score leaderboard

MMLU (2/2)

- Humanities
  - Philosophy, History, Law…
- Social Sciences
  - Economy, Politics, Sociology…
- STEM
  - Mathematics, Physics, Chemistry, Biology
- Other
  - Global facts, Marketing…

BIG-bench

- *Beyond The Imitation Game: Quantifying And Extrapolating The Capabilities Of Language Models*
- Published by Srivastava et al. in 2022 by Google
- Collaborative project of more than 200 tasks
- Public online score leaderboard

Preprocessing

Tokenization (1/8)

- Processing sentences of words into a series of tokens, for organized input to model
- Three types:
  - Word-based Tokenization
  - Character-based Tokenization
  - Subword-based Tokenization

Tokenization (2/8)

- Word-level Tokenization
  - Separation by space, punctuation, and some simple rules
  - Implemented by Python packages like Spacy and NLTK
  - Ex: [It’s cold now, don’t go out.]

    => [It | ‘s | cold | now | , | do | n’t | go | out | .]
  - Fast and simple, what’s the problem?

Tokenization (3/8)

- Synthetic Language
  - Including much of Indo-European languages, English is considered weakly synthetic
  - These languages contains many prefixes, suffixes and inflections for words
    - Ex: [Boy, boy, boys] , [go, goes, going] , [do, undo]
  - Some languages are even "Agglutinative"
    - Ex: Donaudampfschiffahrtsgesellschaft (*Danube-Steamboat-Shipping Company*) (German)

Tokenization (4/8)

- Synthetic Language
  - Simply using word-based tokenizer will bloat up the dictionary, unless the rules are excessively modified

- Analytic Language
  - Including Chinese
  - Words in these languages are more “Isolated”
  - Tokenization will be less of a problem

Tokenization (5/8)

- Character-based Tokenization
  - Separation to each alphabet
  - Ex: [Hello.]

    => [H | e | l | l | o | .]
  - Dictionary size can be very small, but word-level meaning is completely removed
  - Where's the middle ground?

Tokenization (6/8)

- Subword-based Tokenization ★
  - For any corpus, we can generate a dictionary of set size with a given algorithm, and it can be used to represent the complete corpus
  - This is the most common method now
  - Common algorithms:
    - Byte-Pair Encoding (BPE)
    - WordPiece
    - Unigram

Tokenization (7/8)

- Byte-Pair Encoding (2015)
  - Words are initially separated to each alphabet to form the basic dictionary, and frequency of each word is recorded
  - The most-occurred token pair is merged to form a new, longer token and added into dictionary
  - Repeat until the dictionary has expanded to the set size

Tokenization (8/8)

– 1. Given corpus and frequency, separate into character

```
("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
```

```
("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
```

– 2. Find the most occurred token pair, in this case “u” and “g”, add “ug” into vocabulary

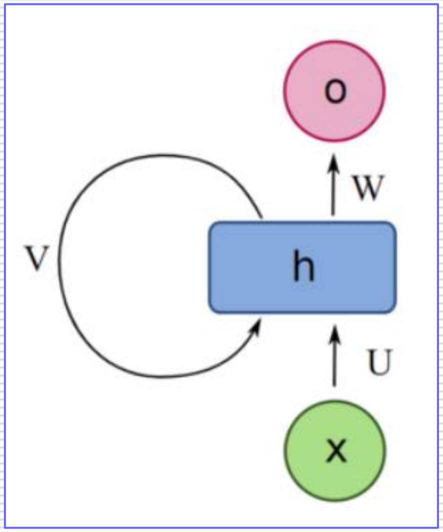
```
("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)
```

– 3. Now the most occurred token pair changes to “u” and “n”, add “un” into vocabulary and repeat this process…

Introduction to RNN

Introduction

- Recurrent Neural Network
  - 循環神經網路，簡稱 RNN
  - Developed to model sequential problem
  - A neural network with changing memory

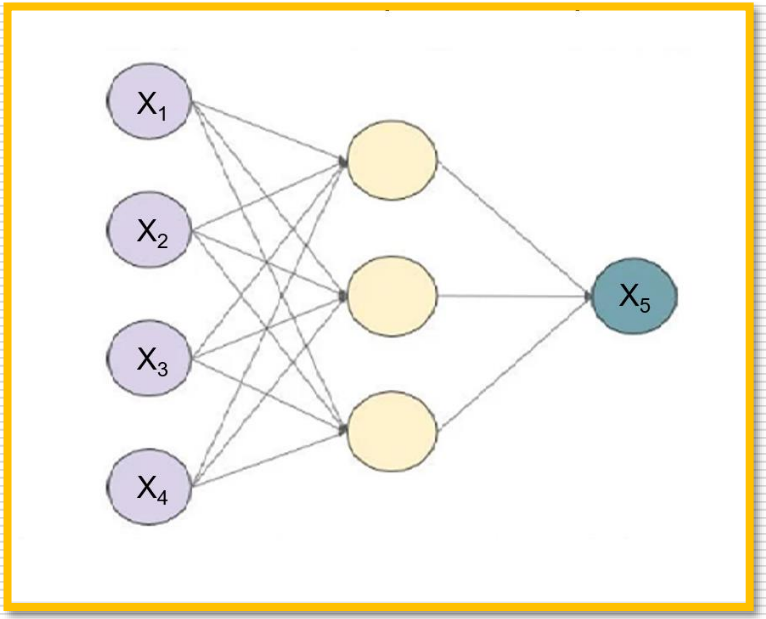


- Recurrence Relation
  - Initial condition
  - Equation for next number

$$F_n = F_{n-1} + F_{n-2}$$
$$F_0 = 0$$
$$F_1 = 1.$$

Why do we need RNN? (1/2)

- Sequence Modeling
  - Can an FNN be used to model a sequential problem?
- As a though experiment…
  - Can an FNN be trained to predict sequences like $[X_1, X_2, X_3, X_4, X_5]$?



Why do we need RNN? (2/2)

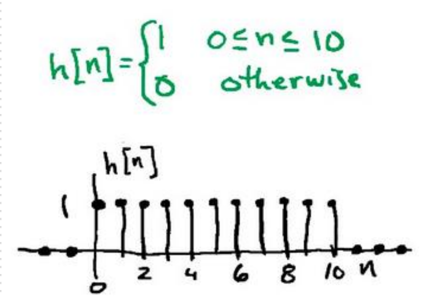
- Sequence Modeling
  - Yes, it's certainly "possible" with FNN…
  - … But there are some big issues

- Back in 1970s and 1980s
  - Early FNN were quite shallow, often less than 10 layers, with limited expressiveness for long sequence
  - Before backpropagation, early training algorithm were quite slow, and computation power was very limited
  - The addition of memory element grants much greater possibility for modeling

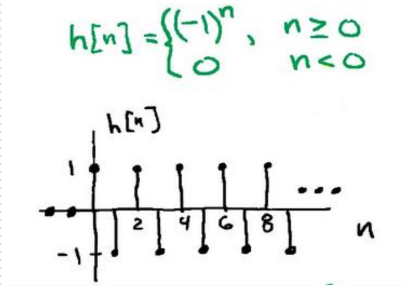
Characteristics

- Signal Processing
  - Basic analysis of NN using concept of impulse response

- Feedforward NN
  - Finite impulse response (FIR)
  - With a fixed window length for input



- Recurrent NN
  - Infinite impulse response (IIR)
  - Memory element is preserved from the start
  - IR can be extended to infinity (theoretically)



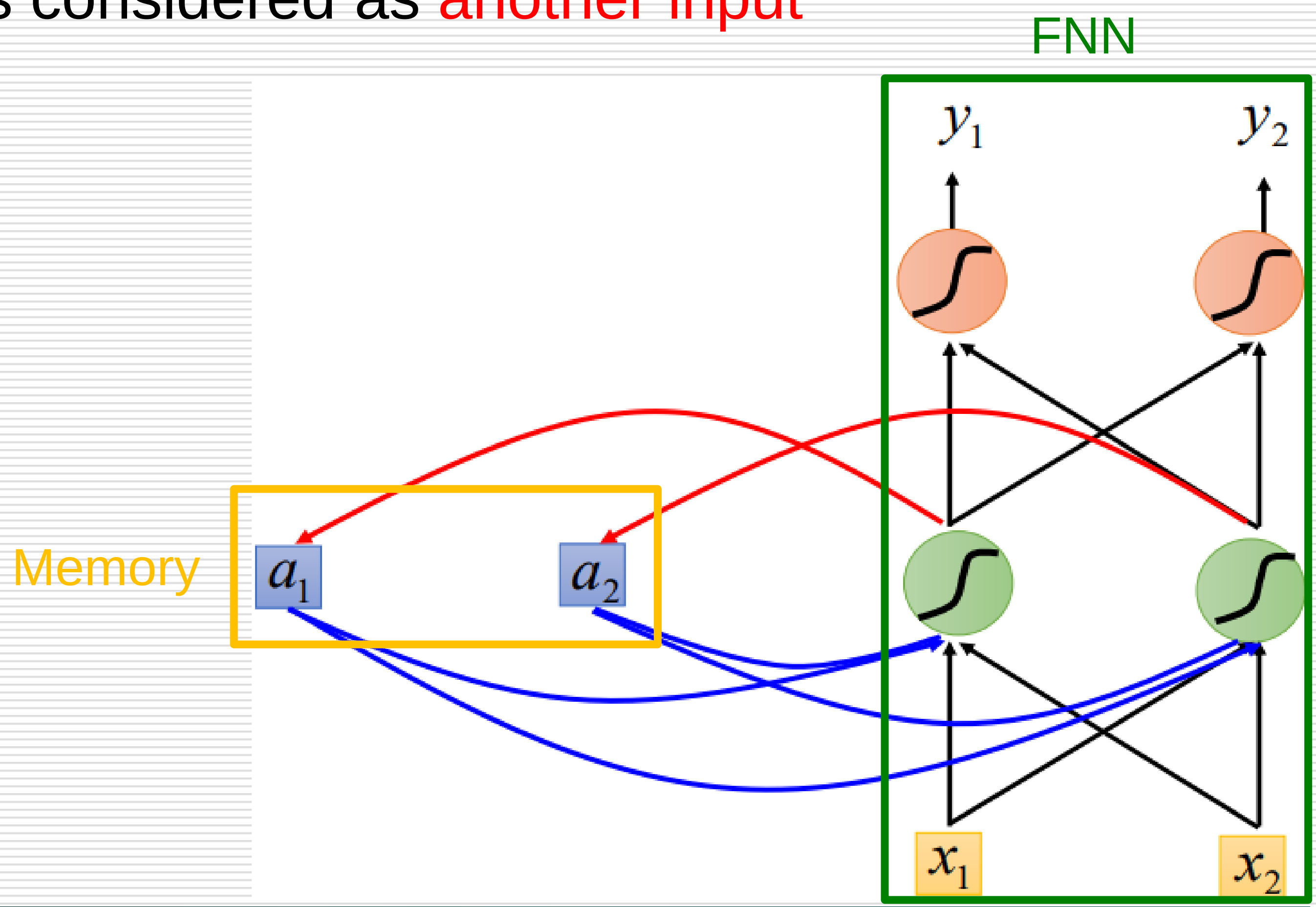
Structures

Structures

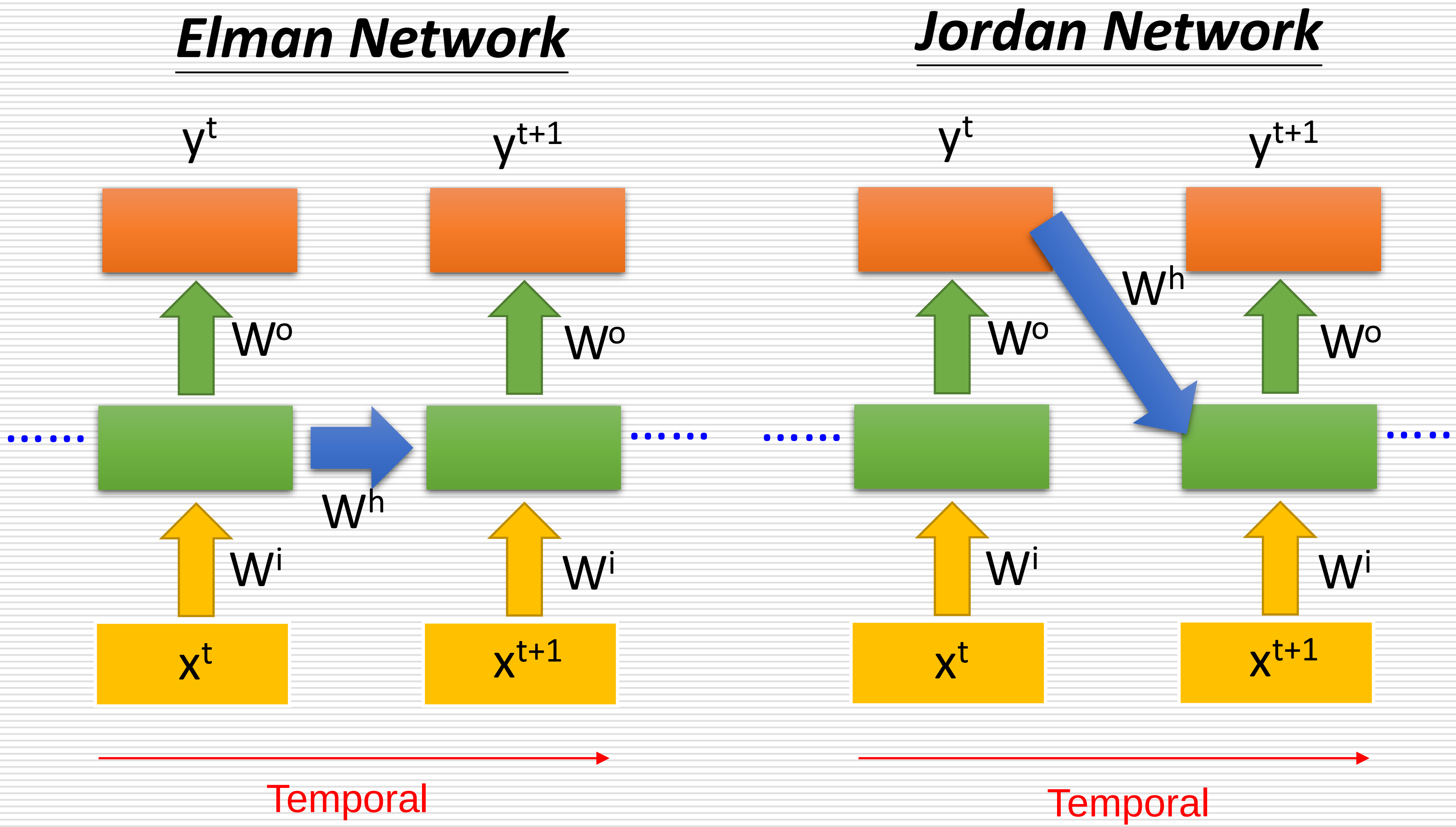
- Simple Recurrent Network (SRN)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Simple Recurrent Network

- Think of it as an FNN with memory
  - output of hidden layer are stored in the memory
  - memory is considered as another input



Variant Structures of SRN

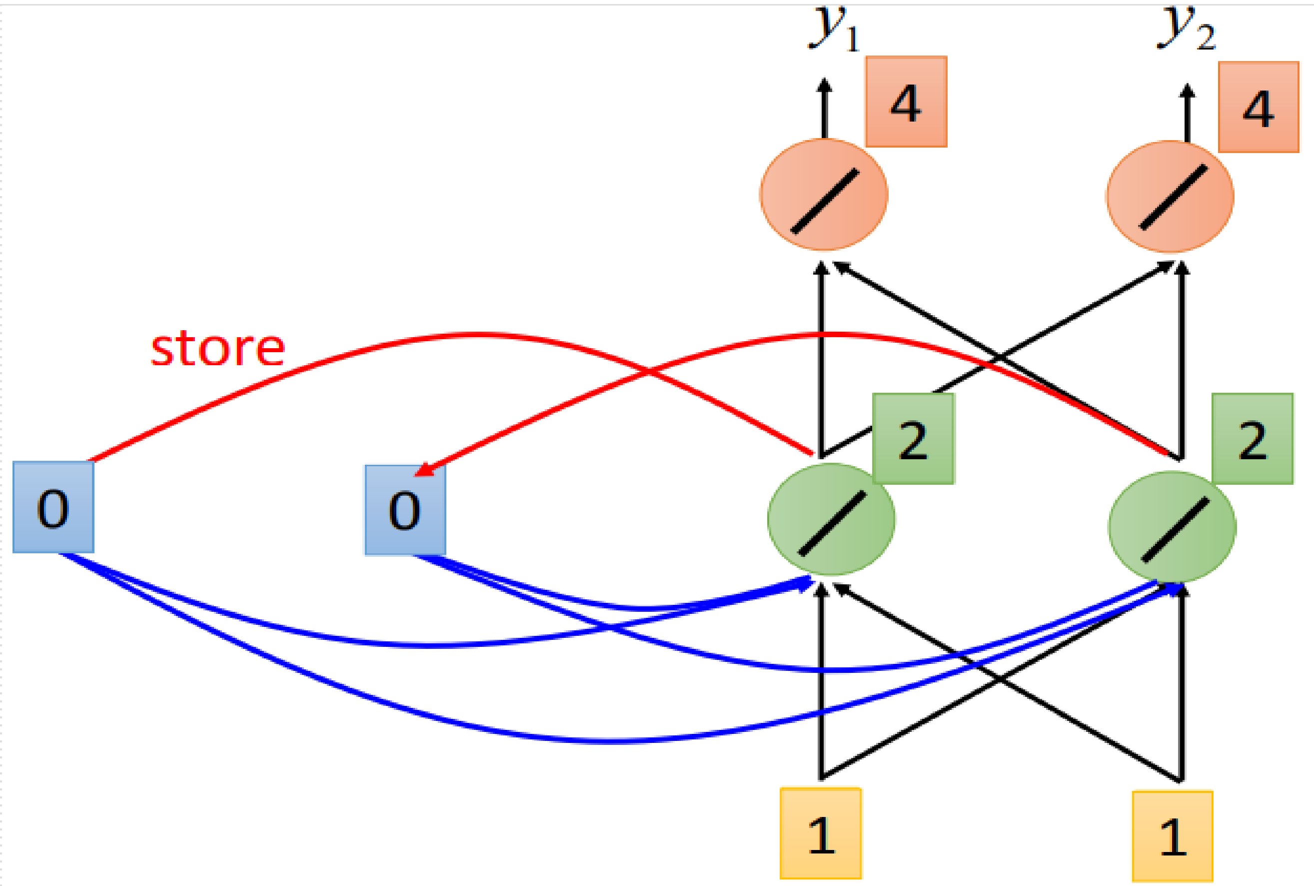


Example (1/4)

- For simplicity, assume
  - weights are all 1, bias are all 0
  - all activation functions are linear
  - initial value of memory (a1, a2) are all 0
- Three input sequences of 2 elements
  - [x1, x2] : [1, 1] → [1, 1] → [2, 2]

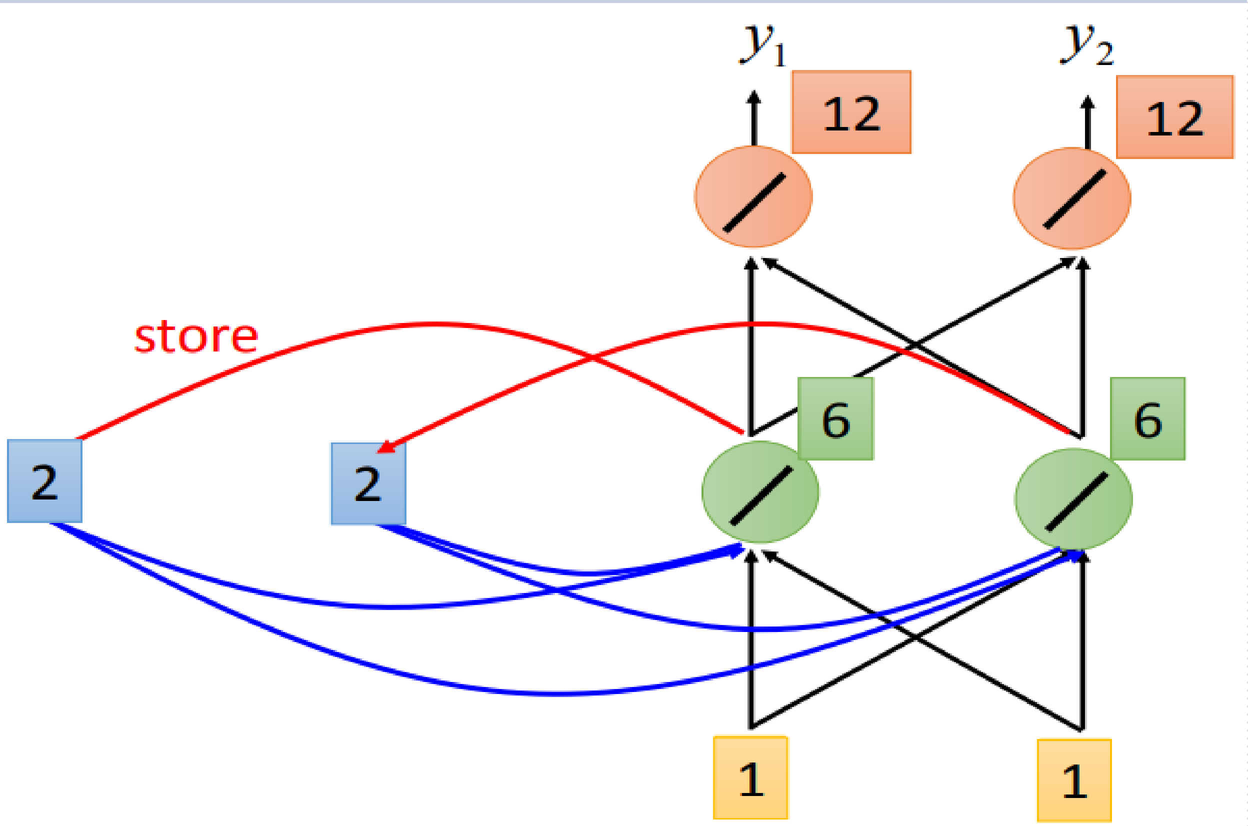
Example (2/4)

- For t = t0, input [x1, x2] = [1, 1], output [y1, y2] = [4,4]
- Store the output of hidden layer [2,2] in memory



Example (3/4)

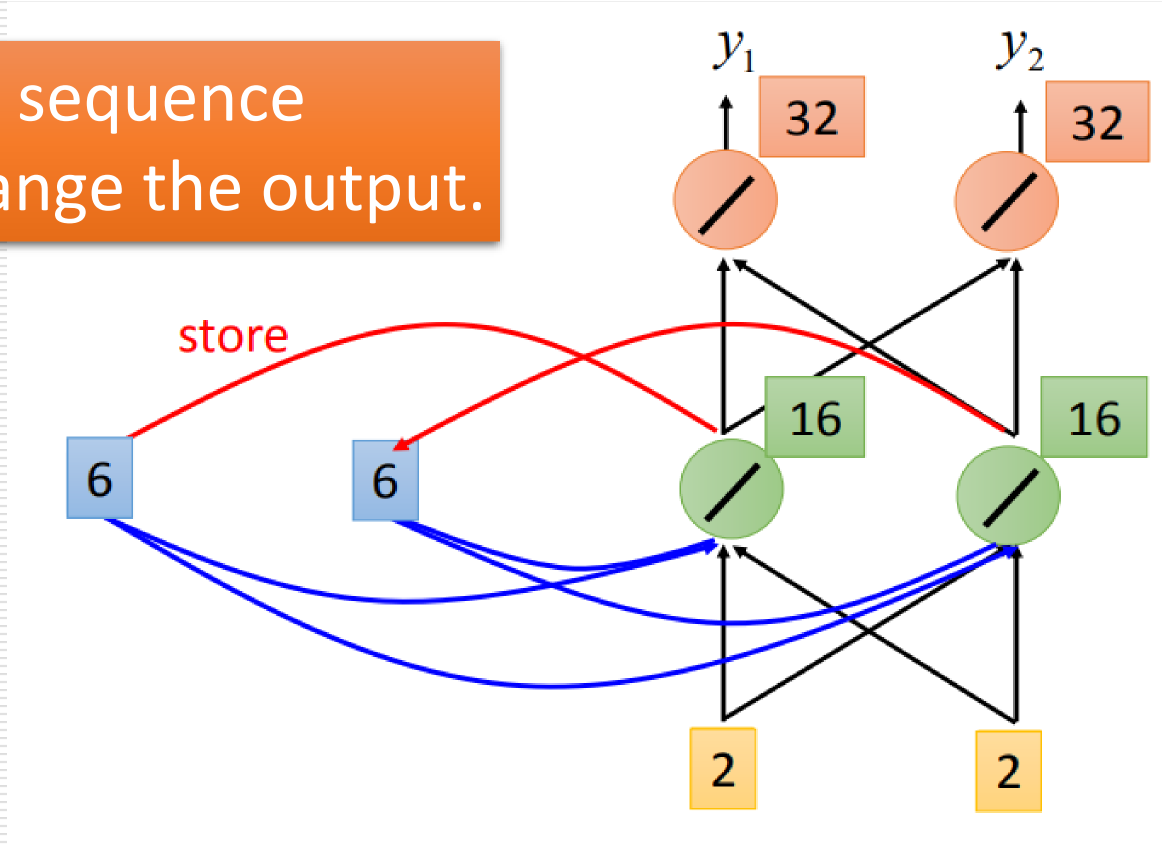
- For t = t1, input [x1, x2] = [1, 1], consider values in memory as another inputs
  - values in memory affects outputs
- Store the output of hidden layer [6,6] in memory



Example (4/4)

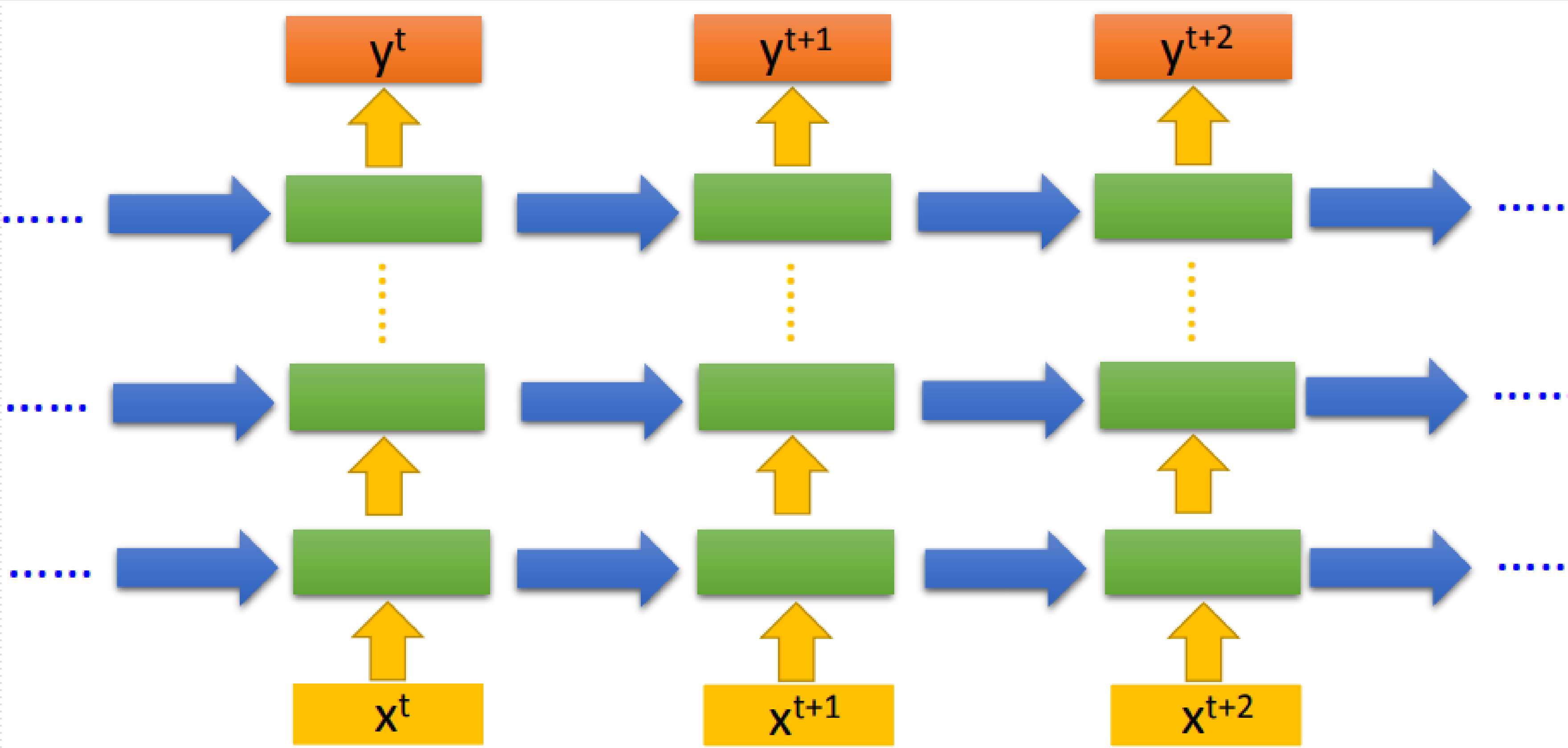
- For t = t2, input [x1, x2] = [4, 4], needs to consider the value in memory as another inputs again
- Store the output of hidden layer [16,16] in memory

Changing the sequence order will change the output.



Go Deeper

- RNN can go deeper by adding more hidden layers
  - just like CNN models, usually have better performance, up to a limit



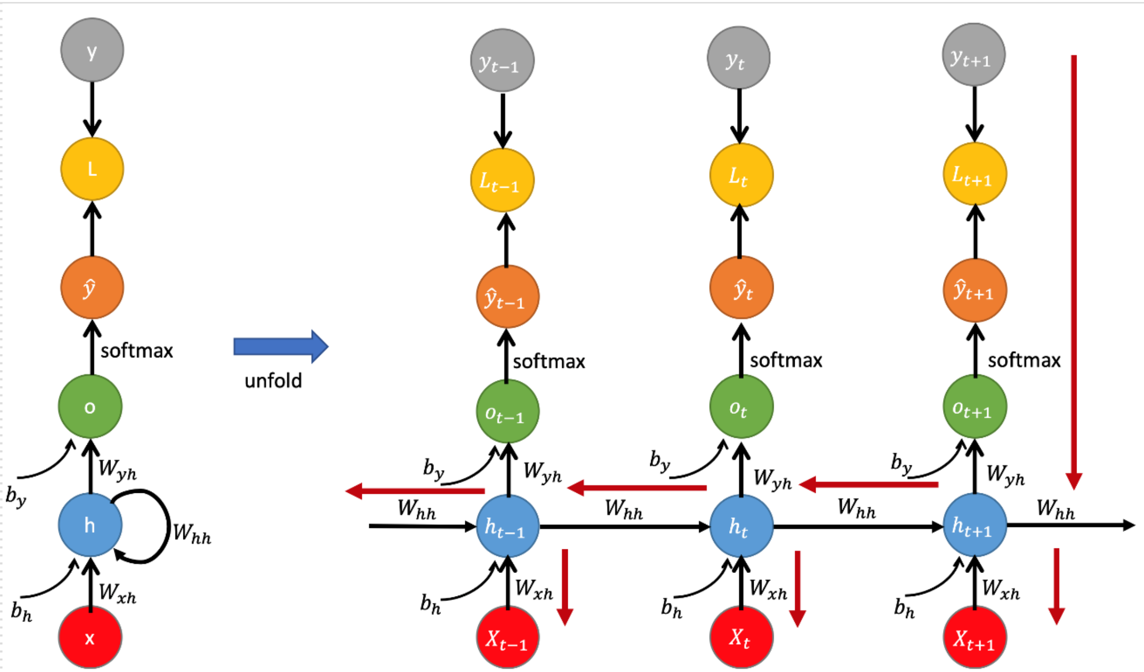
Training (1/3)

- Backpropagation on MLP
  - MLP has acyclical structure
  - Backpropagation can be calculated in 1 pass

- Backpropagation on RNN
  - RNN has cyclical structure
  - How to propagate?

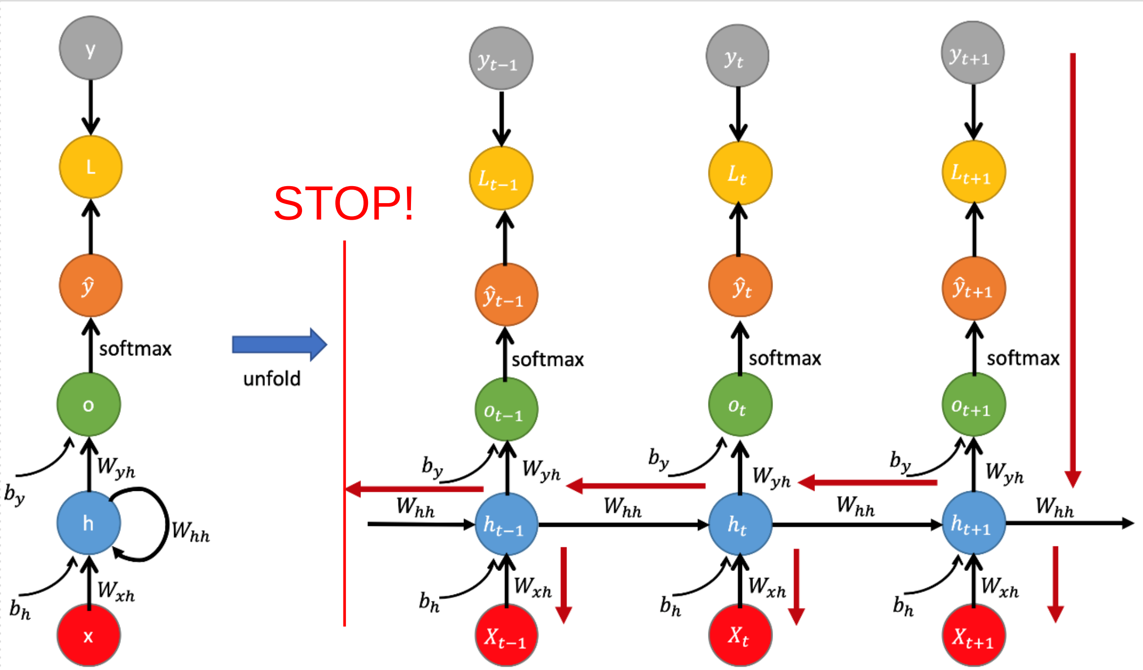
Training (2/3)

- Backpropagation Through Time (BPTT)
  - Developed in the late 1980s
  - Unfolding the looping structure of RNN
  - Gradient on the same parameter at different timestep will be accumulated



Training (3/3)

- Backpropagation Through Time (BPTT)
  - Realistically, for very long sequences, BPTT needs to stop at a determined maximum timestep, this method is called Truncated BPTT (TBPTT)



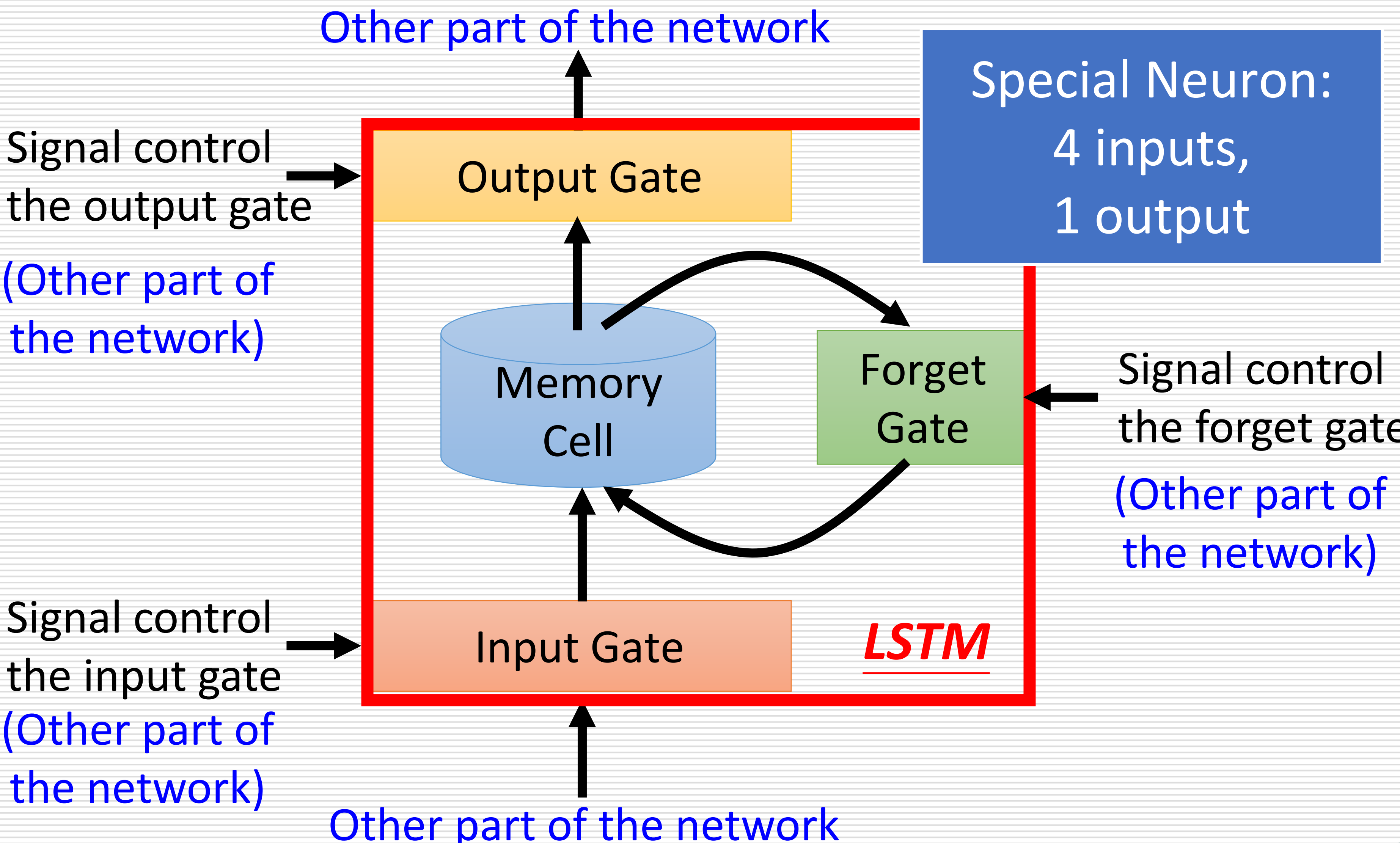
There Are Some Problems (1/2)

- As with MLP and CNN, RNN suffers from the vanishing and exploding gradient problem
  - This can make training unstable, and also limit RNN to very shallow architecture
  - This problem is now solved with residual connection in CNN like ResNet, thus allowing deeper architecture
  - The same solution can't be easily applied to temporally unfolded RNN
  - LSTM uses input and output gating to enforce a theoretical constant error flow

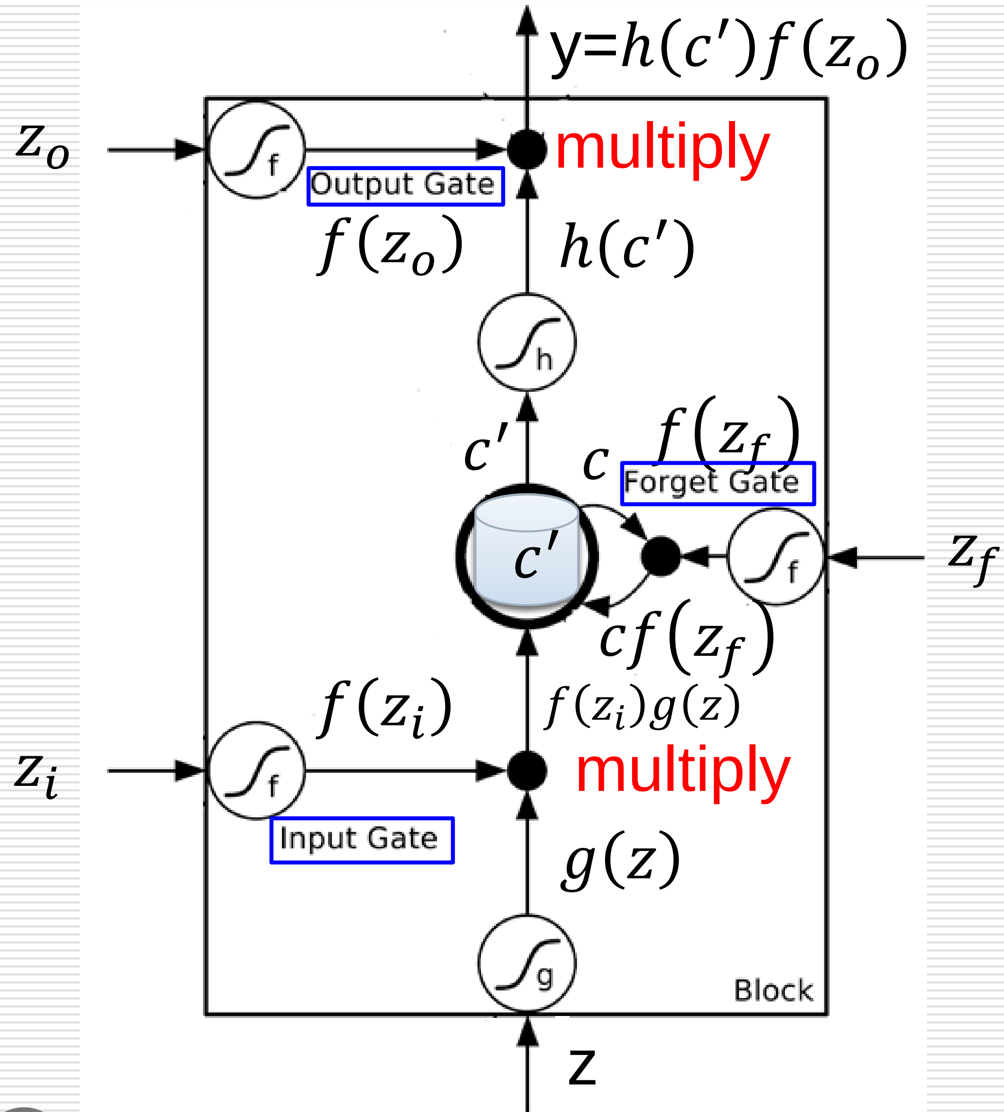
There Are Some Problems (2/2)

- As more time step increases, influences of the previous information will approach 0
  - RNN is like a “Goldfish Brain”, only has short-term memory
  - LSTM solves this problem by using forget gating to control the retention of long-range memory

Long Short-Term Memory (LSTM) (1/3)



Long Short-Term Memory (LSTM) (2/3)

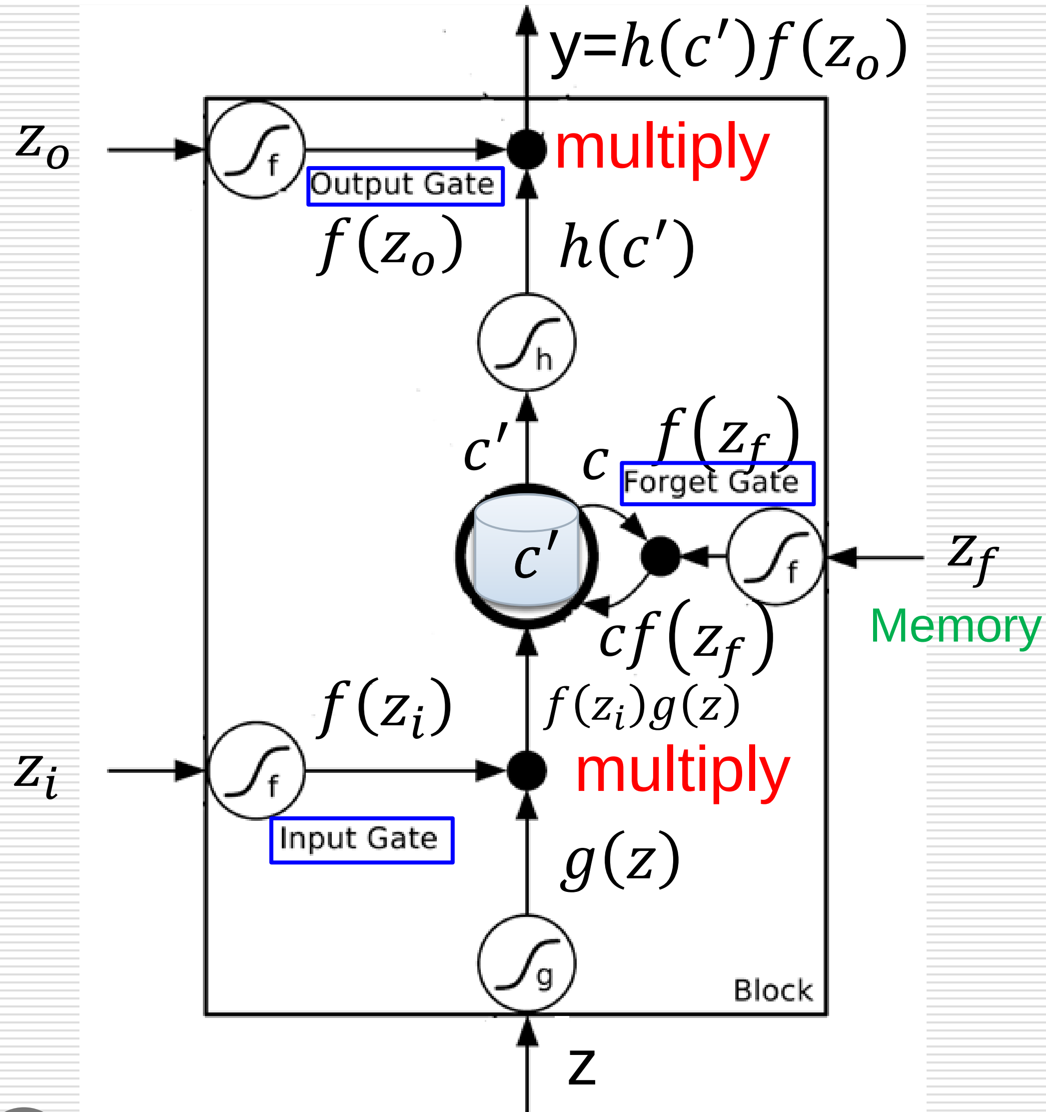


Three gates:
Input, Forget, Output

Single number
Between 0 and 1
Mimic open and closed gate

Activation function f is usually a sigmoid function

Long Short-Term Memory (LSTM) (3/3)



- Output Gate: How much of memory should be outputted
- Forget Gate: How much of old memory should be forgotten
- Input Gate: How much of new input should be remembered

LSTM – Example (1/7)

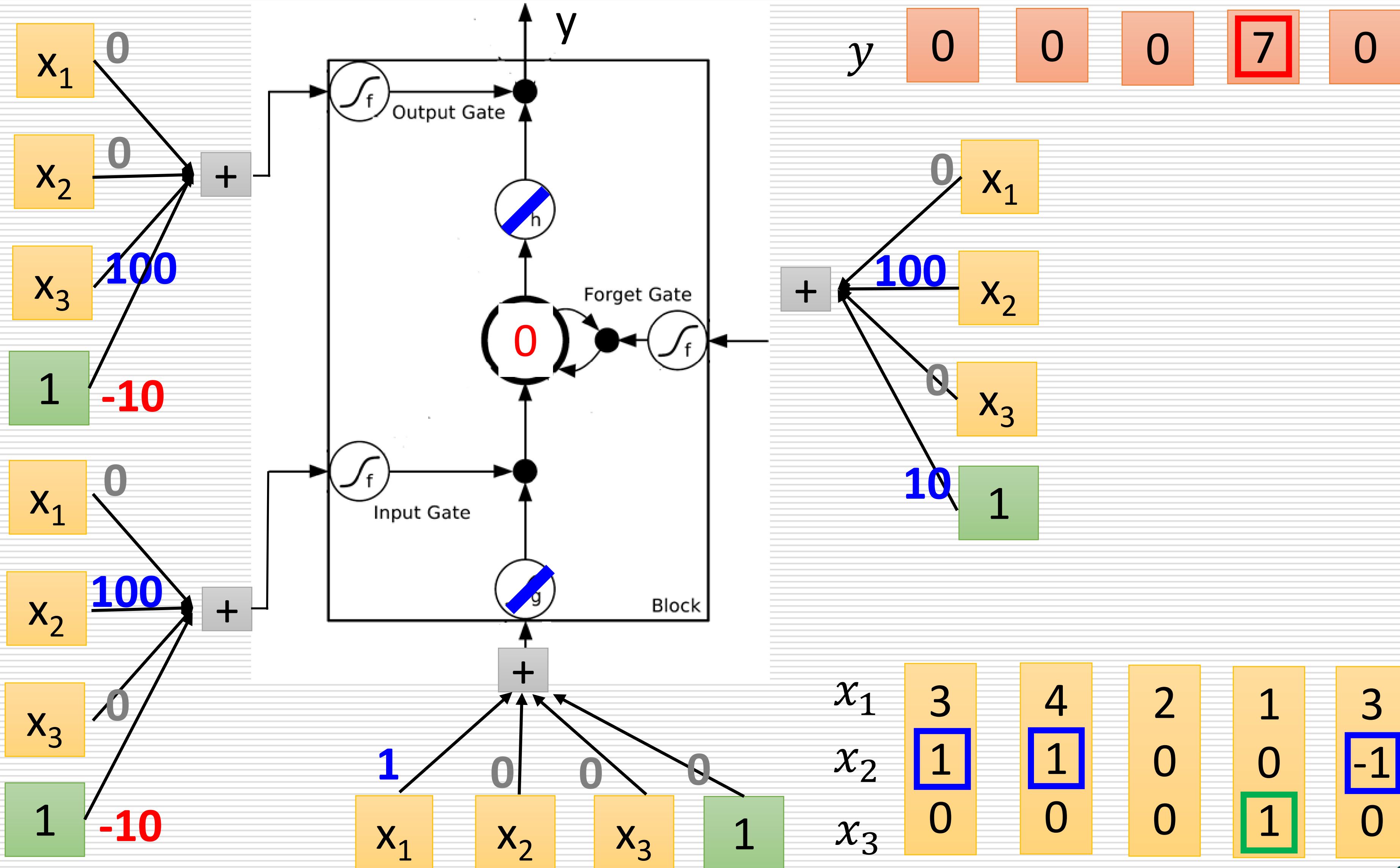
	0	0	3	3	7	7	7	0	6
x_1	1	3	2	4	2	1	3	6	1
x_2	0	1	0	1	0	0	-1	1	0
x_3	0	0	0	0	0	1	0	0	1
y	0	0	0	0	0	7	0	0	6

When $x_2 = 1$, add the numbers of x_1 into the memory

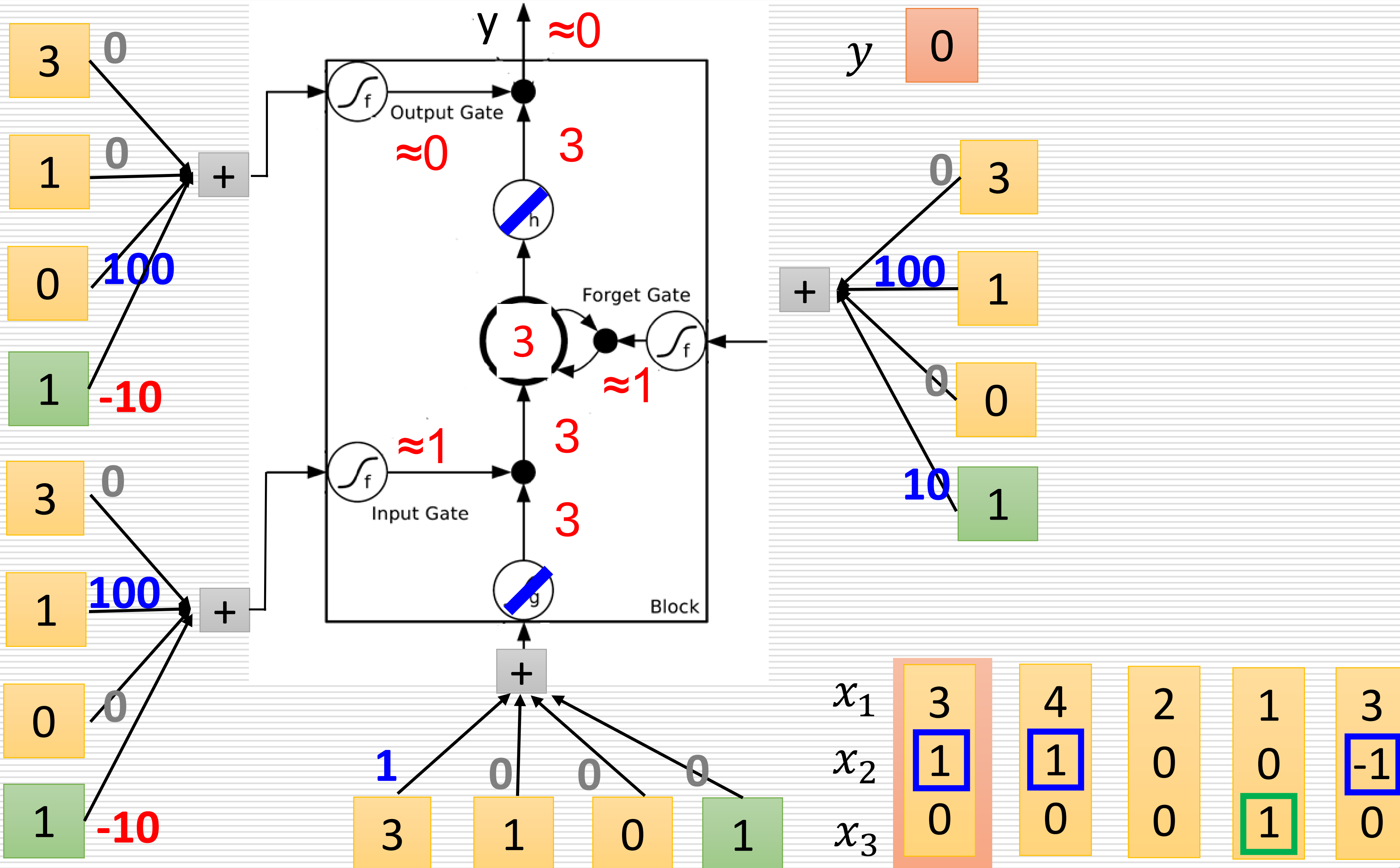
When $x_2 = -1$, reset the memory

When $x_3 = 1$, output the number in the memory.

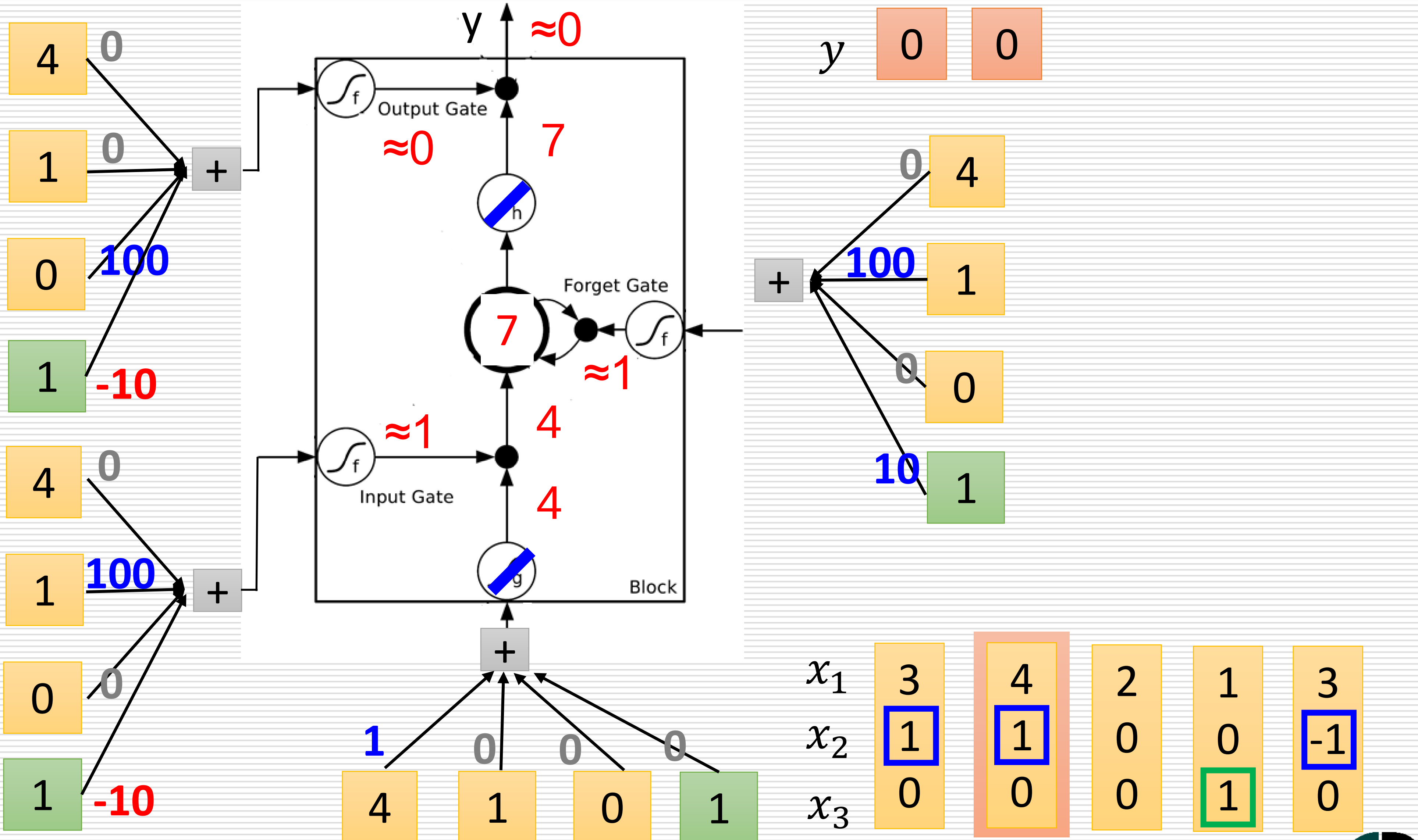
LSTM – Example (2/7)



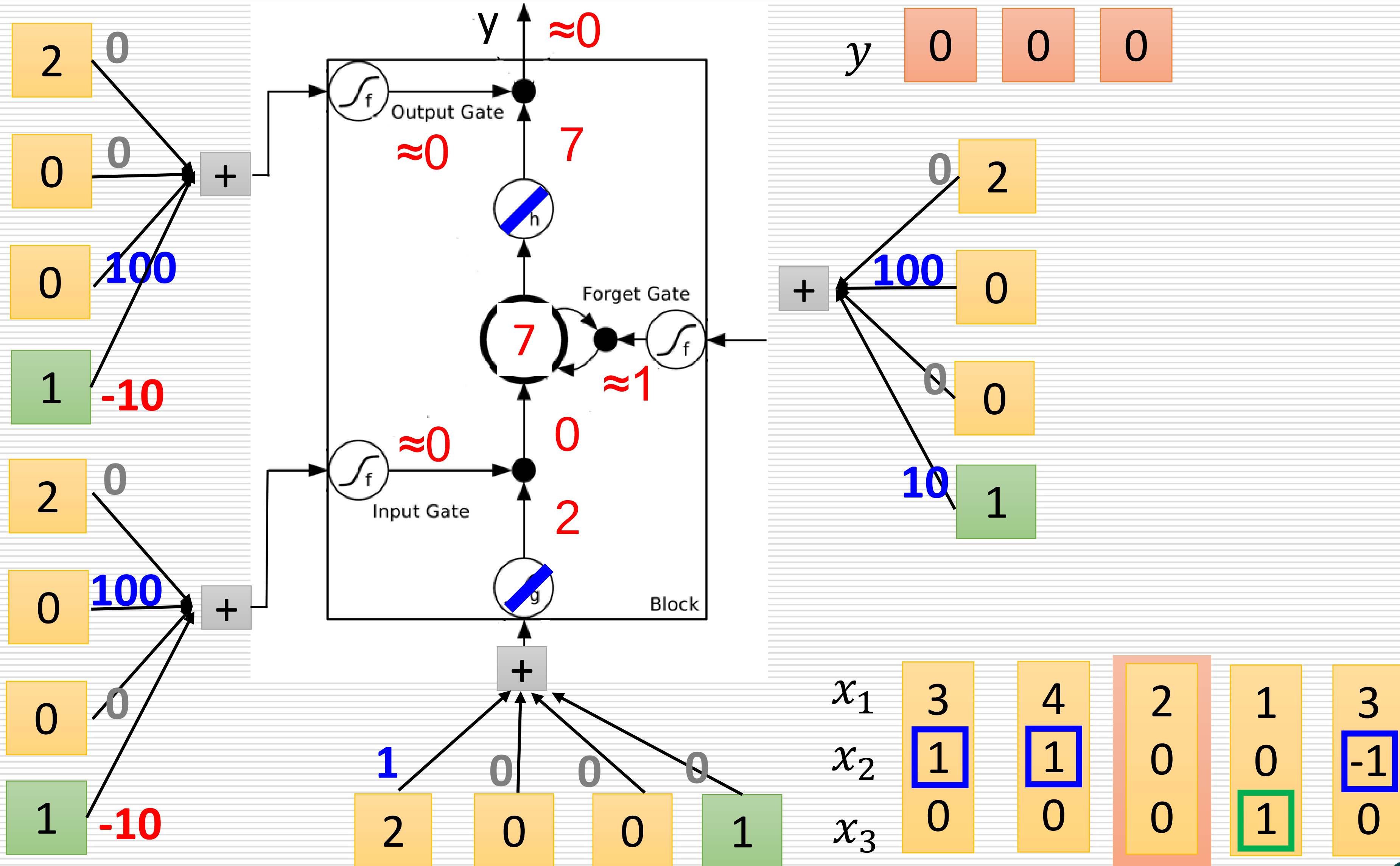
LSTM – Example (3/7)



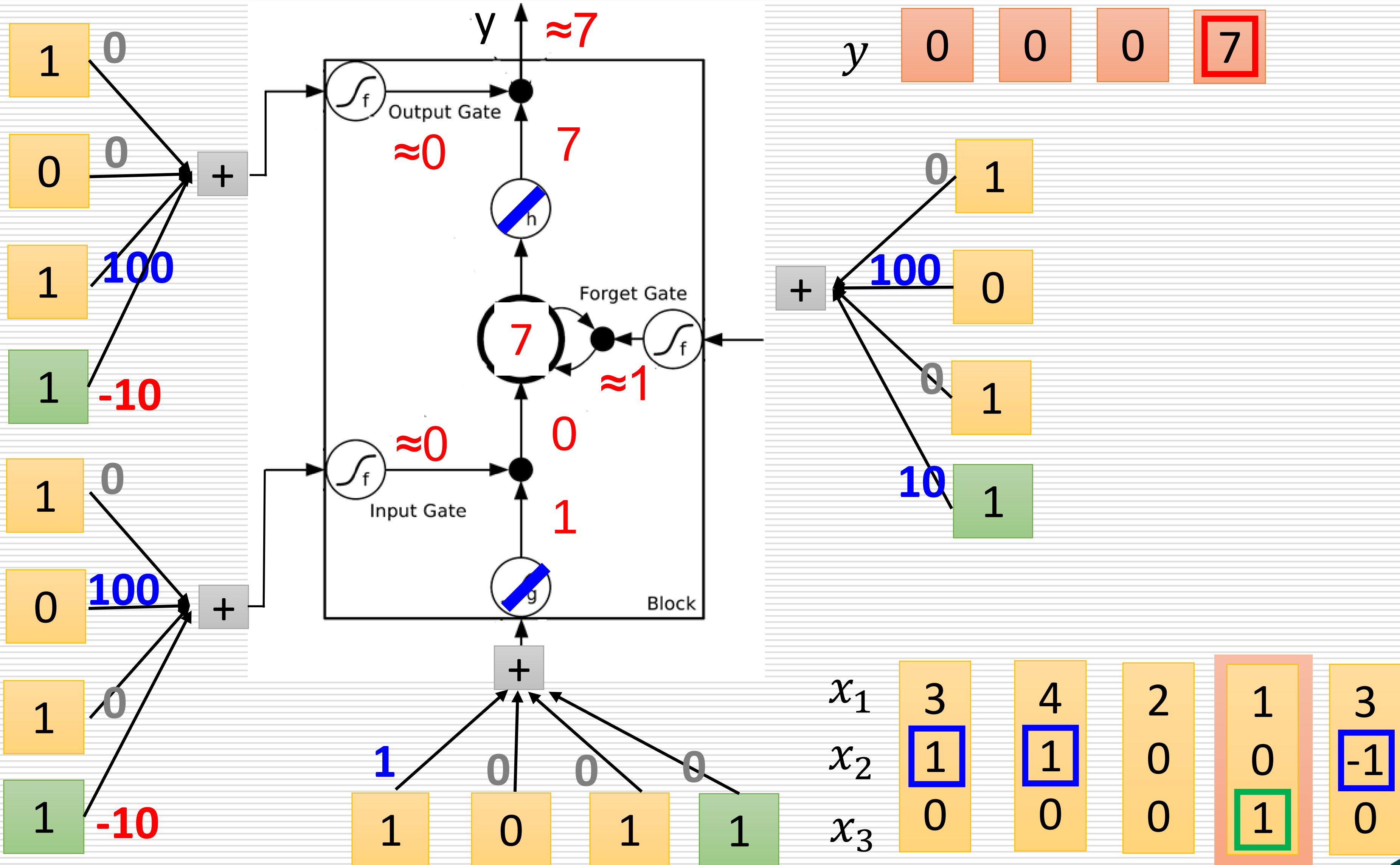
LSTM – Example (4/7)



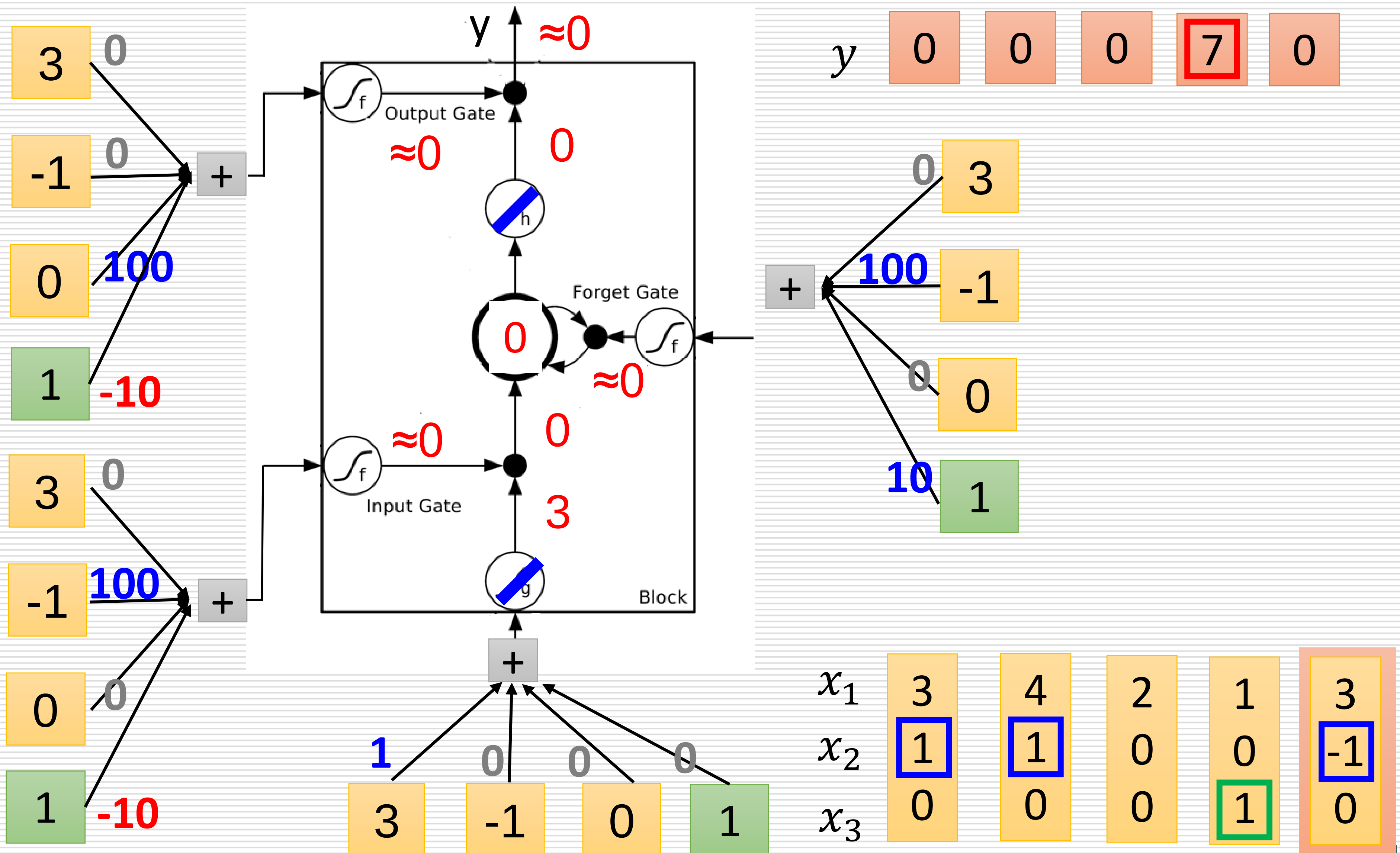
LSTM – Example (5/7)



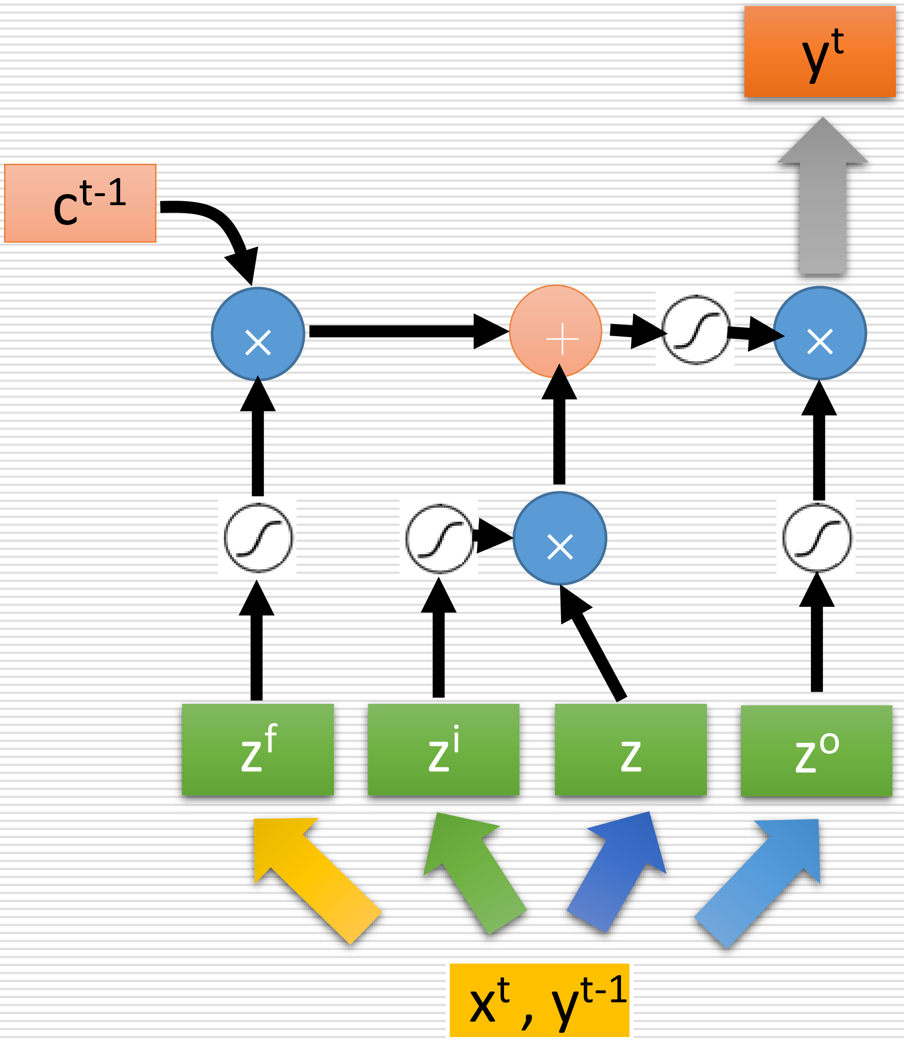
LSTM – Example (6/7)

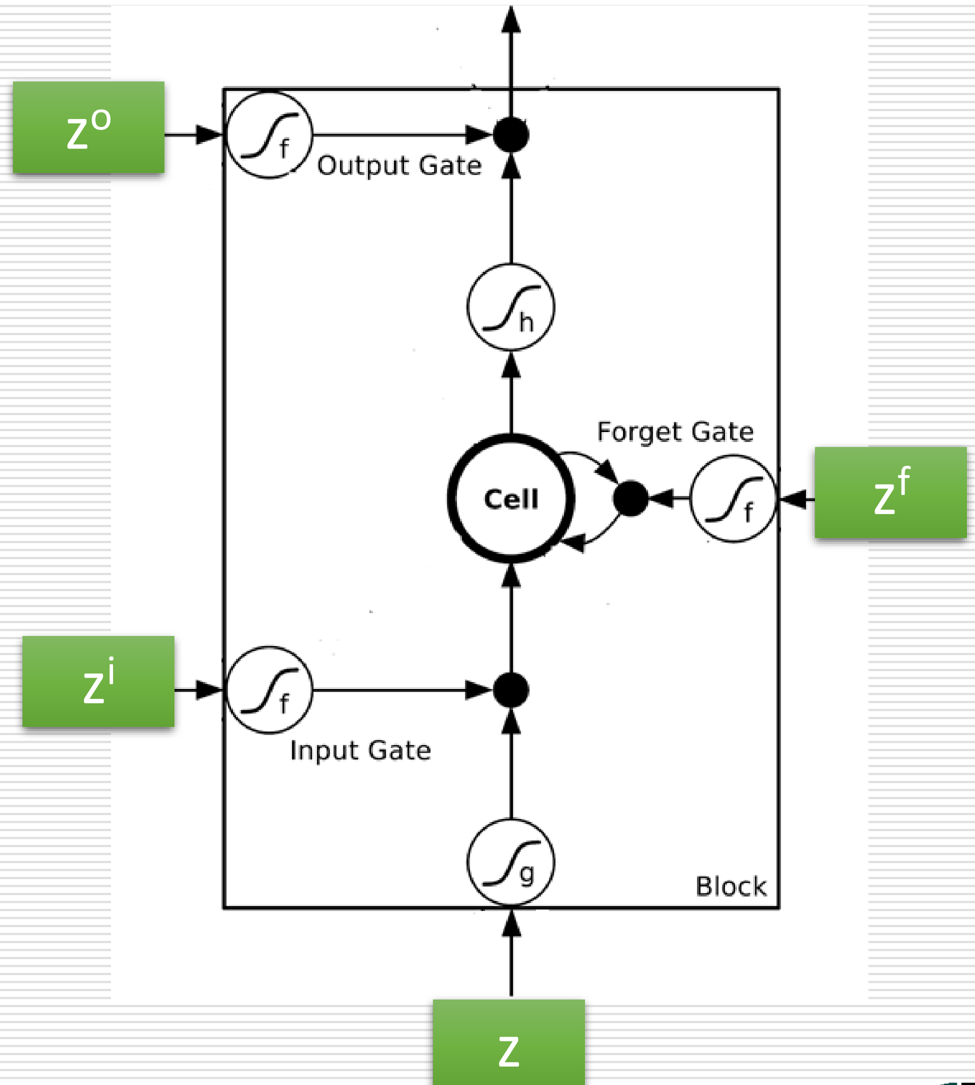


LSTM – Example (7/7)



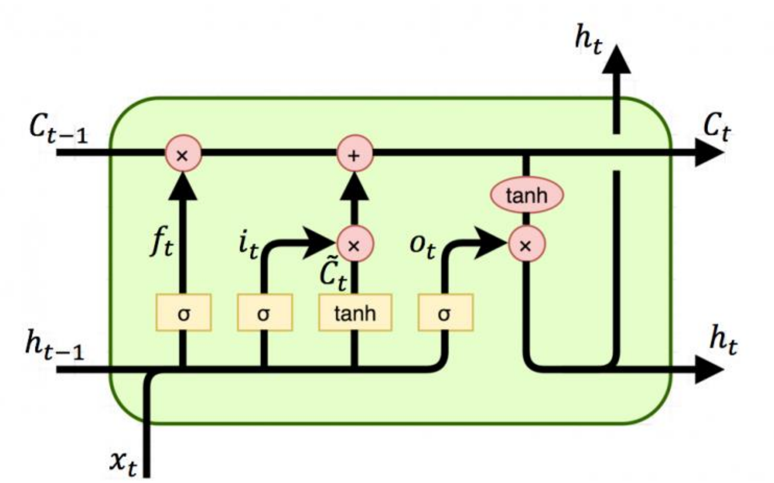
LSTM – Block Diagram (1/2)





LSTM – Block Diagram (2/2)

- Common representation
  - W: weight on input state, U: weight on hidden state
  - Wf: Forget gate, Wi: Input gate, Wo: Output gate, Wc: Cell control
  - Cell control uses tanh activation, unlike gating



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

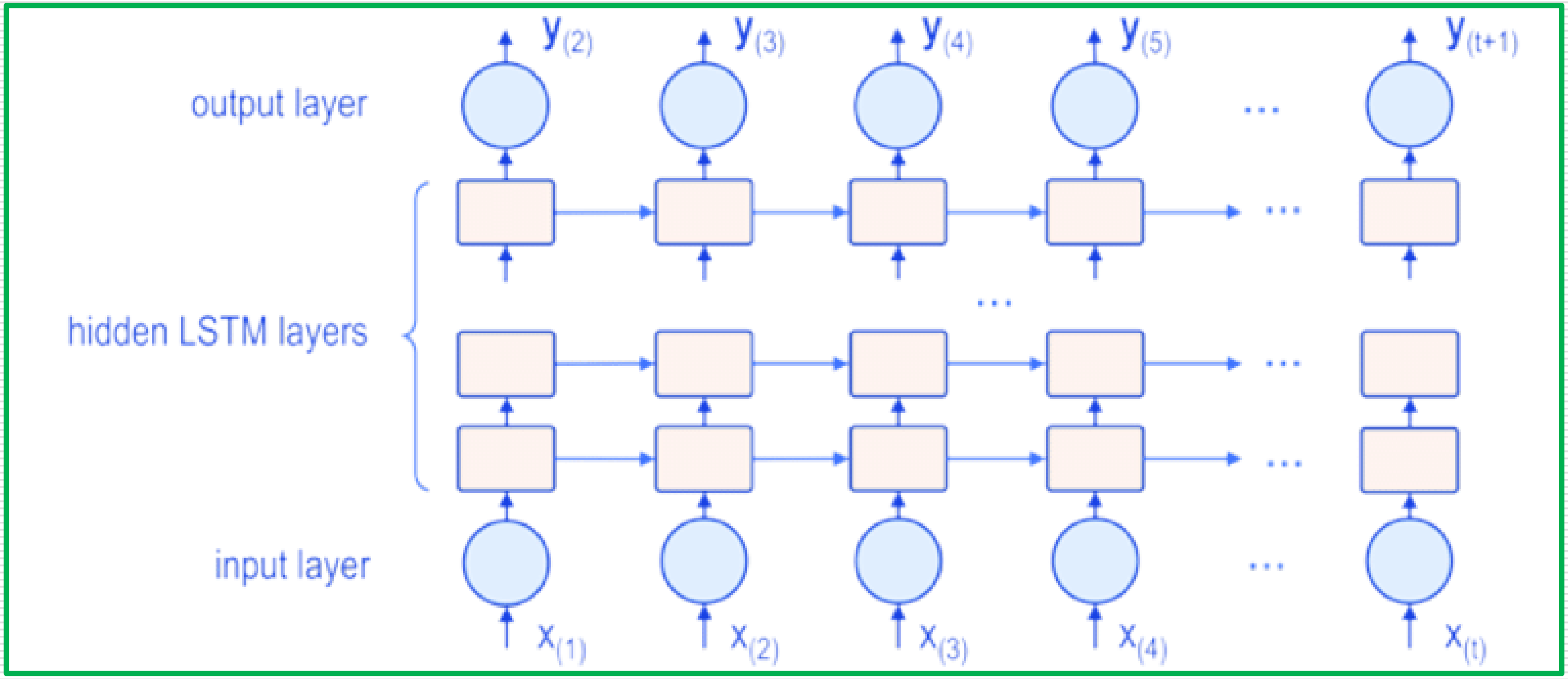
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

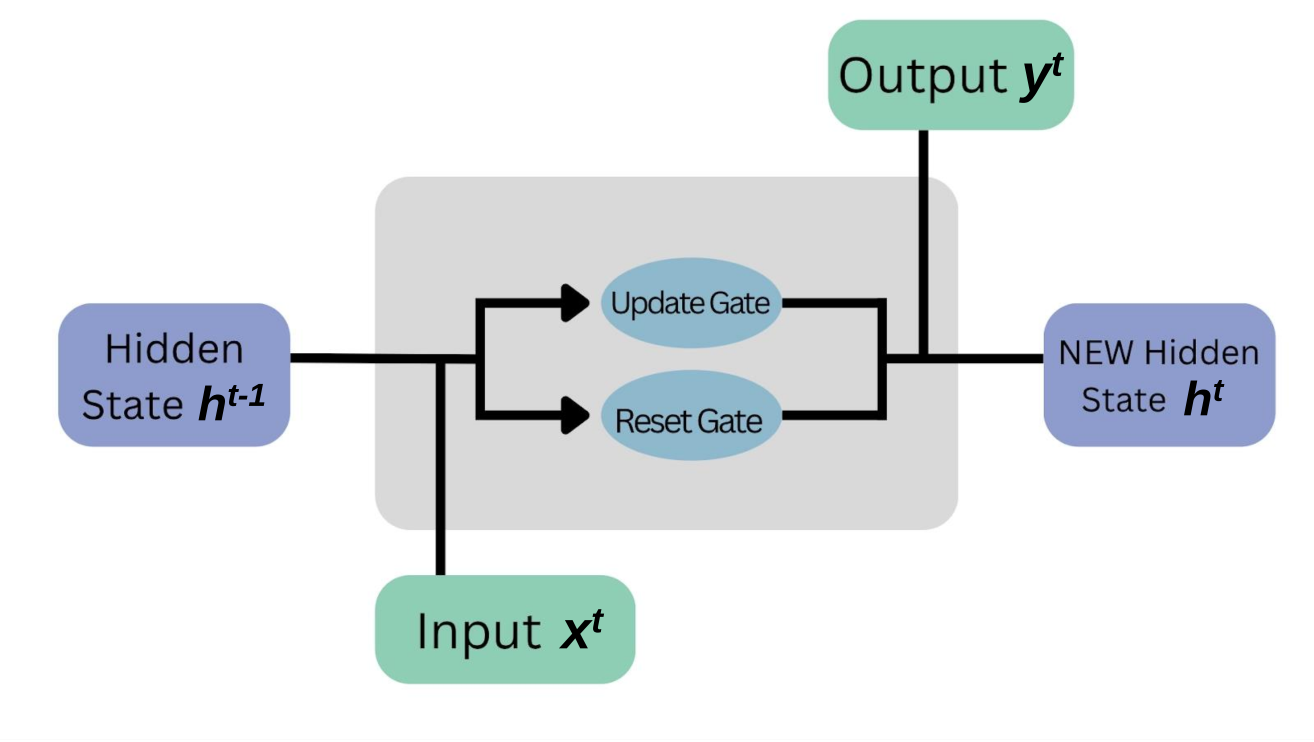
$$h_t = o_t \odot \sigma_h(c_t)$$

Multi-Layer LSTM



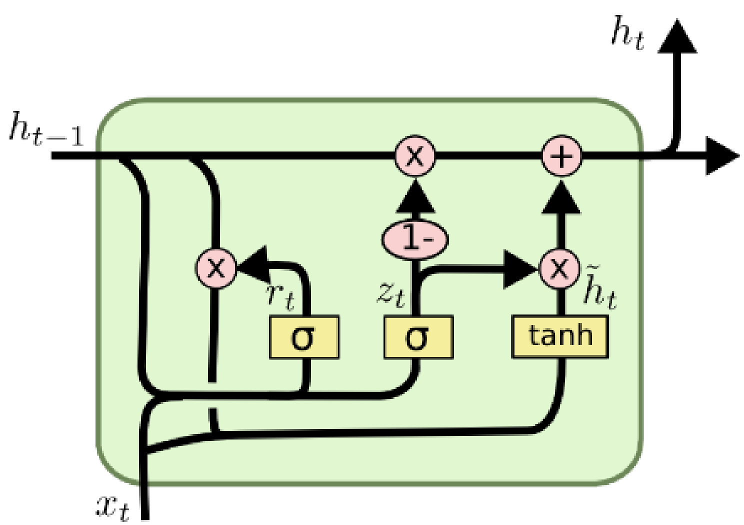
Gated Recurrent Unit (GRU)

- Simplify LSTM structure by merging input and forget gate into update gate
- Cell state and hidden state are merged together



GRU – Block Diagram

- Common representation:
  - Wr : Update gate (Forget + Input), Wz: Reset gate
    W: Output control



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Tasks and Applications

Tasks and Applications

- Natural Language Processing
  - Machine Translation
  - Language Model
  - Question Answering…
  - This is a major part of research in our lab!
- Audio Processing
- Video Processing

Google Translate

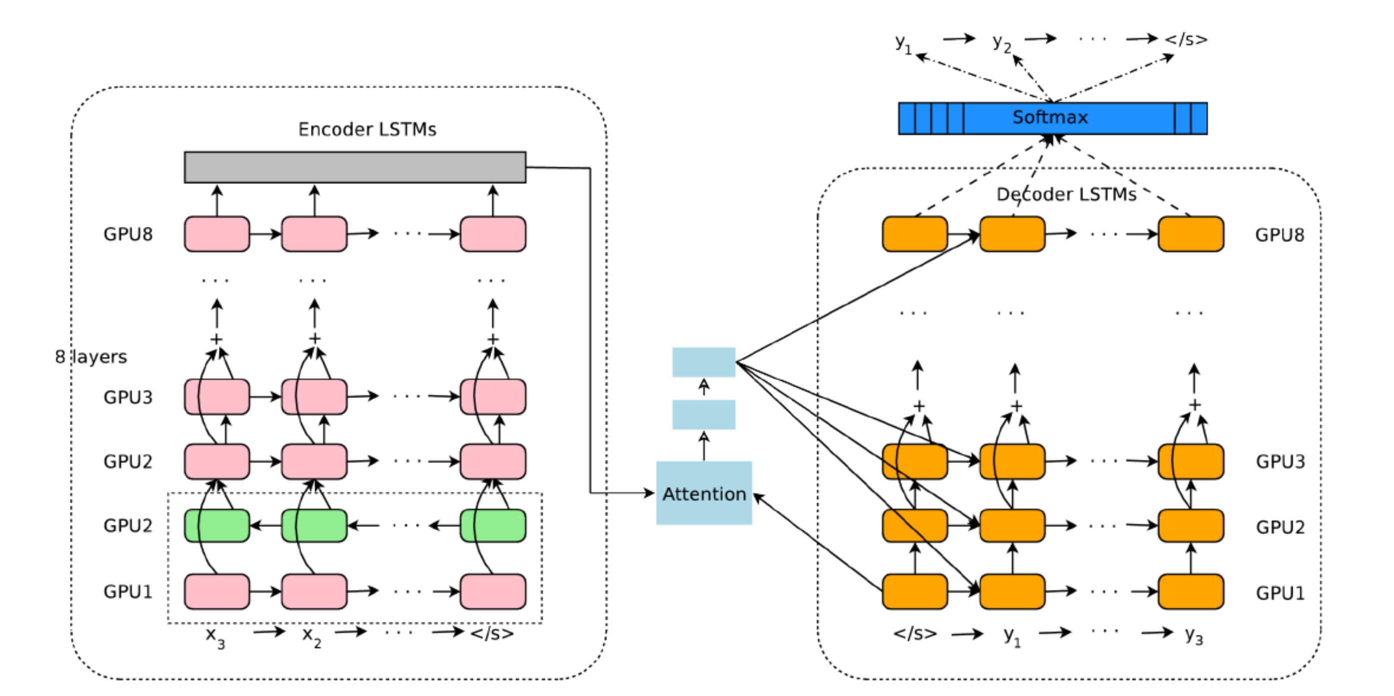
- Google started their Translate service in 2007
- Statistical Machine Translation (SMT) method was used initially
  - A statistical and probabilistic approach
  - Model a sentence as probable word transition
  - Probable translations are obtained from bilingual source texts
  - United Nation and European Parliament documents are the source texts
  - English is used as the intermediate language
- Evolved into NMT model in 2016

Google NMT (1/3)

- *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*
- Yonghui Wu et al.
- Announced in 2016
- Greatly increasing translation accuracy

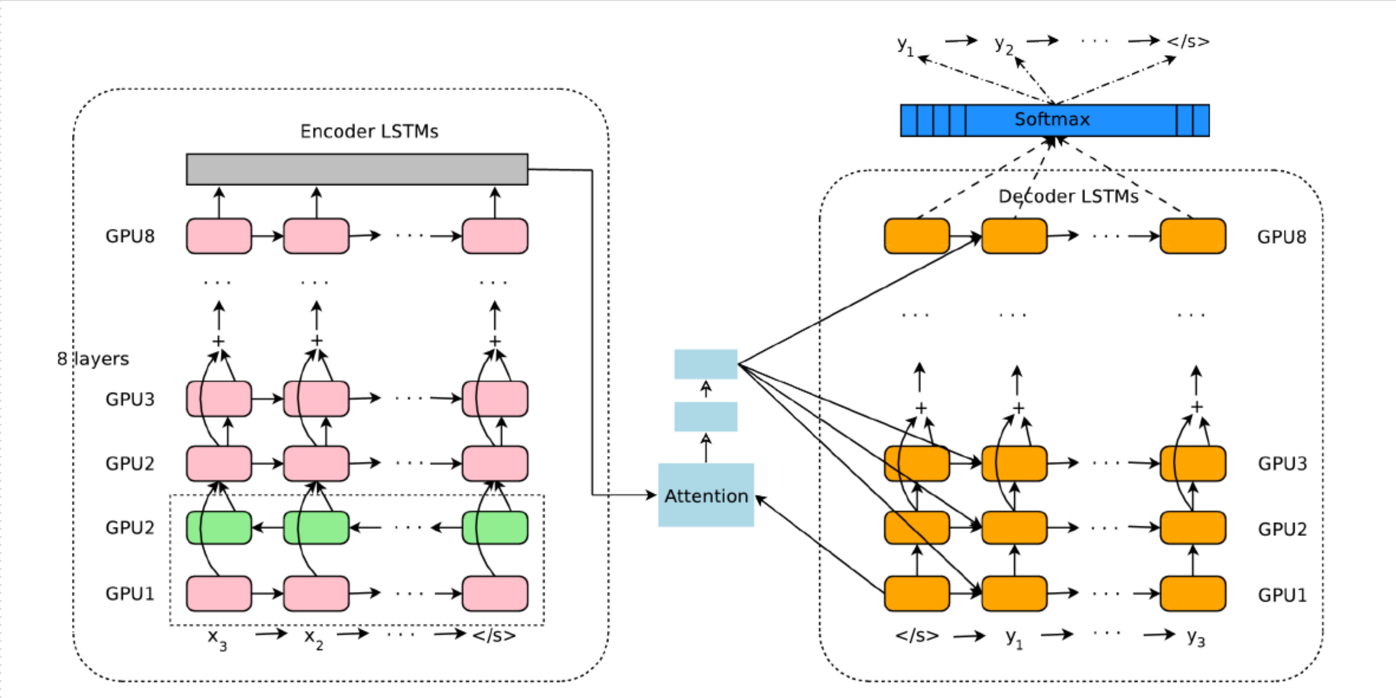
Google NMT (2/3)

- Encoder-Decoder Architecture
  - **Encoder** transform each word in source into vector form
  - **Decoder** select the most fitting word in target language according to the vector form sequence



Google NMT (3/3)

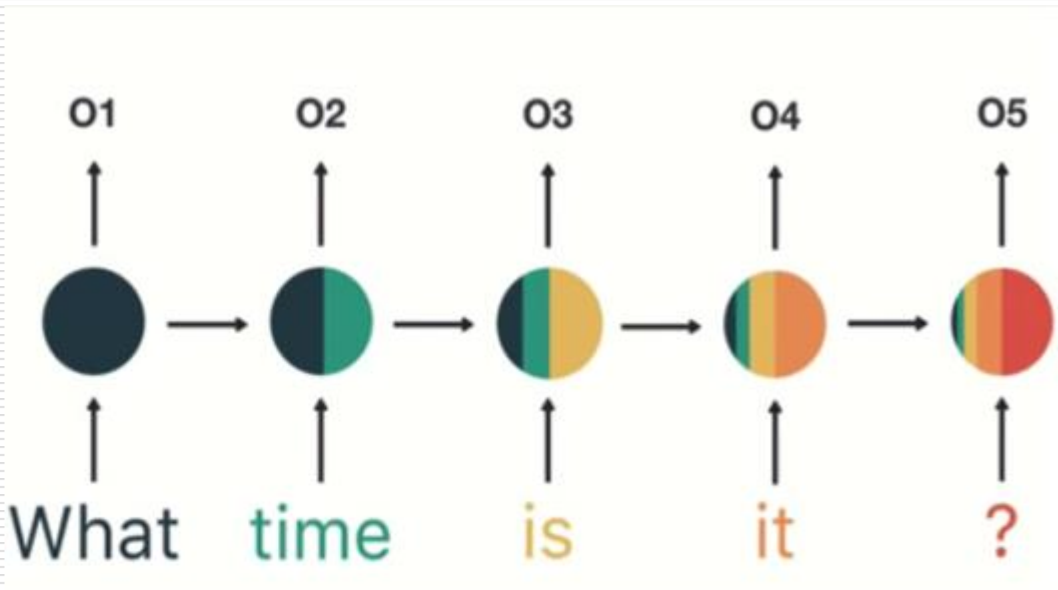
- Encoder-Decoder Architecture
  - 8 layers of LSTM encoder, 8 layers of LSTM decoder
  - Residual connection between LSTM layers
  - Augmented with attention mechanism



Problems and Limitations

Sequential Limitation

- Computation of each token needs to be completed in series to obtain the correct memory update
- Difficult to speedup with additional computation hardware, like more GPU on cluster



Architectural Limitation

- BPTT is very computation-intensive and time-consuming, this is exacerbated by growing model depth and hidden dimension
- Memory is difficult to optimize for both generality and performance
- LSTM and other RNN architecture doesn't show good depth scalability

- We need a better solution for sequential modeling…

Homework

Homework (1/3)

- Please answer the following questions about GRU structure (30%)
  - What are its strength and weakness compared to LSTM?
  - Can we say GRU is an improvement over LSTM? Give your detailed reasoning
    - You are encouraged to read the following paper for insights
    - https://arxiv.org/abs/1412.3555

Homework (2/3)

- How are recurrent neural networks different from other deep learning networks? (10%)

- What are the limitations of recurrent neural networks? (10%)

Homework (3/3)

- Please introduce a subtask of NLP (50%)
  - What is its goal?
  - What common dataset does it use?
  - How to calculate its metric?
  - What are its practical applications in real-life?

- Deadline: 7/29 (Sun.) 23:59
- Filename: HW5_[帳號].pdf
- Submit a PDF report to https://docs.google.com/forms/d/1Xk6M5u-T3EGSTVJ-u7CwHBkOjDud6ijSqRR0-Pt2FcM/edit

References

References

- 台大李宏毅老師機器學習課程
  - https://speech.ee.ntu.edu.tw/~hylee/ml/2016-fall.php
- Annotated History of Modern AI and Deep Learning
  - https://people.idsia.ch/~juergen/deep-learning-history.html
- Recurrent Neural Network
  - https://en.wikipedia.org/wiki/Recurrent_neural_network
- Machine Translation
  - https://en.wikipedia.org/wiki/Machine_translation
- Sequence Models and Long Short-Term Memory Networks (PyTorch Tutorial)
  - https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

References

- Long Short-Term Memory
  - https://www.bioinf.jku.at/publications/older/2604.pdf
- Gated Recurrent Unit
  - https://arxiv.org/abs/1406.1078
- Google's Neural Machine Translation System
  - https://arxiv.org/abs/1609.08144

Thank you