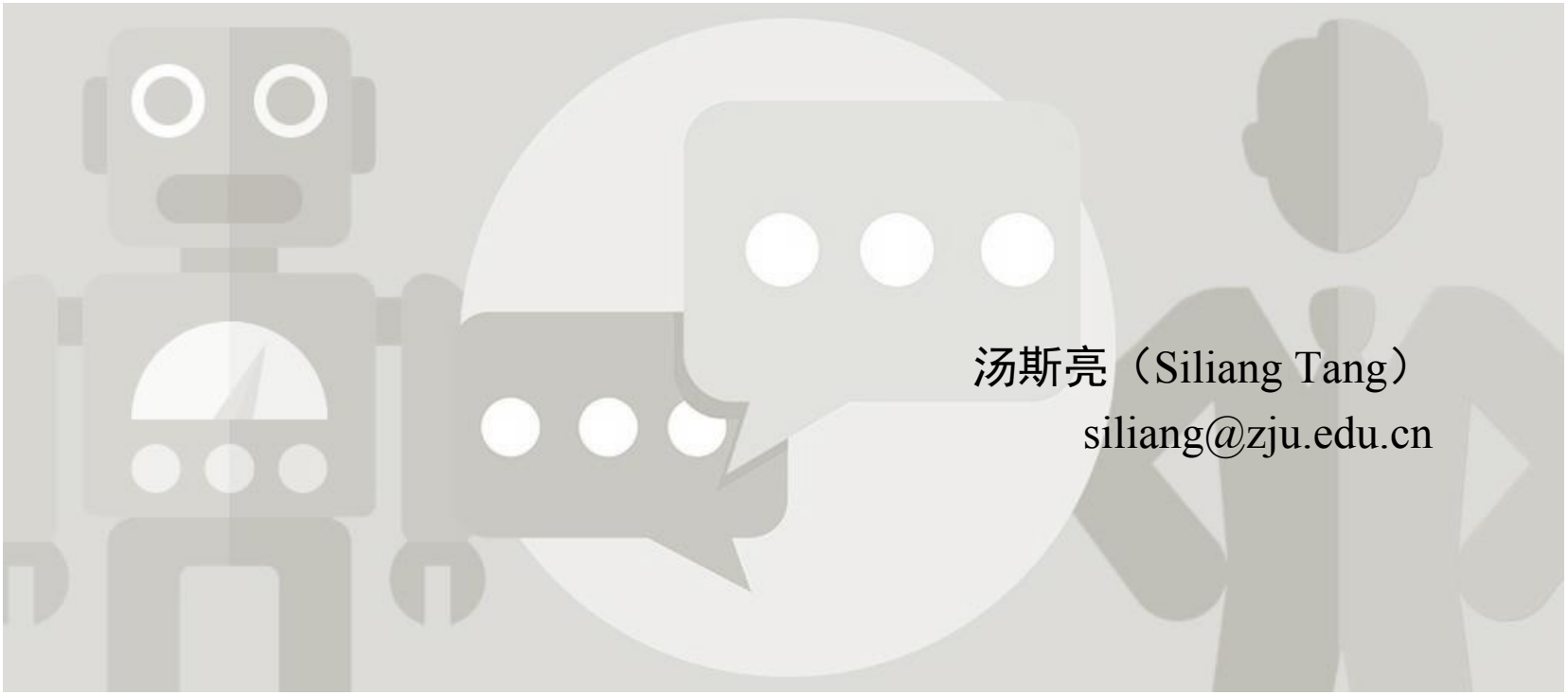
Recurrent Neural Networks



汤斯亮（Siliang Tang）
siliang@zju.edu.cn

Variants of Neural Networks

Convolutional Neural Network (CNN)

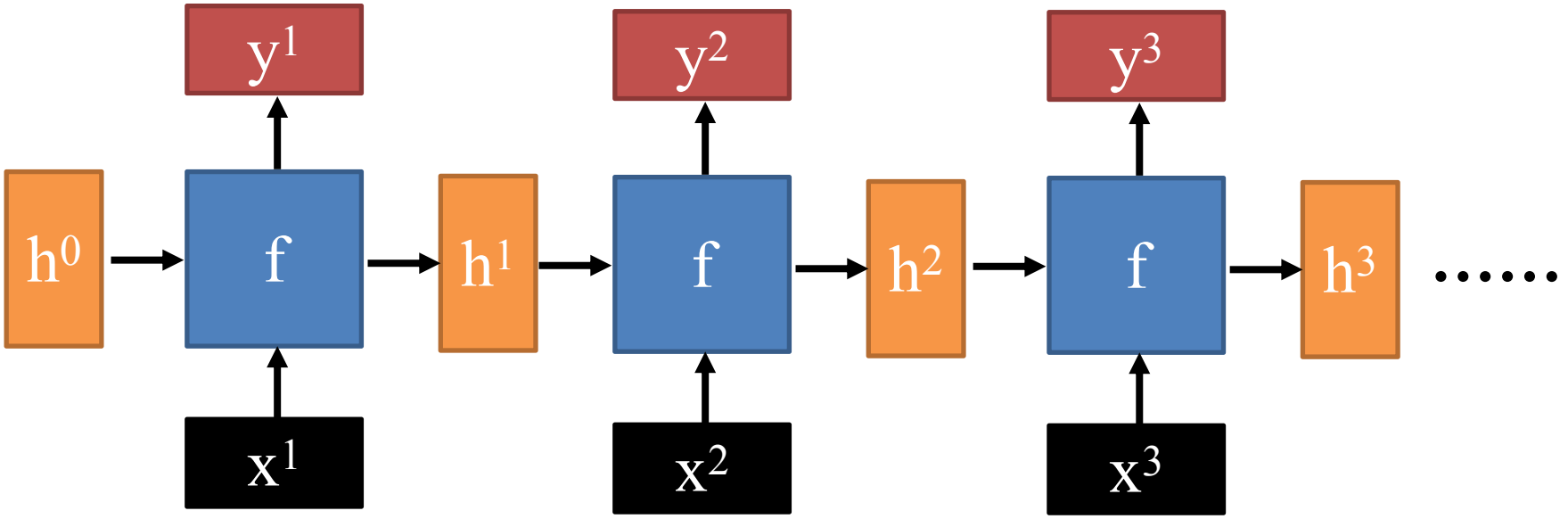
Recurrent Neural Network (RNN)

Neural Network with Memory

Recurrent Neural Network

- Given function f: $h', y = f(h, x)$

h and h' are vectors with the same dimension

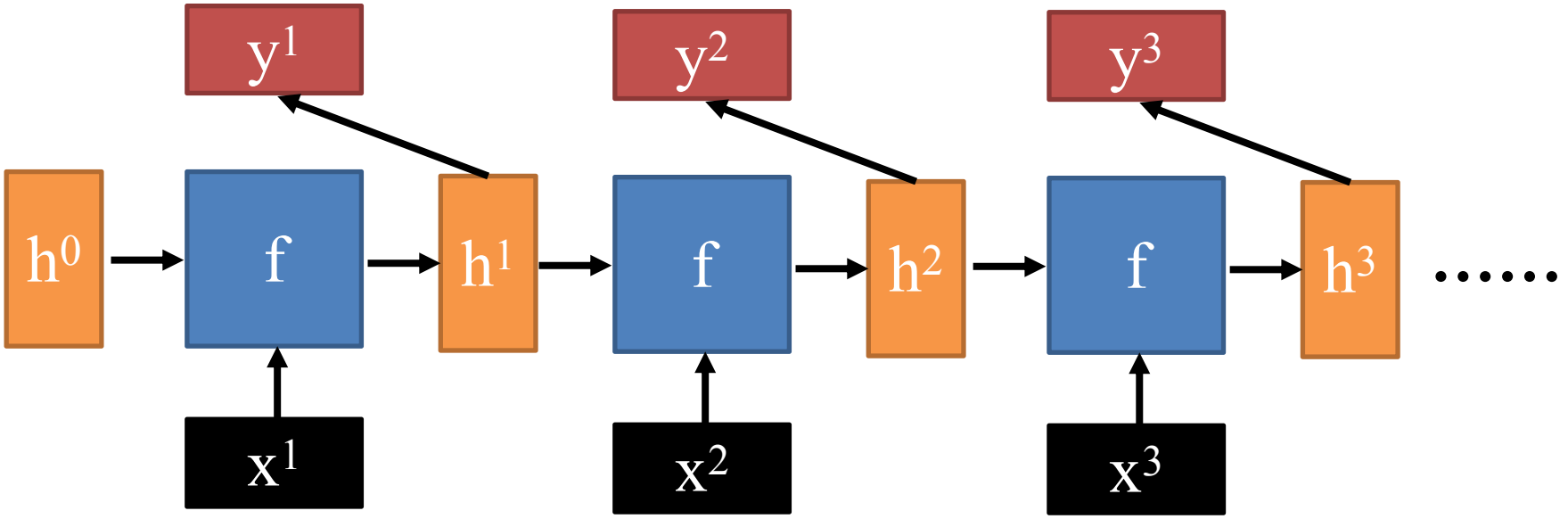


No matter how long the input/output sequence is, we only need one function f

Recurrent Neural Network

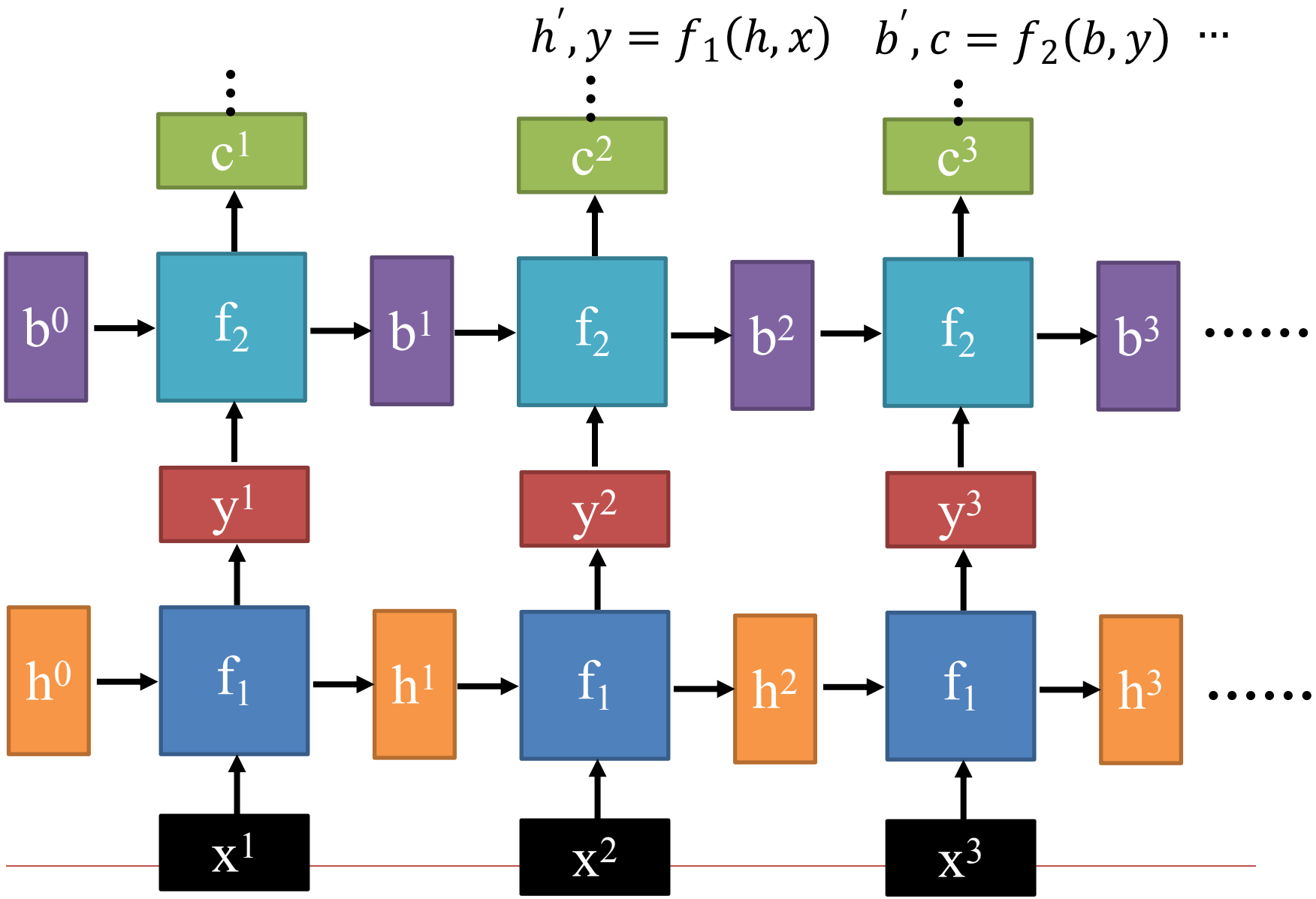
- Given function f: $h', y = f(h, x)$

h and h' are vectors with the same dimension



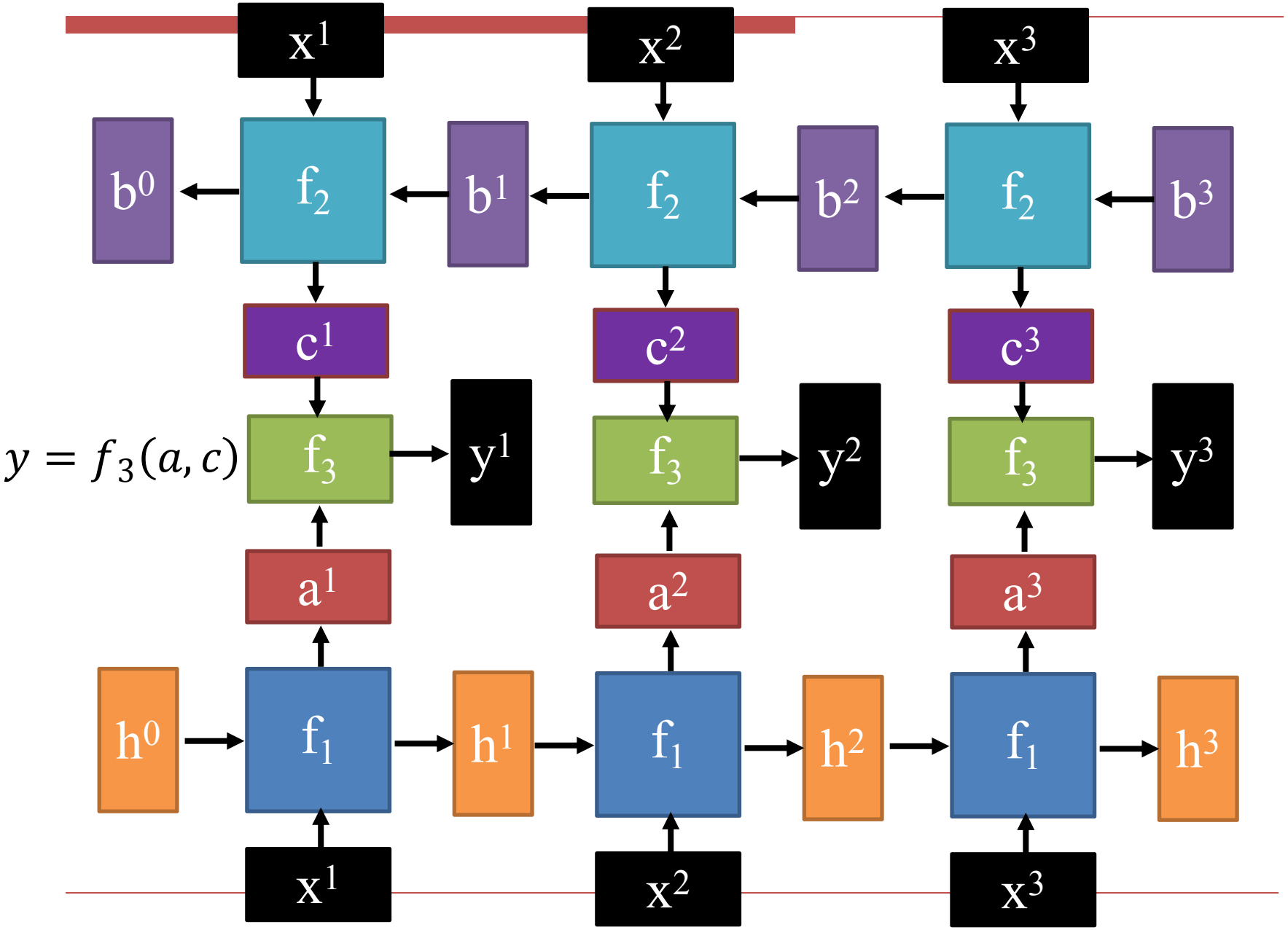
No matter how long the input/output sequence is, we only need one function f

Deep RNN


$h', y = f_1(h, x)$
$b', c = f_2(b, y)$
...
c^1
c^2
c^3
b^0
f_2
b^1
f_2
b^2
f_2
b^3
.....
y^1
y^2
y^3
h^0
f_1
h^1
f_1
h^2
f_1
h^3
.....
x^1
x^2
x^3

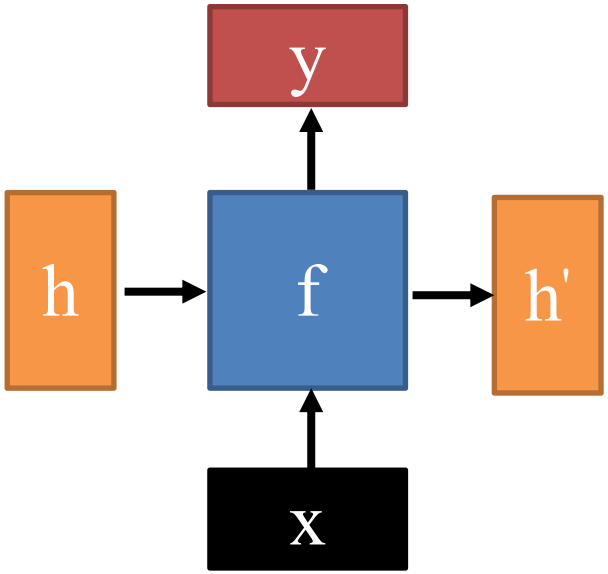
Bidirectional RNN

$h', a = f_1(h, x)$ $b', c = f_2(b, x)$



Naïve RNN

- Given function f: $h', y = f(h, x)$



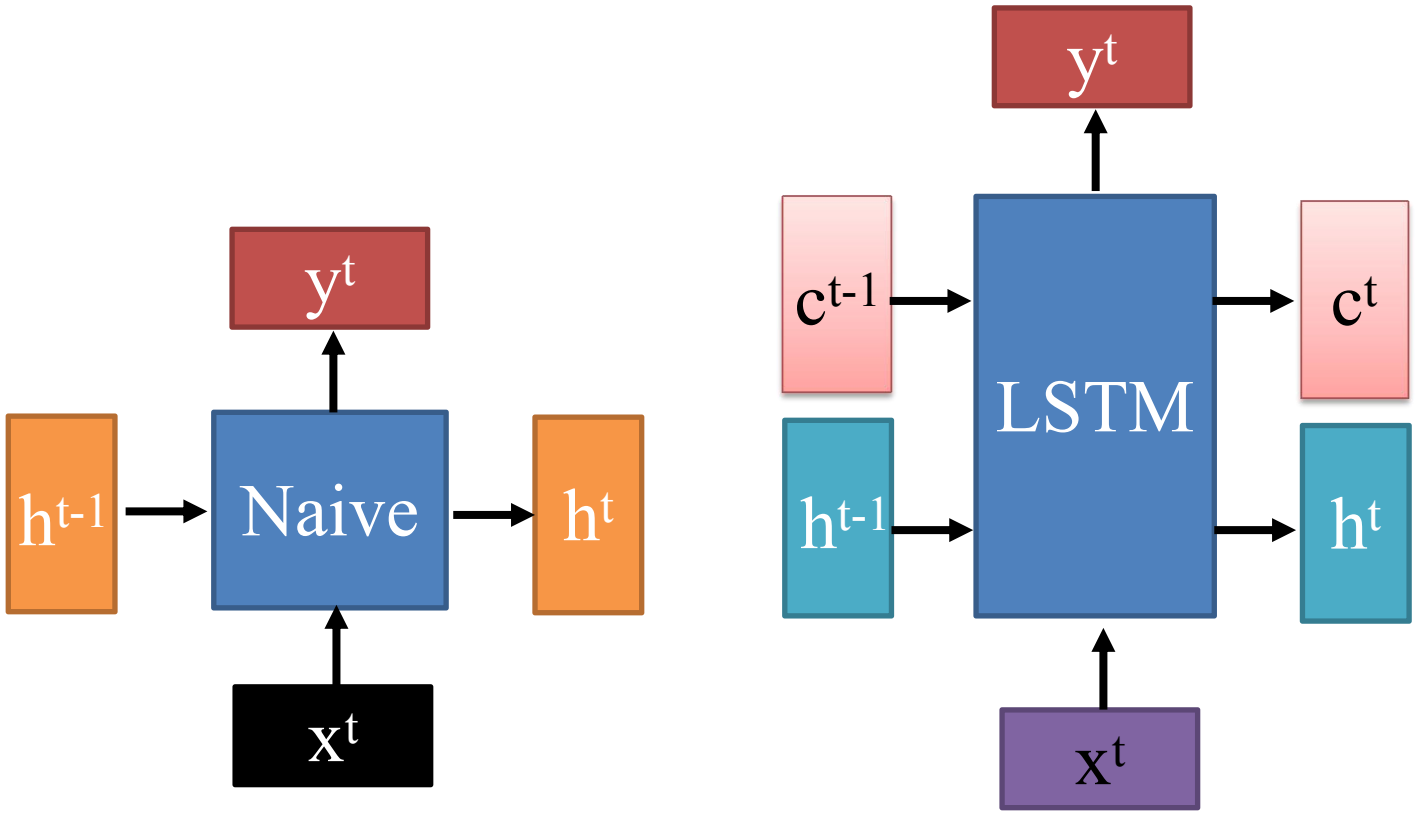
$$h' = \sigma(W^h h + W^i x)$$

$$y = \sigma(W^o h')$$

softmax

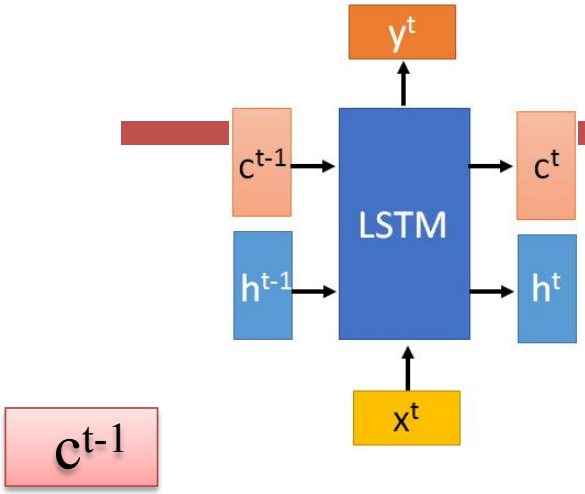
Ignore bias here

LSTM



c changes slowly ➡ c^t is c^{t-1} added by something

h changes faster ➡ h^t and h^{t-1} can be very different



$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

$$z^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

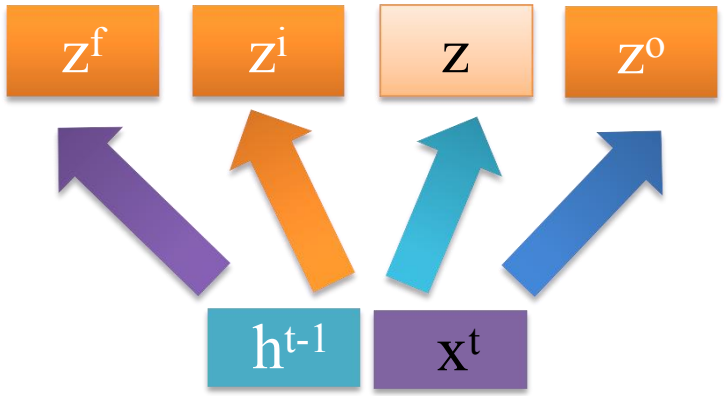
Input gate

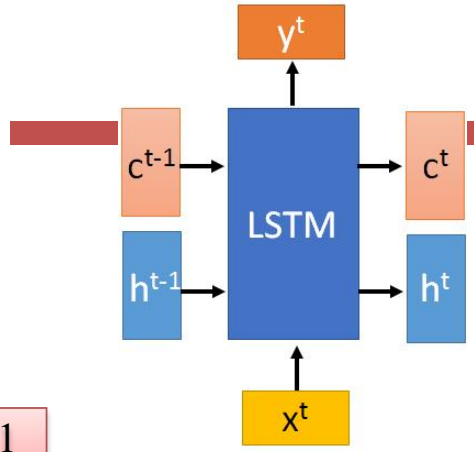
$$z^f = \sigma\left(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

forget gate

$$z^o = \sigma\left(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

output gate





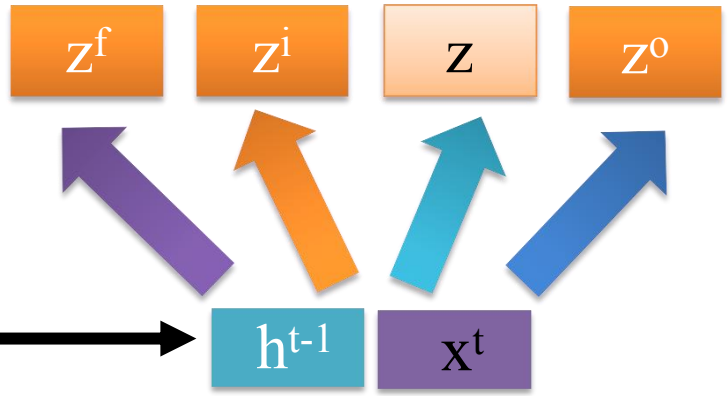
$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \\ c^{t-1} \end{bmatrix} \right)$$

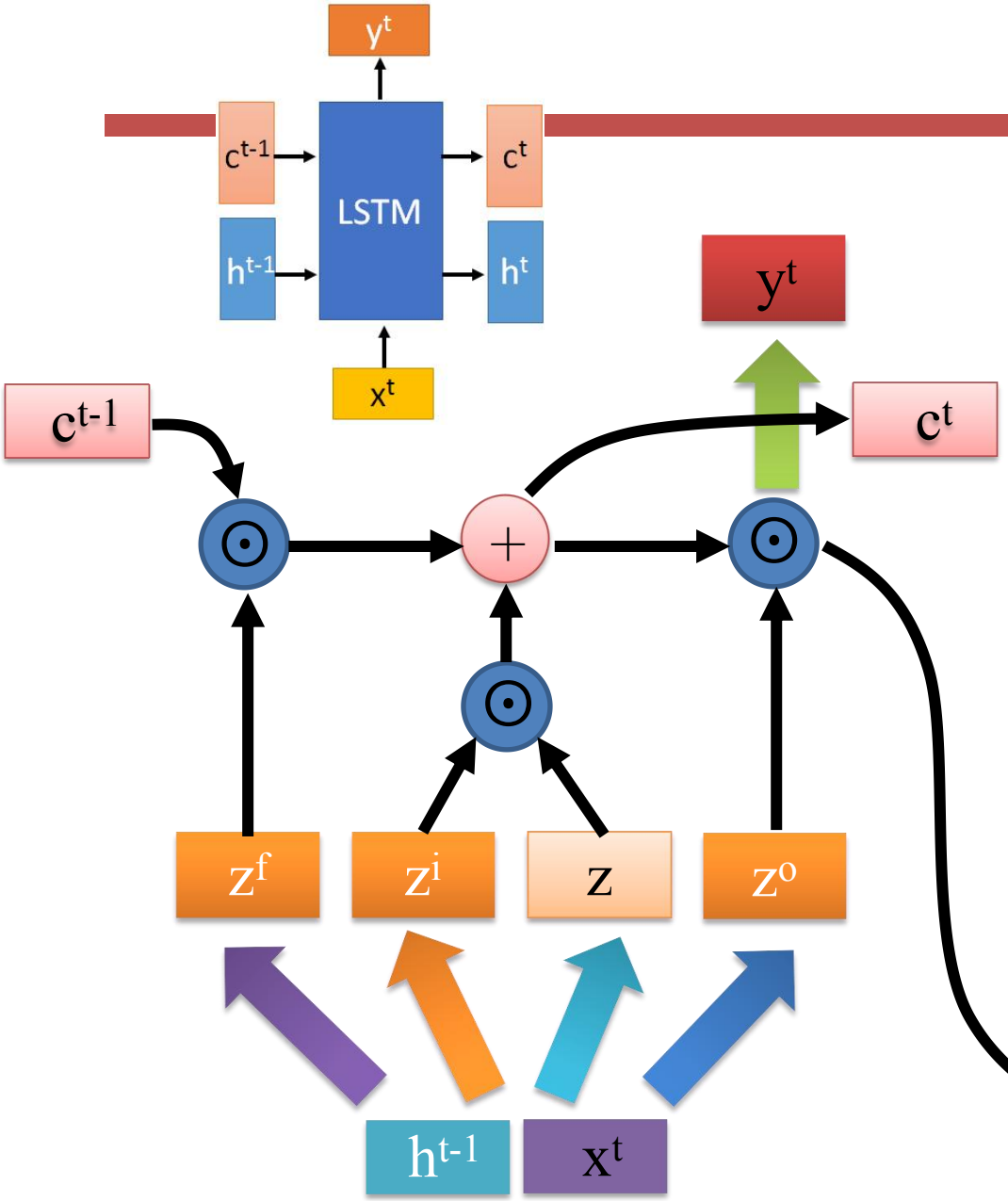
diagonal

c^{t-1}

"peephole"

z^o z^f z^i obtained by the same way



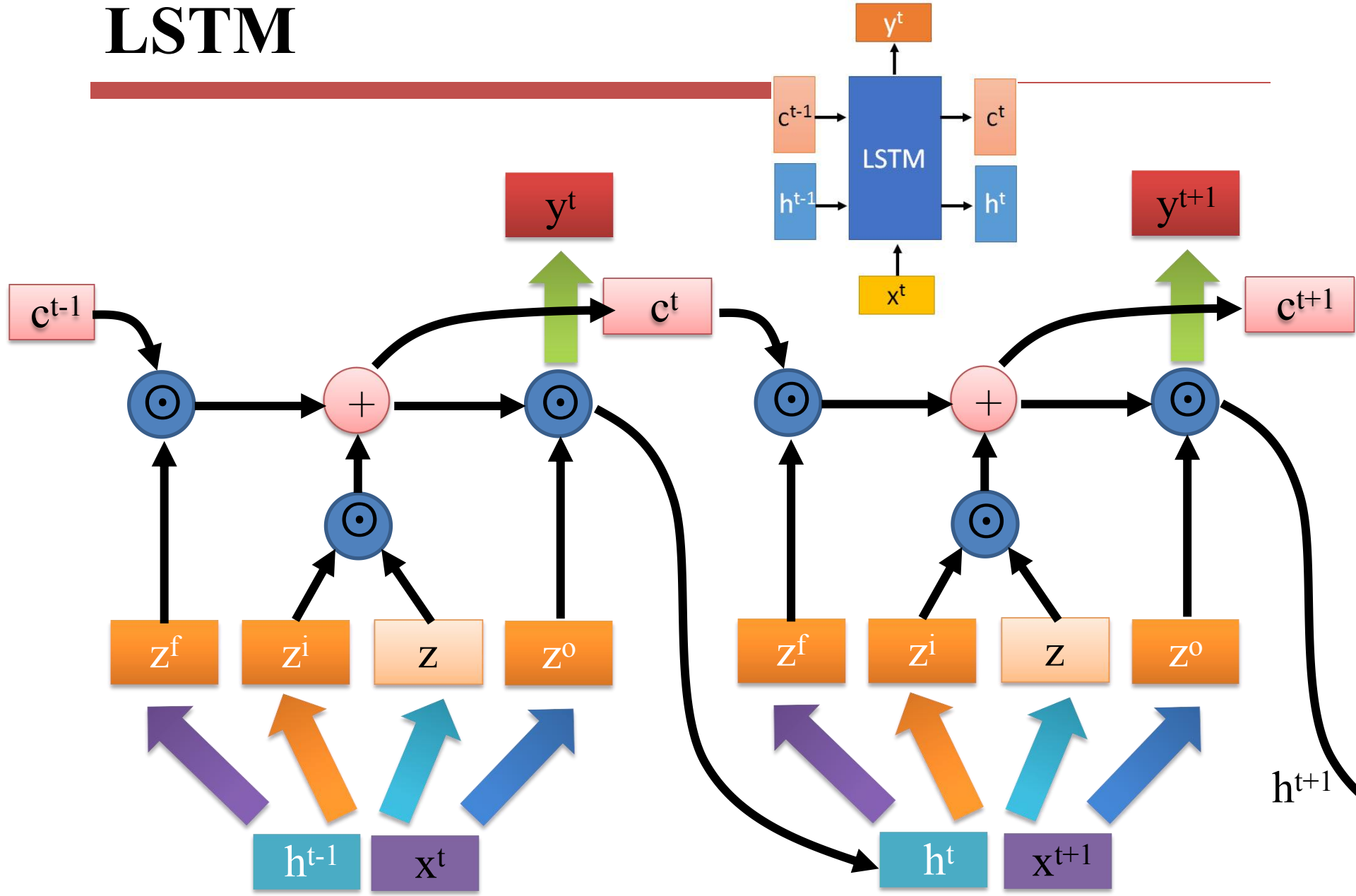


$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

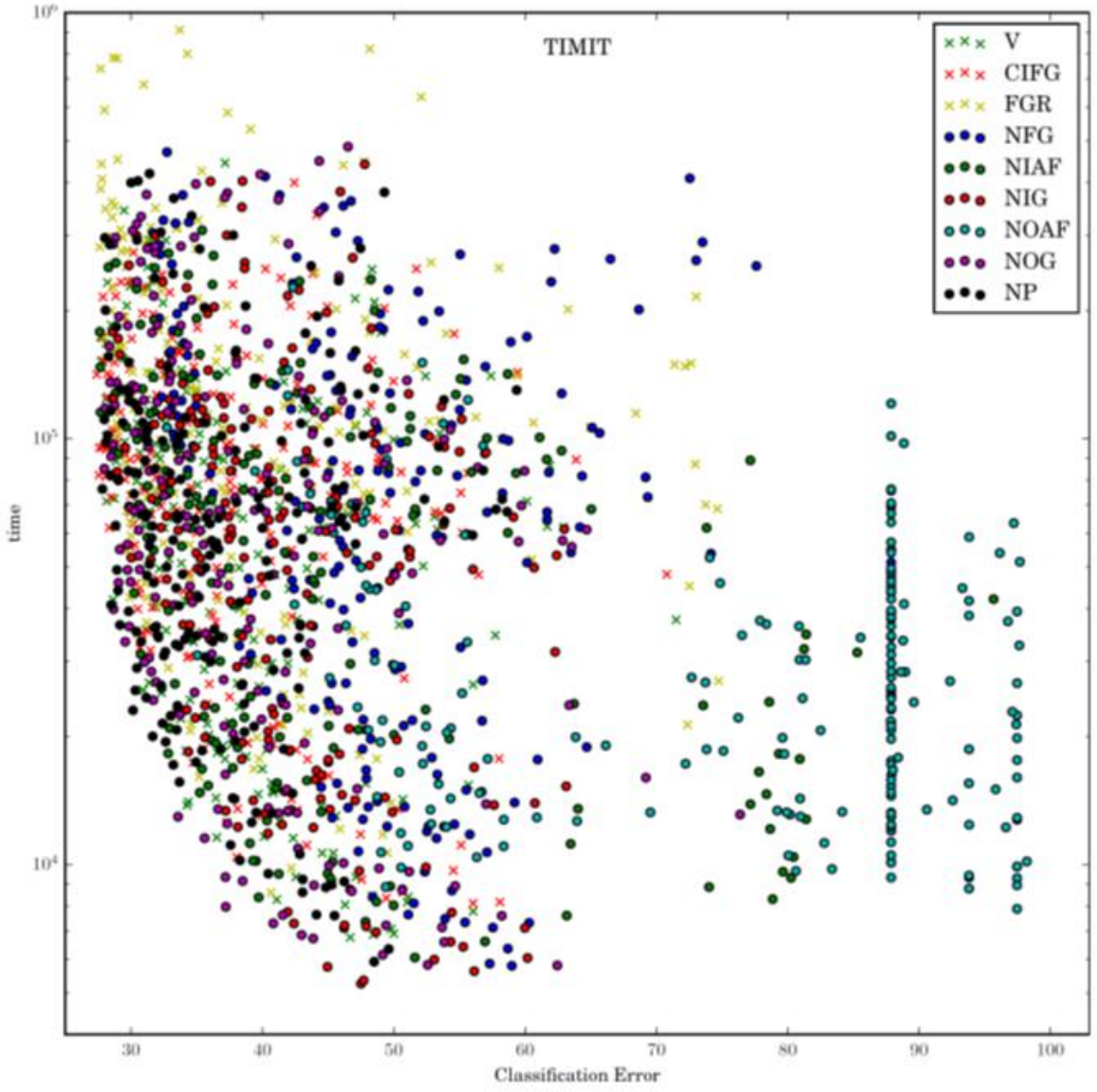
$$h^t = z^o \odot tanh(c^t)$$

$$y^t = \sigma(W' h^t)$$

LSTM

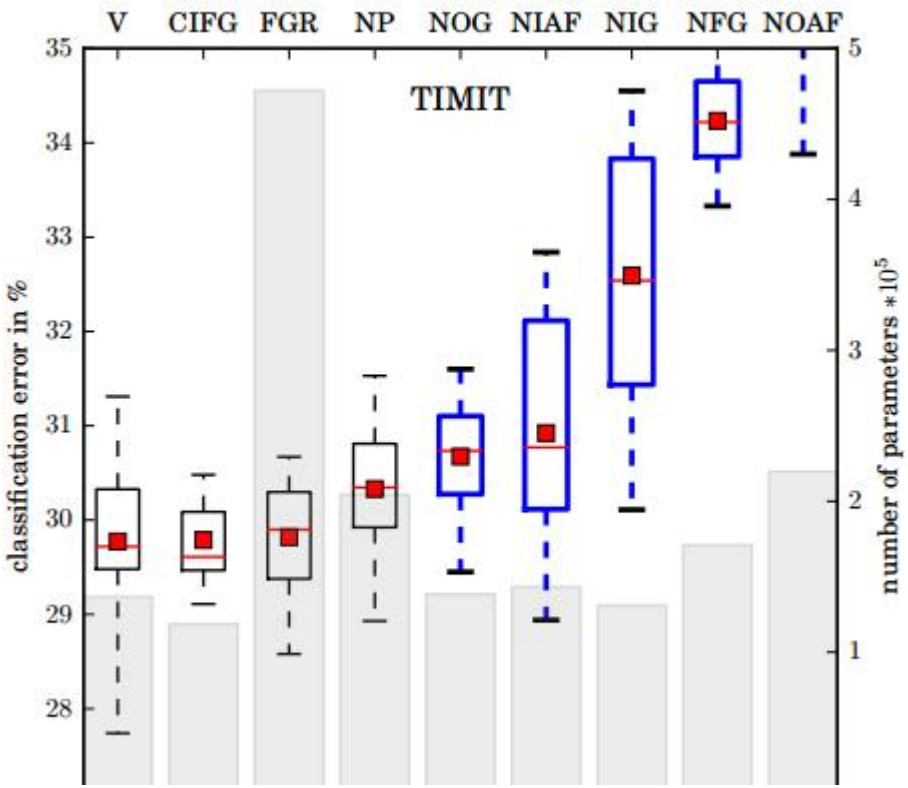

y^t
c^{t-1}
LSTM
c^t
h^{t-1}
h^t
x^t
y^t
y^{t+1}
c^{t-1}
c^t
c^{t+1}
z^f
z^i
z
z^o
z^f
z^i
z
z^o
h^{t-1}
x^t
h^t
x^{t+1}
h^{t+1}

LSTM: A Search Space Odyssey



LSTM: A Search Space Odyssey

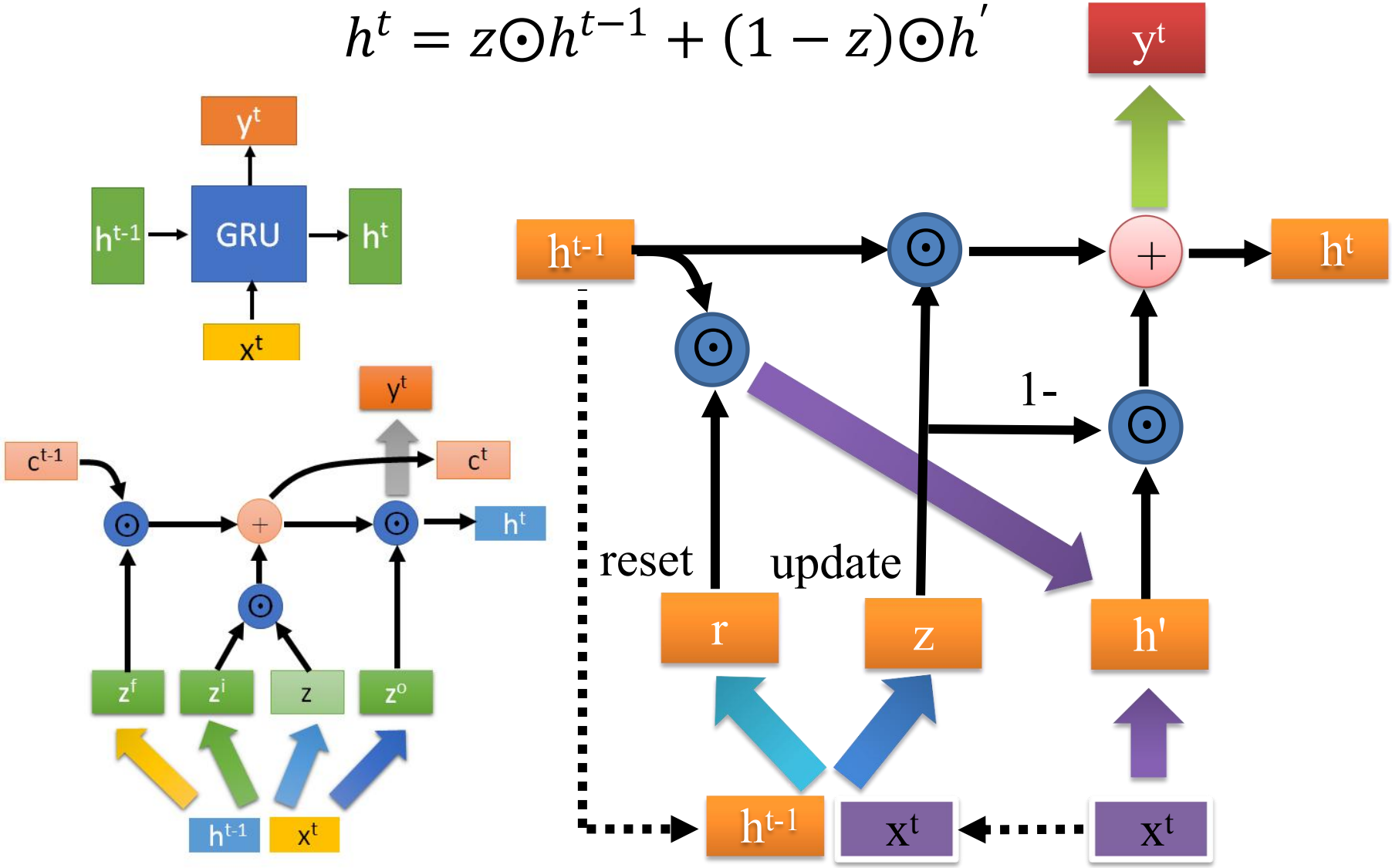
1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)



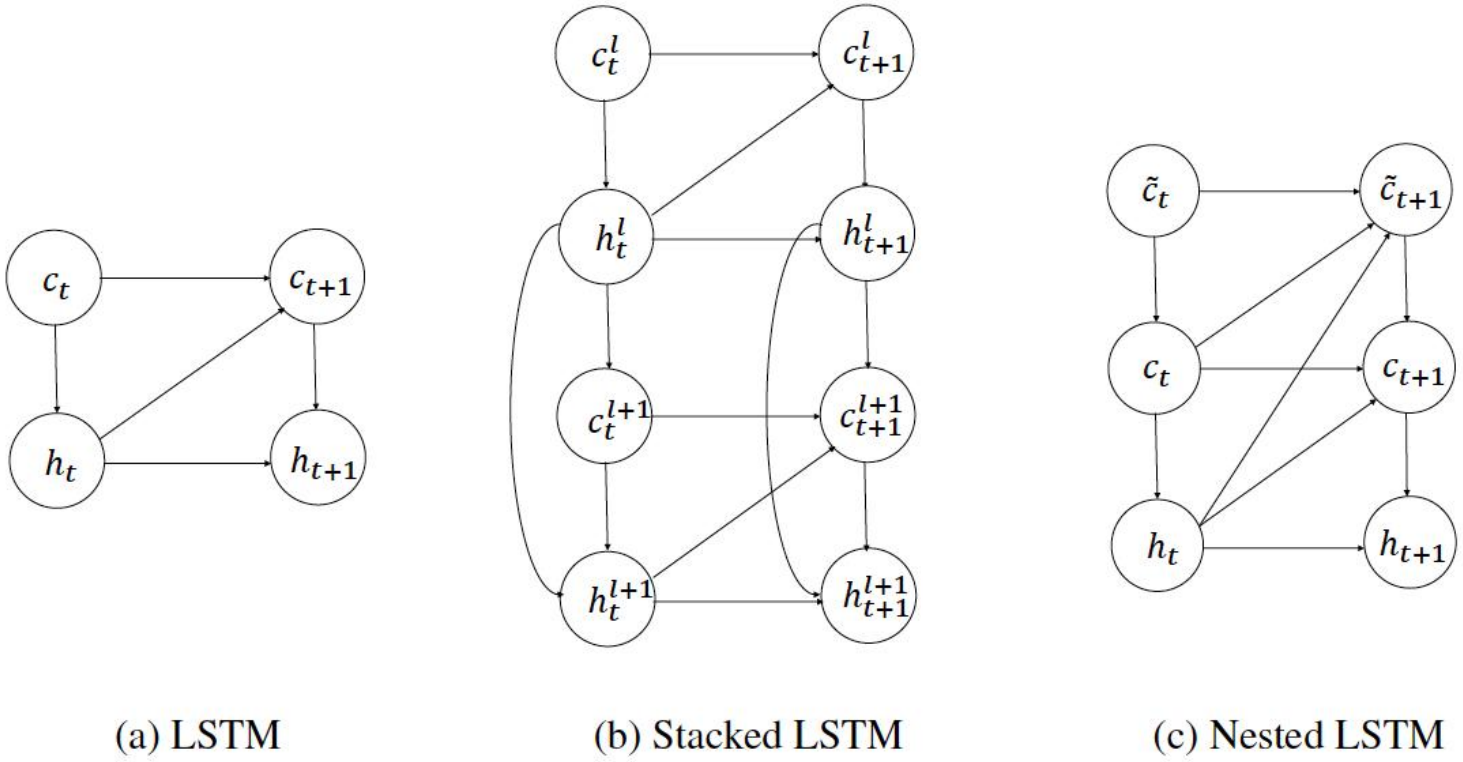
Standard LSTM works well
Simply LSTM: coupling input and forget gate, removing peephole
Forget gate is critical for performance
Output gate activation function is critical

GRU

$$h^t = z \odot h^{t-1} + (1 - z) \odot h'$$



Nested LSTM



(a) LSTM (b) Stacked LSTM (c) Nested LSTM

Moniz J R A, Krueger D. Nested lstms[J]. arXiv preprint arXiv:1801.10308, 2018.

RNN for NLP

Long-distance Dependencies

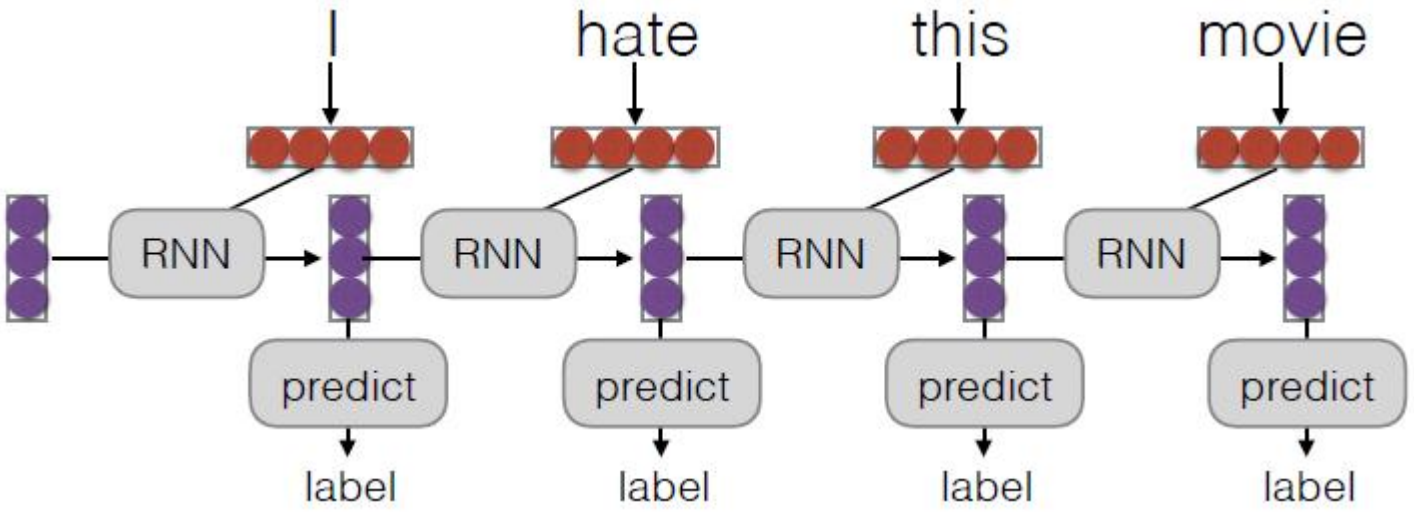
- NLP is full of sequential data
  - Words in sentences
  - Characters in words
  - Sentences in discourse
  - …
- RNNs is good at capturing long-distance dependencies. E.g.,

1. Agreement in number, gender, etc.
   - **He** does not have very much confidence in **himself**.
   - **She** does not have very much confidence in **herself**.
2. Selectional preference
   - The **reign** has lasted as long as the life of the **queen**.
   - The **rain** has lasted as long as the life of the **clouds**.
3. What is the referent of "it"?
   - The **trophy** would not fit in the brown suitcase because it was too **big**.
   - The trophy would not fit in the brown **suitcase** because it was too **small**.

From Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html

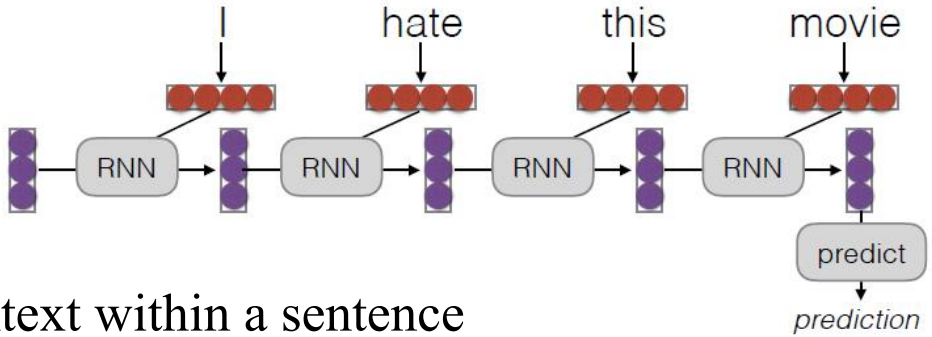
Unrolling in Time

- What does processing a sequence look like?

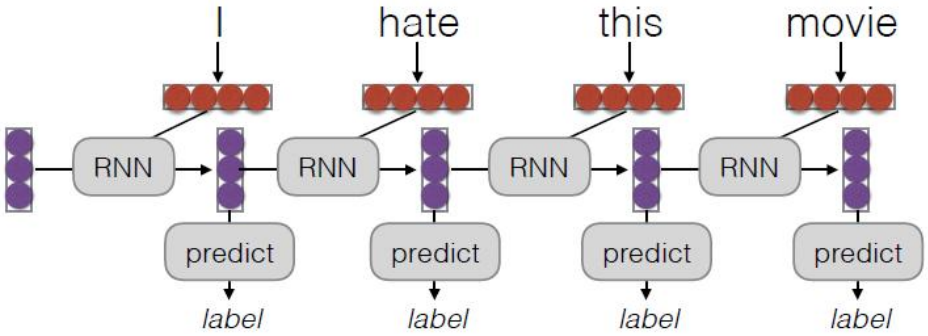


What Can RNNs Do?

- Represent a sentence
  - Read whole sentence, make a prediction
  - Sentence classification, Conditioned generation, Retrieval



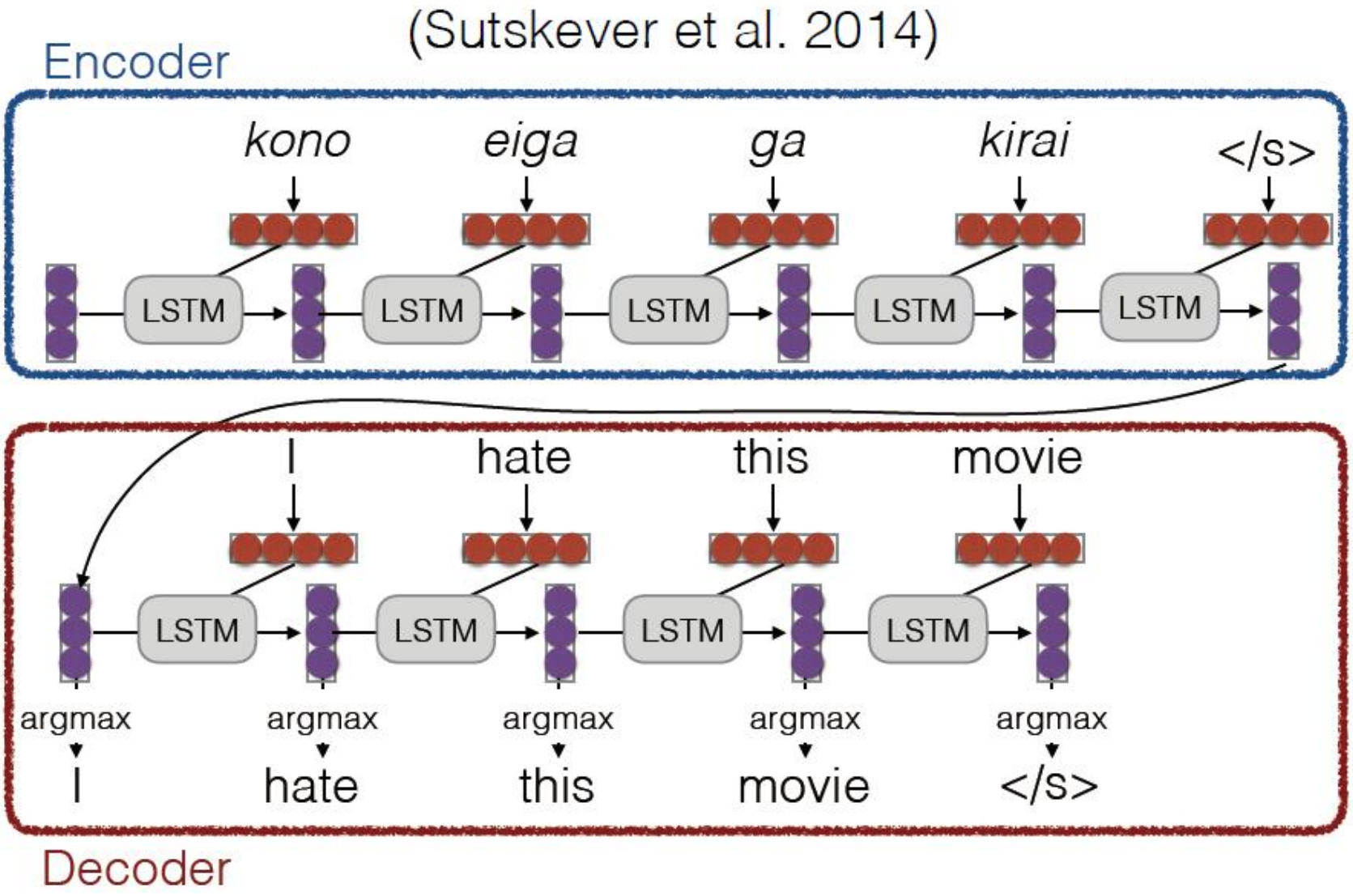
- Represent a context within a sentence
  - Read context up until that point
  - Tagging, Language Modeling, Calculating Representations for Parsing, etc.



What can LSTMs Learn?

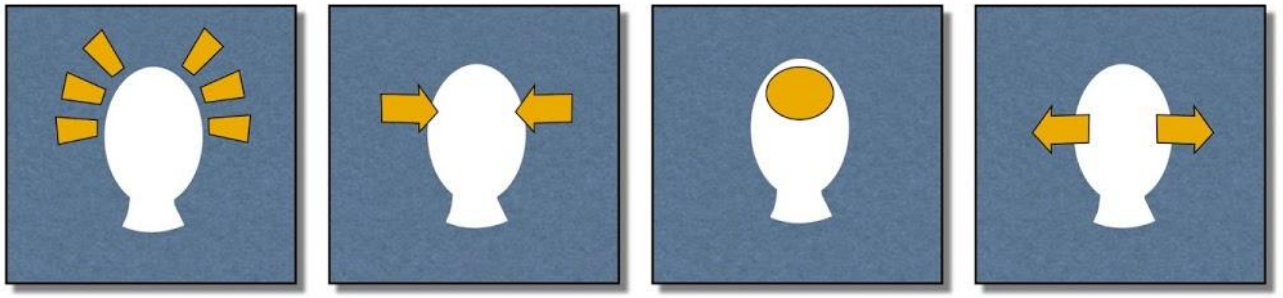
http://lstm.seas.harvard.edu/

Encoder-decoder Models



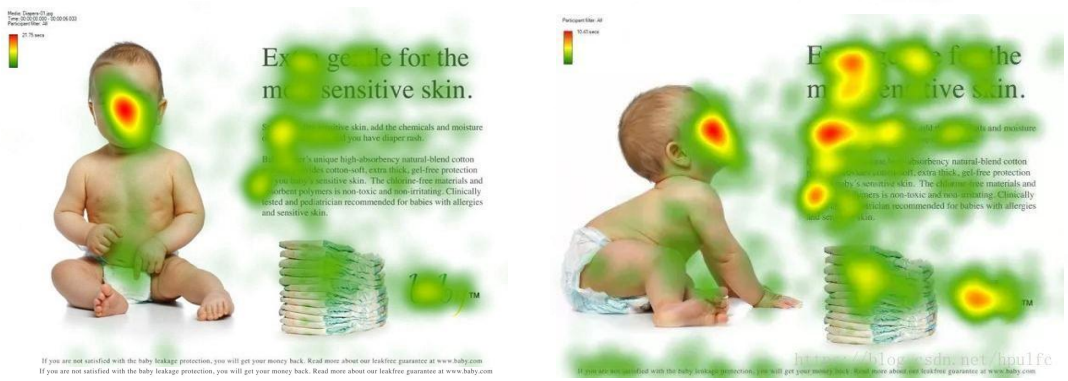
How People Learn:

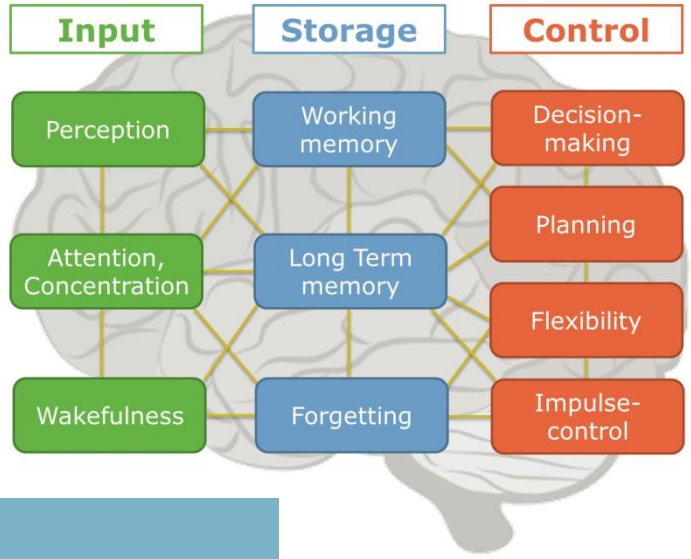
Four cognitive processes every teacher should know

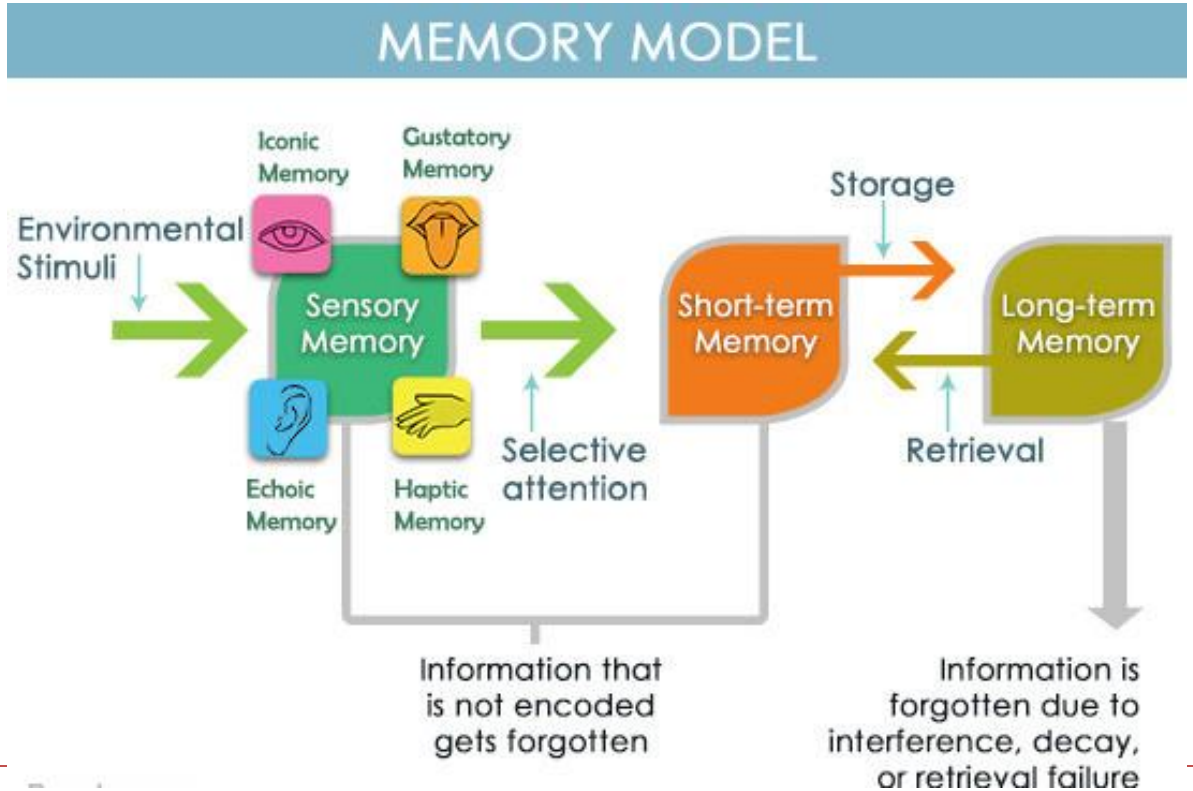


Attention

Attention




Input
Storage
Control
Perception
Working memory
Decision-making
Planning
Attention, Concentration
Long Term memory
Flexibility
Wakefulness
Forgetting
Impulse-control

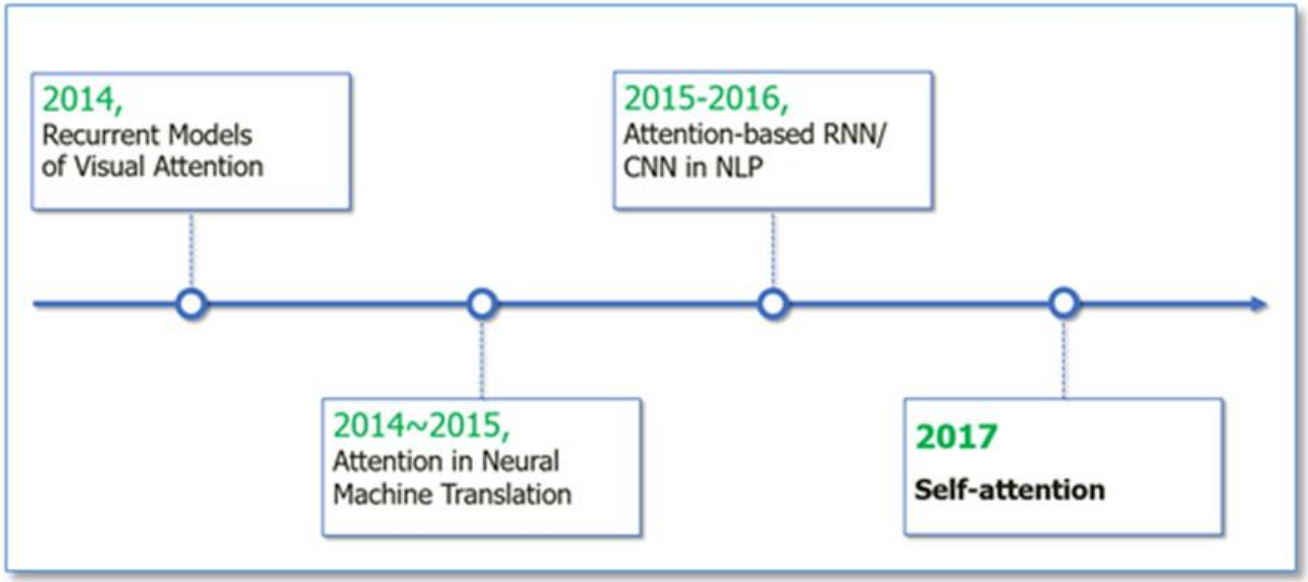

MEMORY MODEL
Iconic Memory
Gustatory Memory
Environmental Stimuli
Sensory Memory
Storage
Short-term Memory
Long-term Memory
Selective attention
Retrieval
Echoic Memory
Haptic Memory
Information that is not encoded gets forgotten
Information is forgotten due to interference, decay, or retrieval failure

Attention

Visual Attention

- Human perception is that one does not tend to process a whole scene in its entirety at once.
- Humans focus attention selectively on parts of the visual space to acquire information when and where it is needed,
- and combine information from different fixations over time to build up an internal representation of the scene, guiding future eye movements and decision making.

Mnih V, Heess N, Graves A. Recurrent models of visual attention. Advances in Neural Information Processing Systems. 2014: 2204-2212.



Attention

- Encode each word in the sentence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
- Use this combination in picking the next word

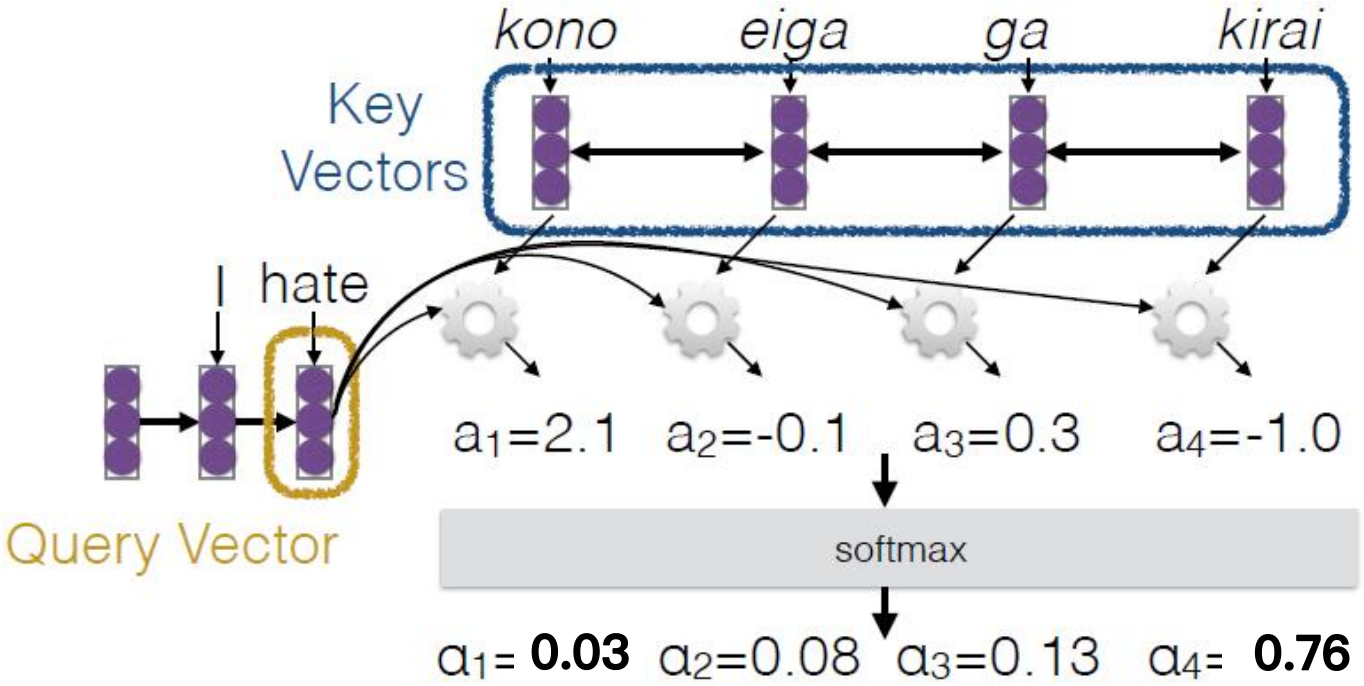
A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

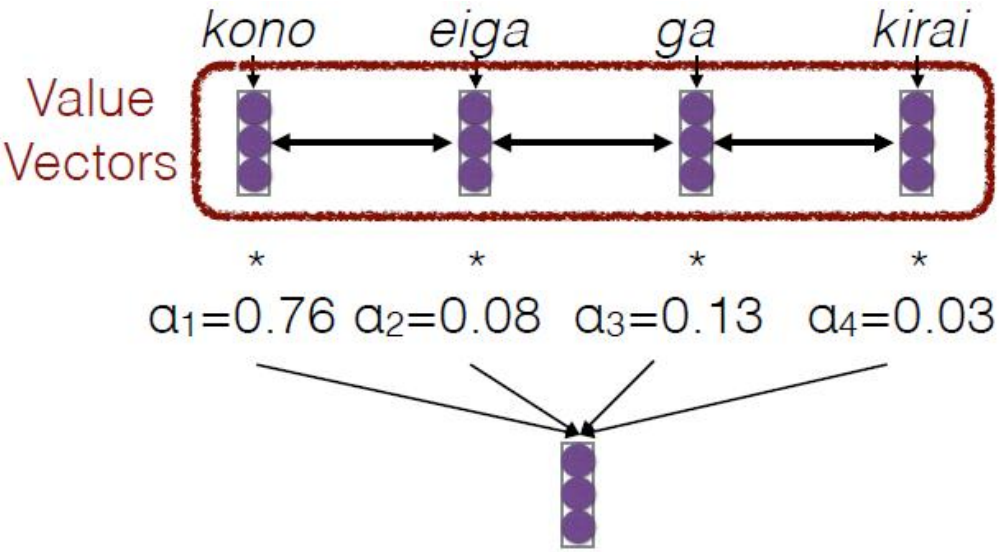
Calculating Attention (1)

- Use "query" vector (decoder state) and "key" vectors (all encoder states)
- For each query-key pair, calculate weight
- Normalize to add to one using softmax



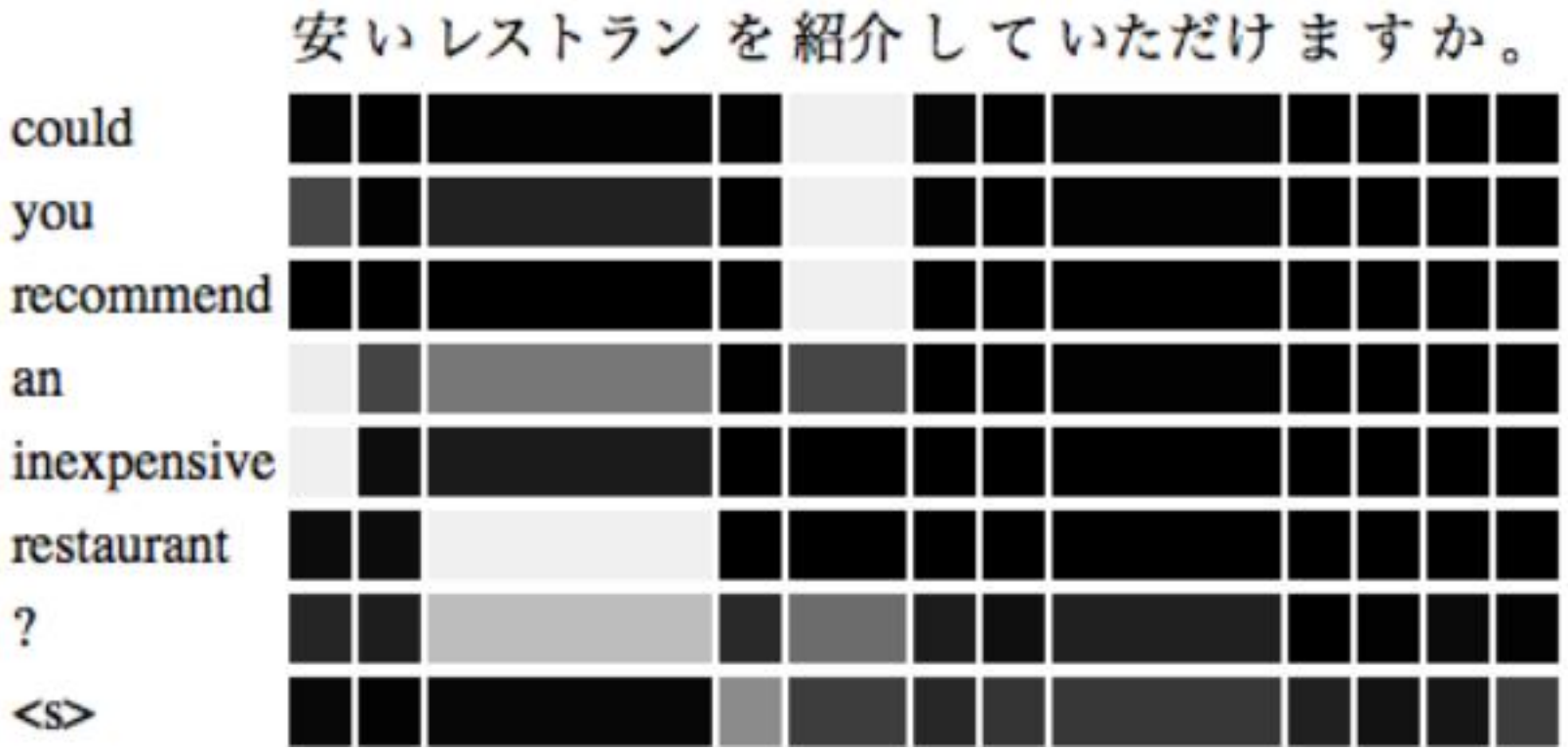
Calculating Attention (2)

- Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



- Use this in any part of the model you like

A Graphical Example



Attention Score Functions

Note: q is the query and k is the key

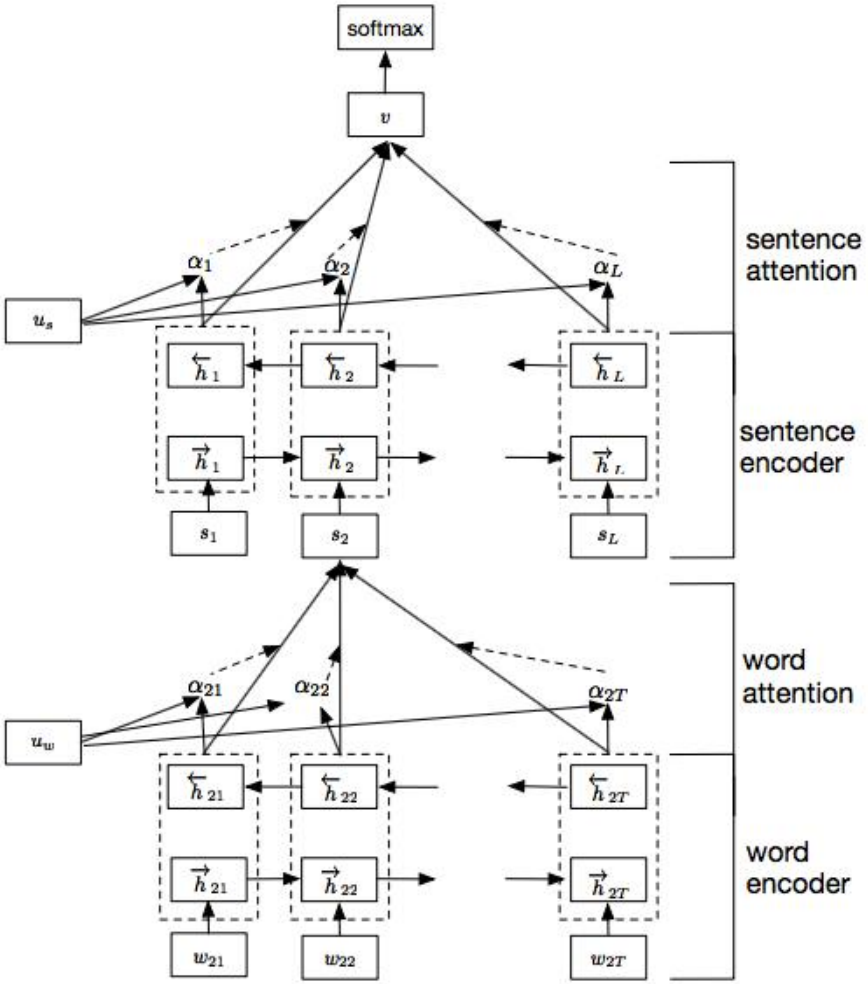
- **Multi-layer Perceptron** (Bahdanau et al. 2015)
  - $a(q,k) = w_2^T \tanh(W_1[q:k])$
  - Flexible, often very good with large data
- **Bilinear** (Luong et al. 2015)
  - $a(q,k) = q^T W k$
- **Dot Product** (Luong et al. 2015)
  - $a(q,k) = q^T k$
  - No parameters! **But requires sizes to be the same**.
- **Scaled Dot Product** (Vaswani et al. 2017)
  - $a(q,k) = \frac{q^T k}{\sqrt{|k|}}$
  - Problem: scale of dot product increases as dimensions get larger
  - Fix: scale by size of the vector

Attention Variants

- Copying mechanism [Gu et al. 2016]
- Attend to the previous words (input, output) [Merity et al. 2016]
- Attend to multi-modal inputs (image, speech) [Xu et al. 2015; Chan et al. 2015;…]
- Attend to multiple sources [Zoph et al. 2015;…]
- …

Hierarchical Structures

- Encode with attention over each sentence, then attention over each sentence in the document

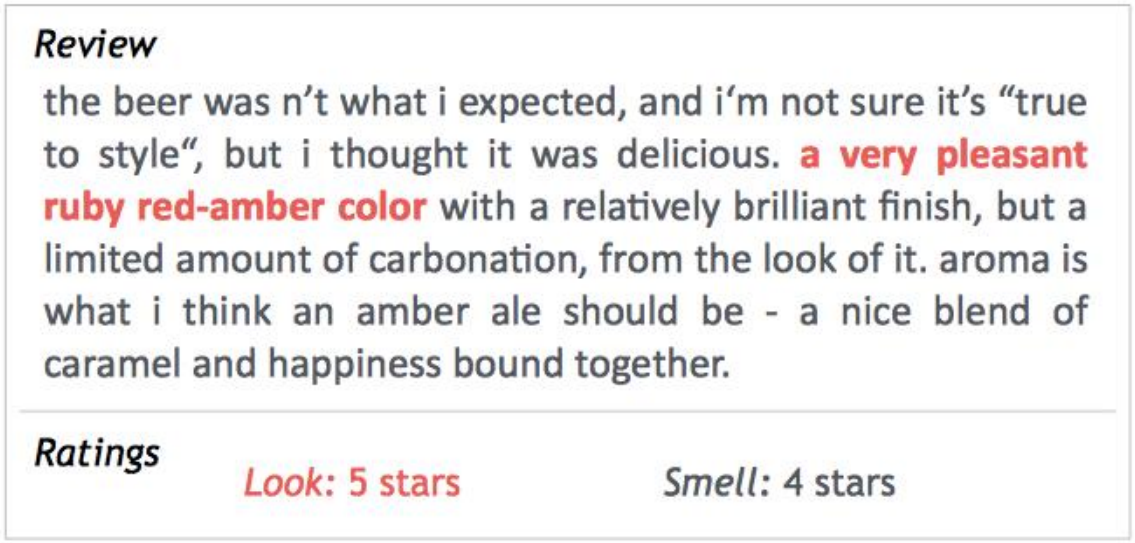


[Yang et al. 2016]

Hard Attention

- Instead of a soft interpolation, make a zero-one decision about where to attend (Xu et al. 2015)
  - Harder to train, requires methods such as reinforcement learning

- Perhaps this helps interpretability? (Lei et al. 2016)

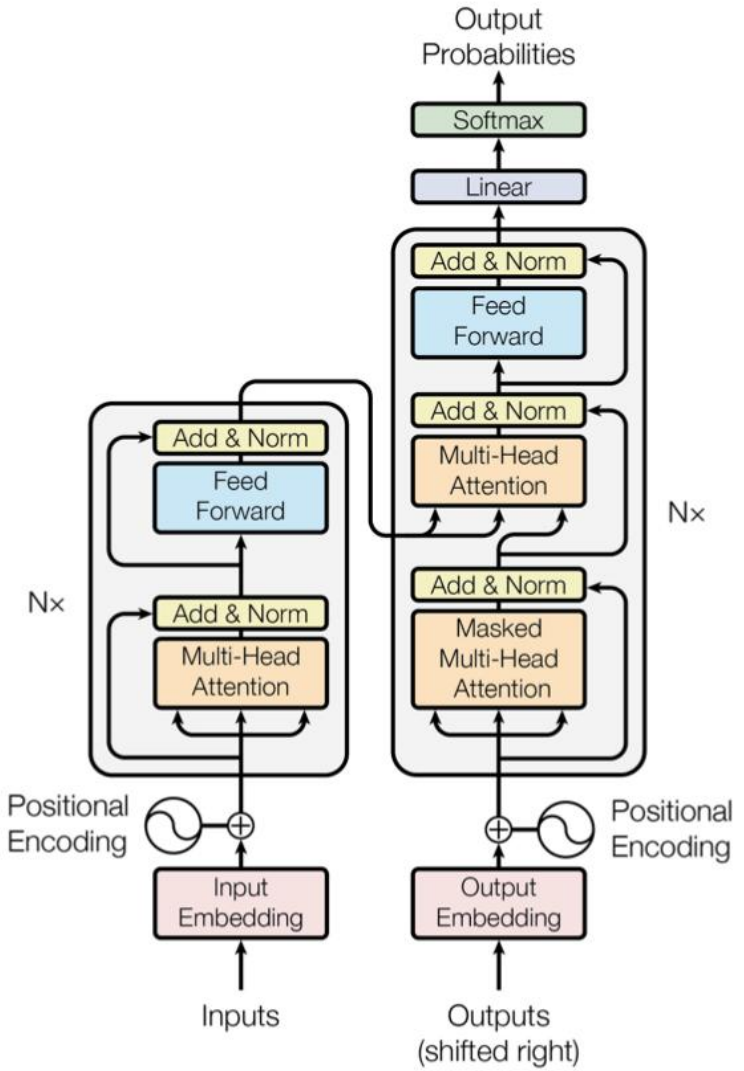


Attention is All You Need

- Transformer
  - A sequence-to-sequence model based entirely on attention
  - Strong results on standard WMT datasets
  - Fast: only matrix multiplications

The Transformer - model architecture

- Encoder and Decoder Stacks
  - **Attention**
  - Position-wise Feed-Forward Networks
  - **Positional Encoding**
  - Add & Norm

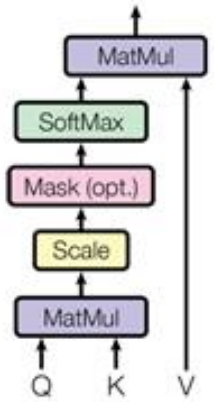


Multi-headed Attention

Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention



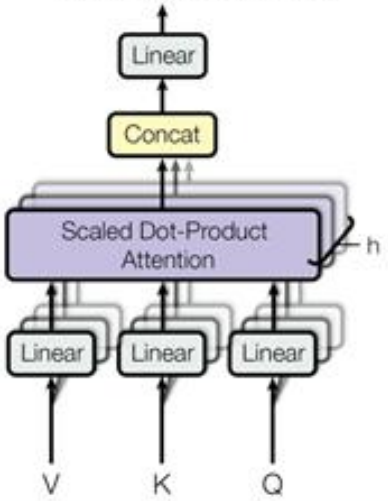
Multi-Head Attention

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

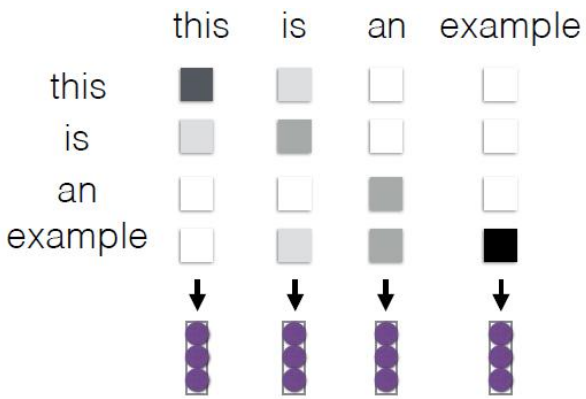
- Multi-head attention allows the model to jointly attend to **information from different representation subspaces** at different positions.

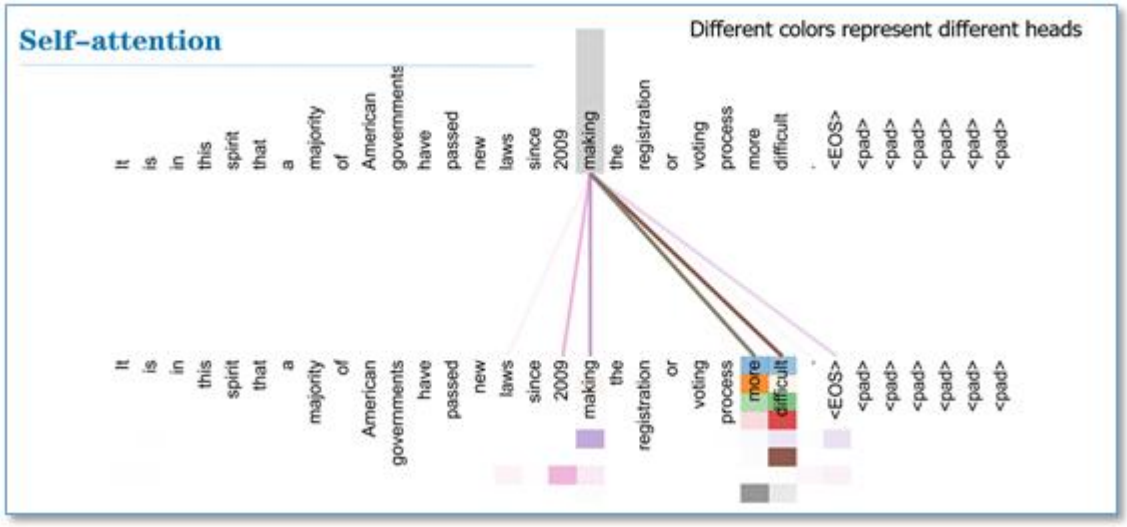
Multi-Head Attention



Intra-Attention / Self Attention

- Each element in the sentence attends to other elements → context sensitive encodings!





Attention is All You Need

Attention in the Model

- **Encoder-decoder attention**

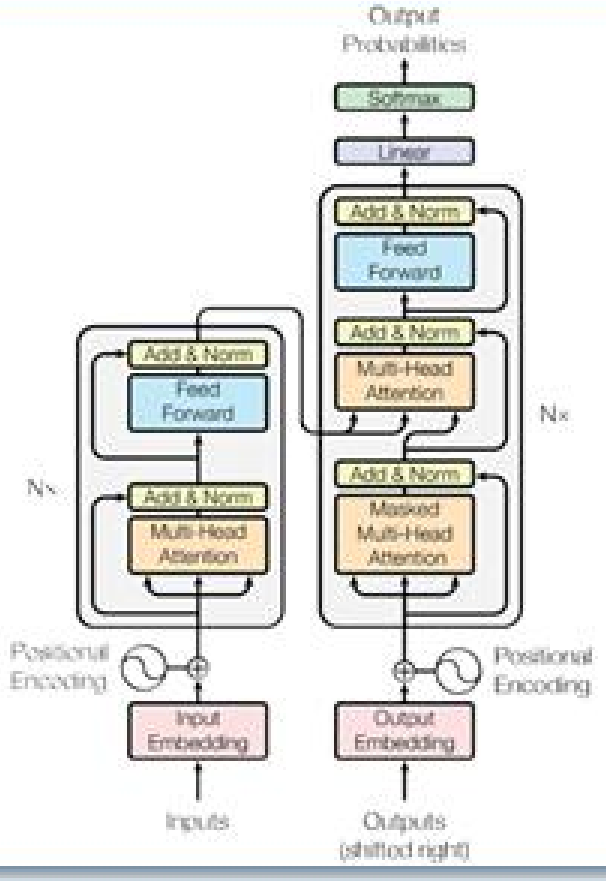
$$Y = \text{MultiHead}(V, K, Q) = \text{MultiHead}(X_e, X_e, X_d)$$

- **Self-attention layers in the encoder**

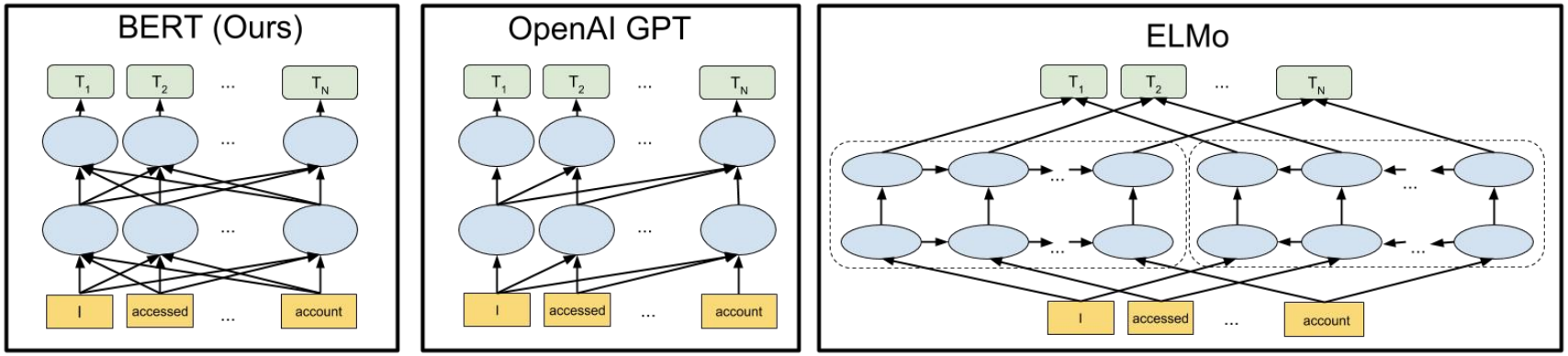
$$Y = \text{MultiHead}(V, K, Q) = \text{MultiHead}(X_e, X_e, X_e)$$

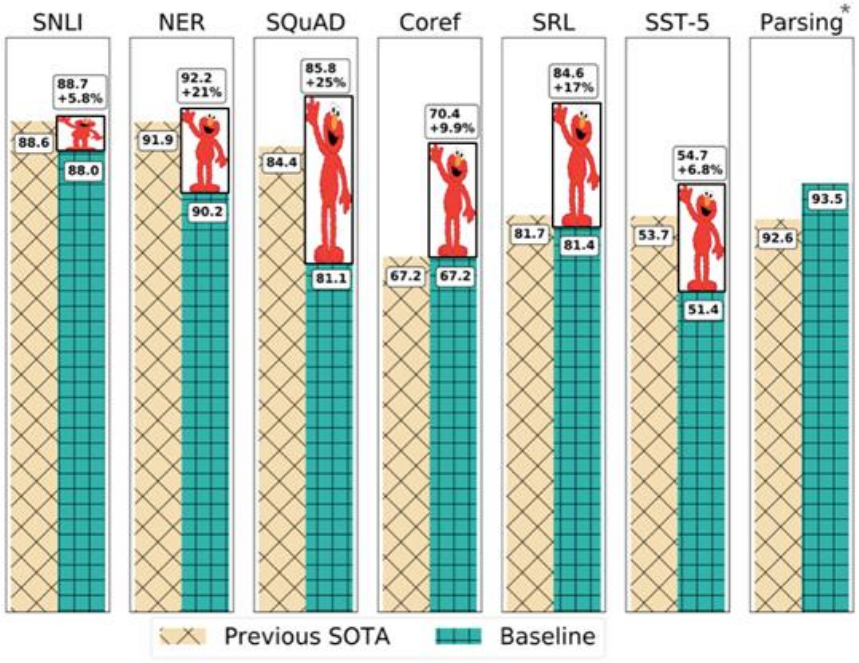
- **Self-attention layers in the decoder**

$$Y = \text{MultiHead}(V, K, Q) = \text{MultiHead}(X_d, X_d, X_d)$$



Language Models other than Word2Vec



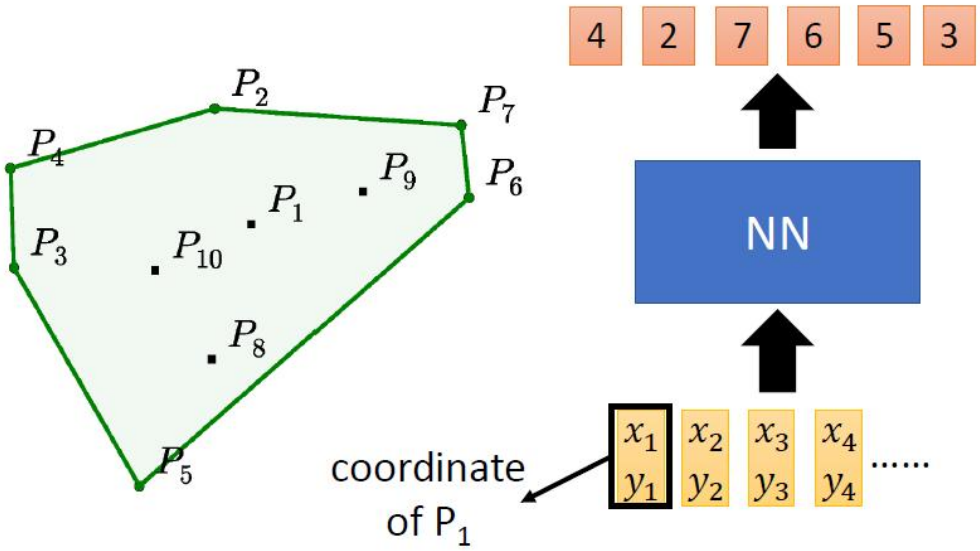


*Kitaev and Klein, ACL 2018 (see also Joshi et al., ACL 2018)

the ImageNet moment of NLP

Pointer Network

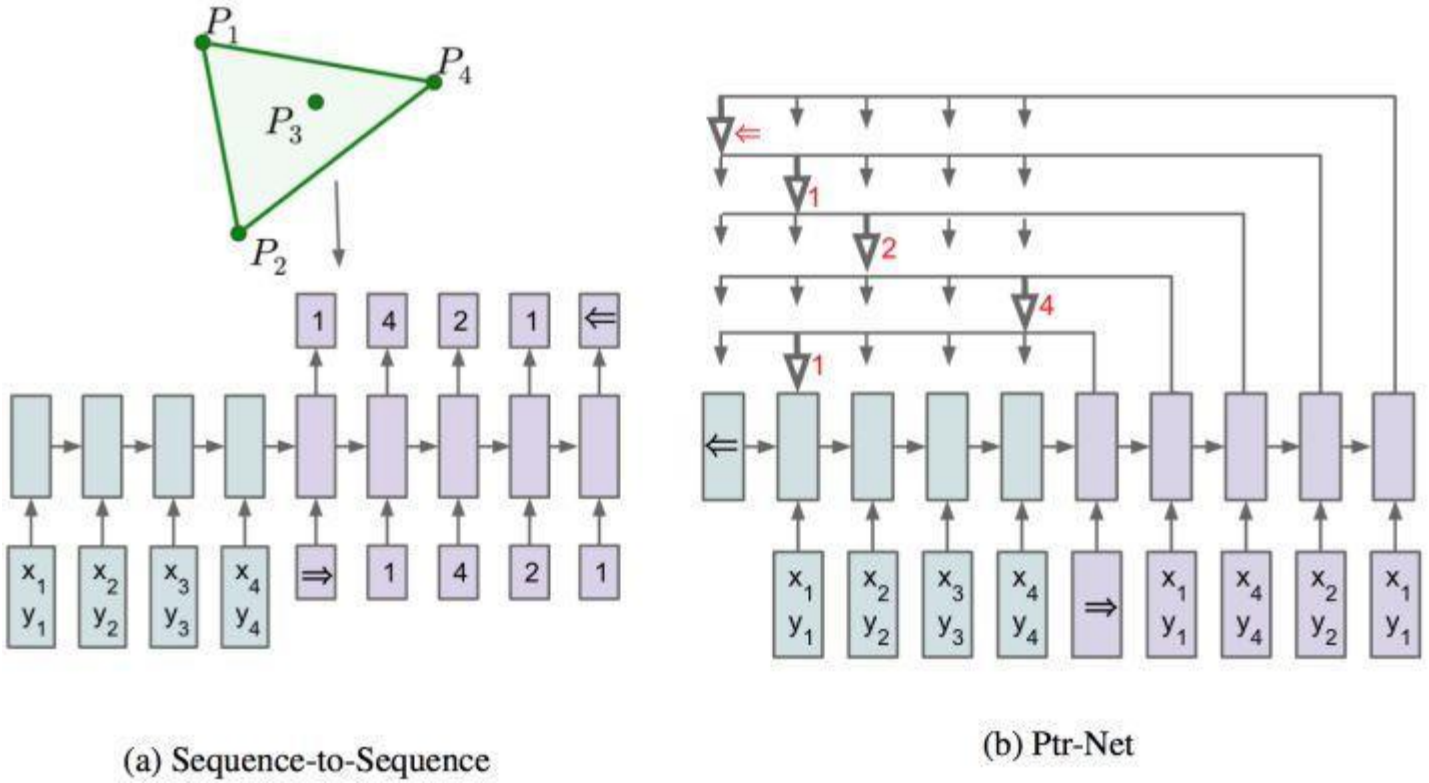
- Looking for a convex hull (Find a few points to wrap all the points)



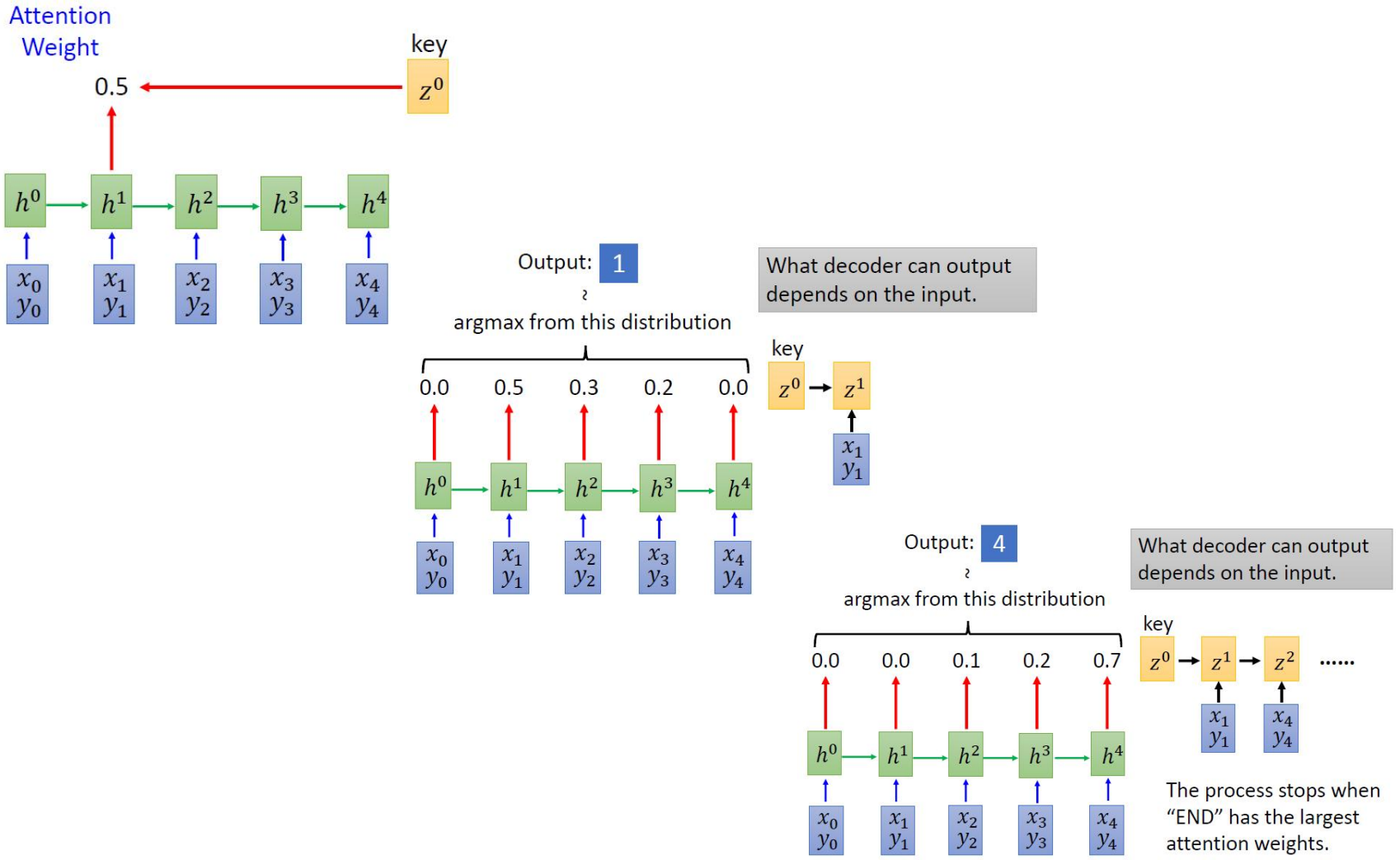
- A simple idea: use Seq2Seq model
  - Input: (x_i, y_i) Output: the convex hull
  - Problem: The output size of decoder is fixed. We need to find a way to dynamically output results. The output size should depend on the input.

Pointer Network

- The solution of the convex hull is the process of selecting points from the input sequence $\{P_i\}$.

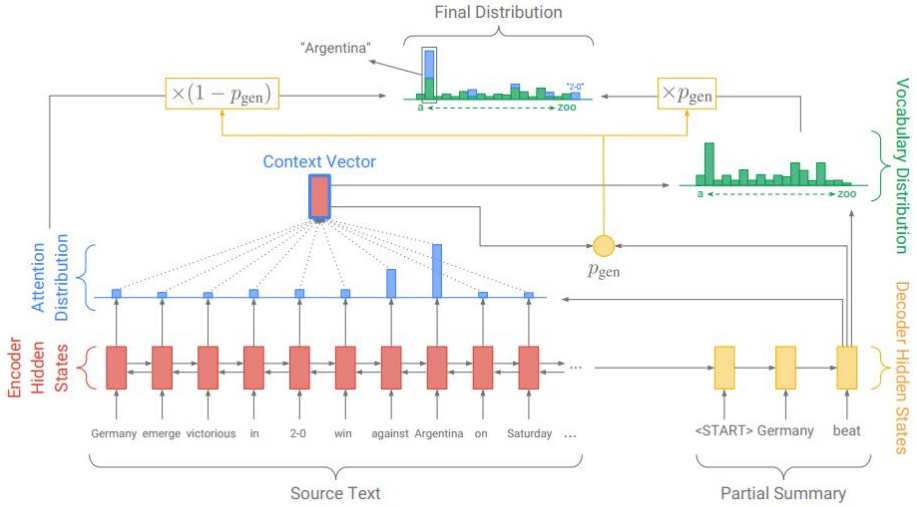


Pointer Network

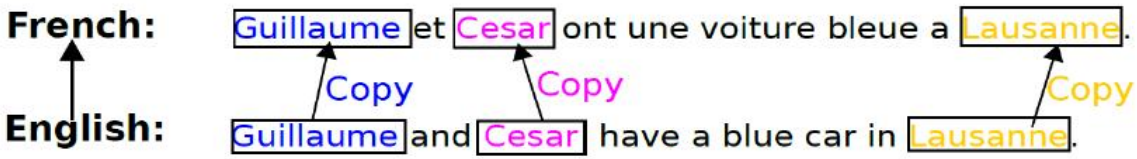


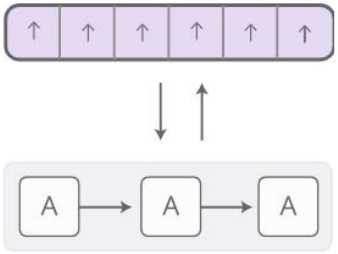
Pointer Network Applications

- Summarization

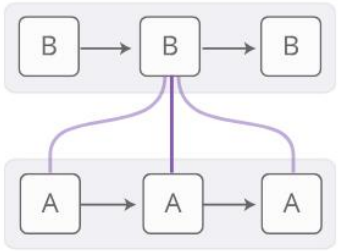


- Machine Translation

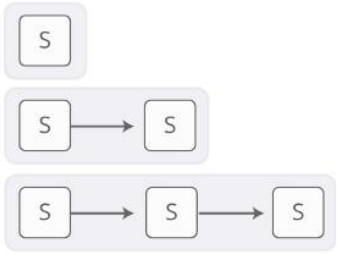




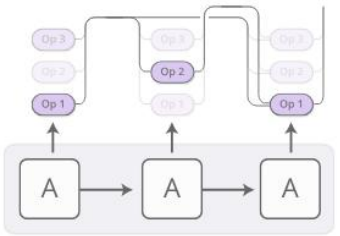
Neural Turing Machines
have external memory that they can read and write to.



Attentional Interfaces
allow RNNs to focus on parts of their input.



Adaptive Computation Time
allows for varying amounts of computation per step.



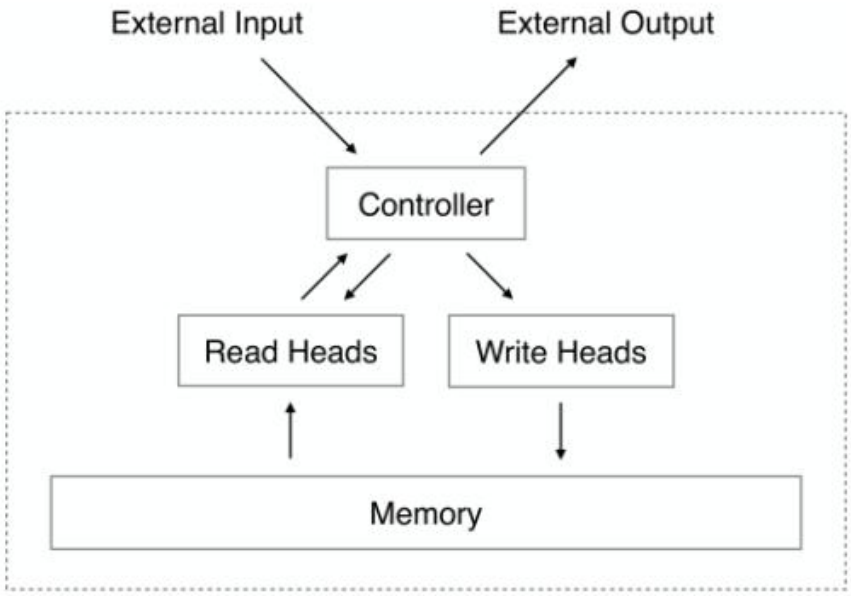
Neural Programmers
can call functions, building programs as they run.

Attention and Augmented RNN

https://distill.pub/2016/augmented-rnns/

NTM: Neural Turing Machines

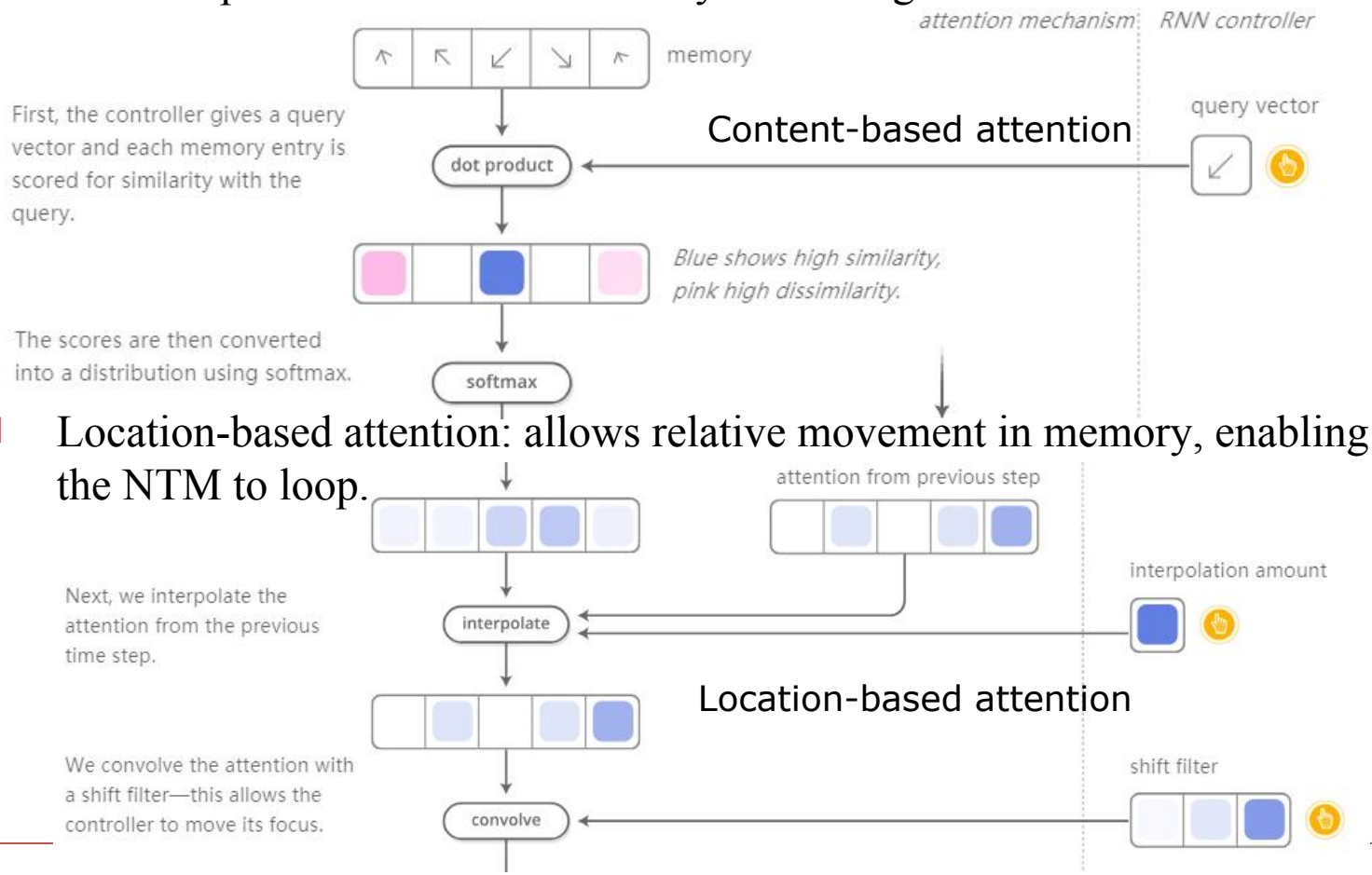
- RNN 是 Turing-complete 的，可以用来模拟任何函数，当然也可以模拟任何程序的功能。
- NTM extends the capabilities of neural networks by coupling neural networks to external memory resources.



- In particular, every component in the architecture is differentiated, which makes gradient descent training more straightforward.

NTM: Neural Turing Machines

- NTMs 如何寻址?
- **Content-based attention + Location-based attention**
  - Content-based attention: allows NTMs to search through their memory and focus on places that match what they're looking for



  - Location-based attention: allows relative movement in memory, enabling the NTM to loop.

DNC: Differentiable Neural Computer

MENU

nature
International journal of science

Article | Published: 12 October 2016

Hybrid computing using a neural network with dynamic external memory

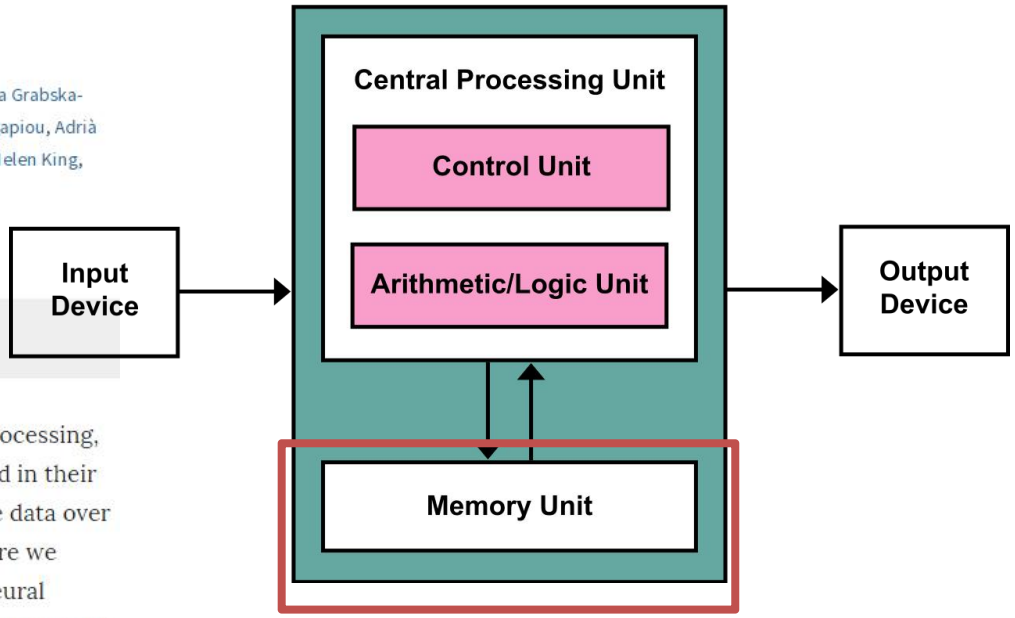
Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu & Demis Hassabis

Nature **538**, 471–476 (27 October 2016) | Download Citation

Abstract

Artificial neural networks are remarkably adept at sensory processing, sequence learning and reinforcement learning, but are limited in their ability to represent variables and data structures and to store data over long timescales, owing to the lack of an external memory. Here we introduce a machine learning model called a differentiable neural computer (DNC), which consists of a neural network that can read from and write to an external memory matrix, analogous to the random-access memory in a conventional computer. Like a conventional

Von Neumann architecture



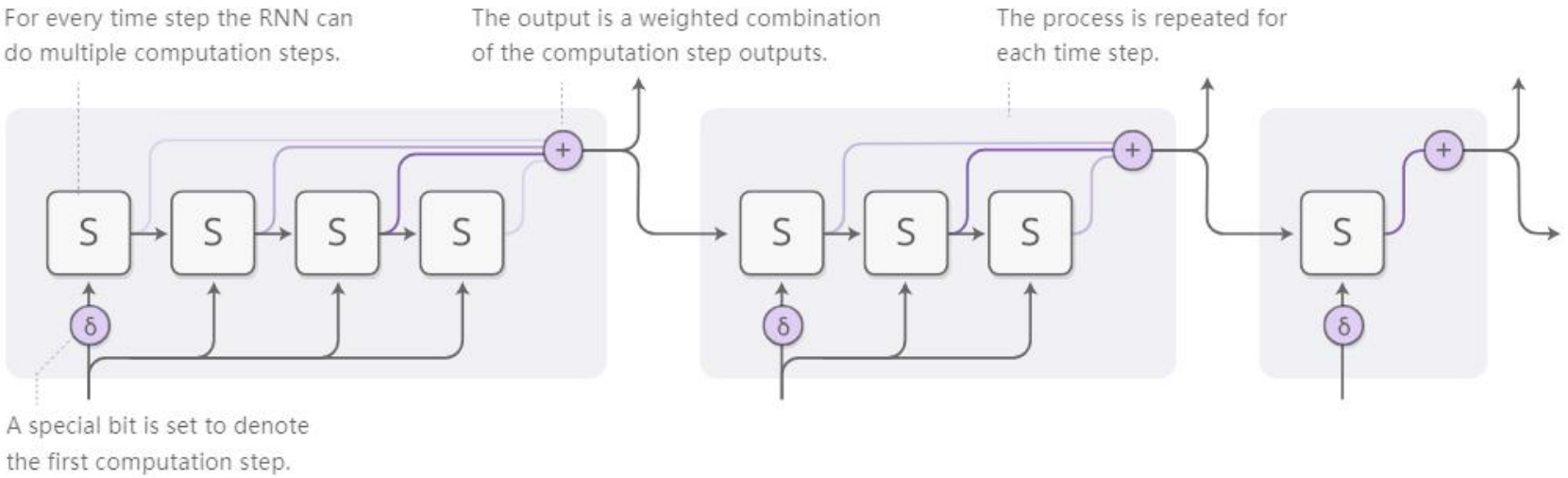
DNC: Differentiable Neural Computer

- DNC 改进了 NTM 的寻址机制

NTM 不能保障多个存储单元之间，不相互重叠，不相互干扰	dynamic memory allocation: allocate a free space
NTM 不能释放存储单元，如果处理很长的序列时，譬如处理一部超长的长篇小说，搞不好所有存储都会被占满，导致系统崩溃	dynamic memory allocation: free gates
NTM 中，一旦某个读写操作，远跳到其它存储区域，那么后续操作也跟着去其它区域。NTM 由于没有时序链接会想不起来原先的存储区域在哪里	temporal link matrix

ACT: Adaptive Computation Time

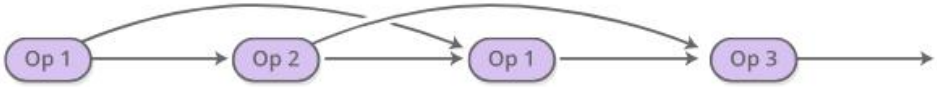
- RNN每次的计算量是固定的（一次执行一个Cell）。自适应计算(ACT) 允许RNN在每次执行多轮（一次执行一个Cell多次），可以理解为在RNN中嵌入RNN。



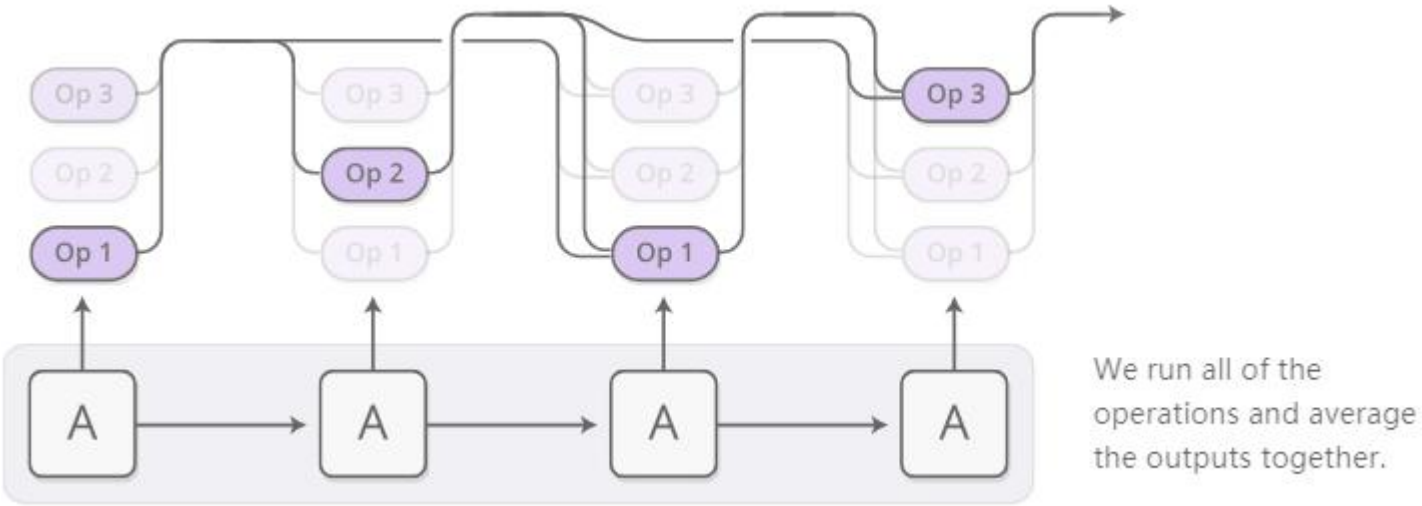
- 采用之前Attention相同的技巧，并不去决定轮数这个离散量，而是在轮数上分配注意力，最后的输出是多轮结果的加权和。
- 在训练ACT模型时，通常会引入“思考成本” (ponder cost)作为损失函数的一部分，对过多计算进行惩罚。

Neural Programmer

- The program is generated one operation at a time by a controller RNN.



- 将神经网络和常规编程结合
- 如：生成SQL代码查询
- 基本想法是运行所有的操作，然后用Attention来分配结果的权重



Any Questions?