Lecture 4: Deep Learning Foundation

Xu Ruifeng

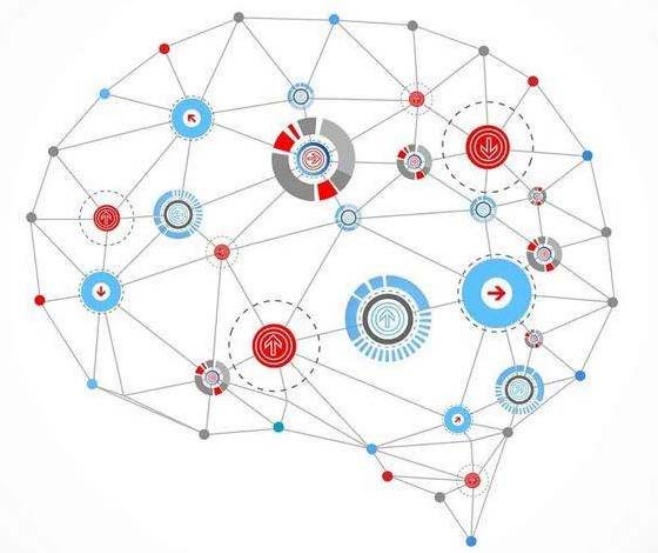
Harbin Institute of Technology, Shenzhen

Last Time

- What is machine learning?
- Linear Classifier
  - Logistic Regression
  - Support Vector Machine
- Structured Learning
  - Hidden Markov Model
  - Condition Random Field

Today's class

- From ML to DL
- Neuron and ANN
- DL models
  - RNN 循环神经网络
  - CNN 卷积神经网络
  - Attention 注意力机制
  - Encoder-Decoder 编码器-解码器网络



From ML to DL

- Let's recap machine learning.

- A **machine learning algorithm**:
  - **input**: Training Data
  - "learns" from the training data.
  - **output**: A "prediction function" that produces output y given input x.

From ML to DL

- Let's recap machine learning.

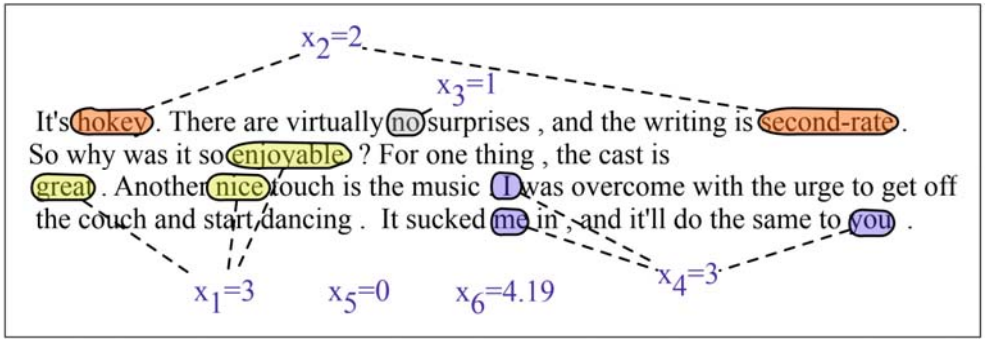
- e.g. Logistic Regression

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x} + b)}$$
$$p(y=0|\mathbf{x}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

From ML to DL

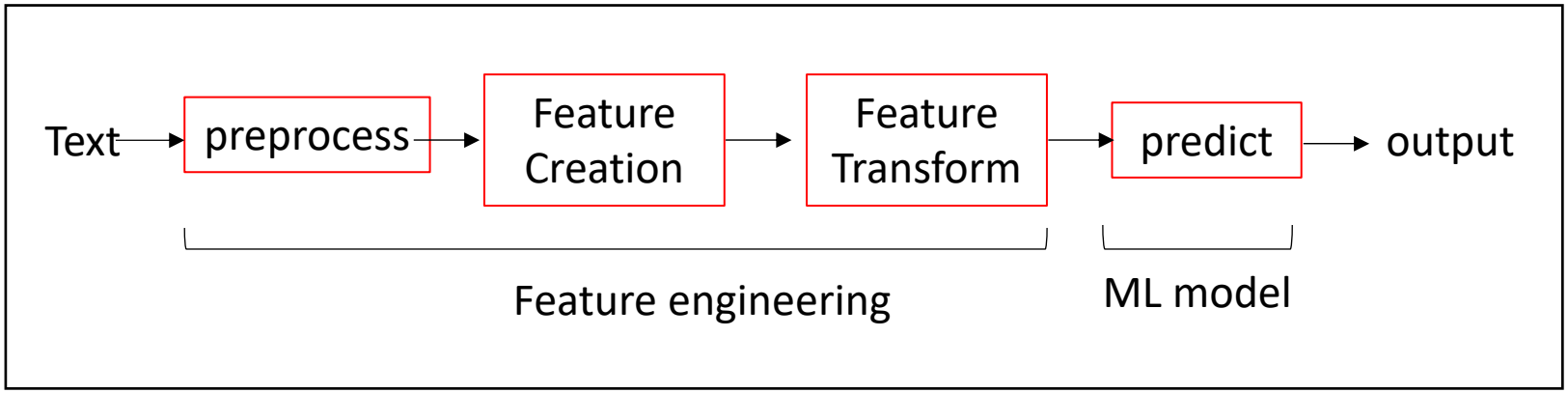
- e.g. Logistic Regression

Var	Definition
x_1	count(positive lexicon) $\in$ doc)
x_2	count(negative lexicon) $\in$ doc)
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)



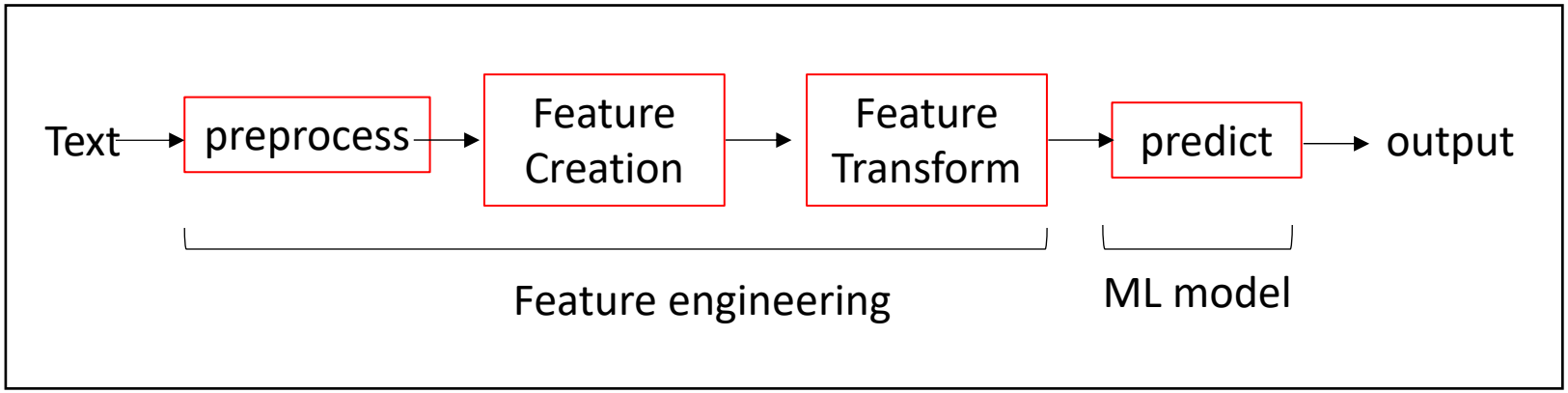
$$\mathbf{x} = [3, 2, 1, 3, 0, 4.19]^\top$$

From ML to DL



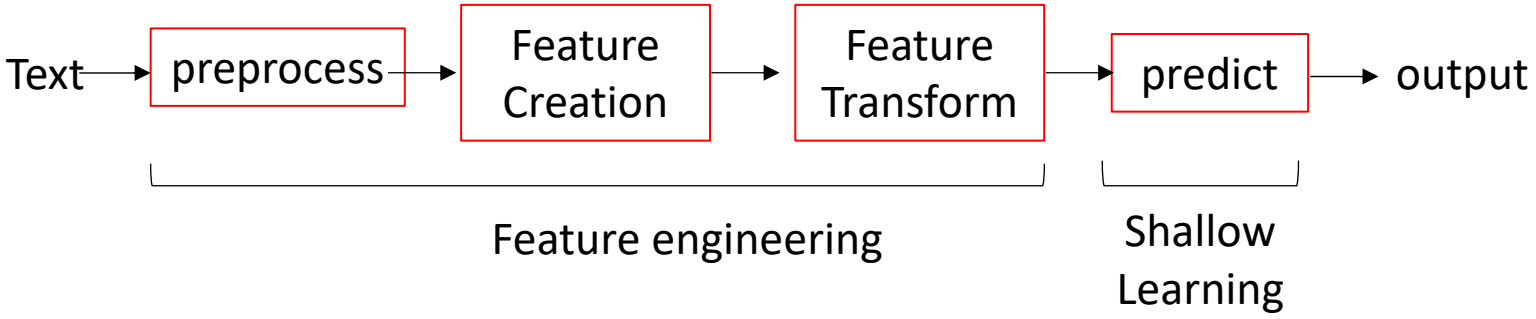
- **Preprocessing:** remove noisy text, remove stop words
- **Feature Creation:** create features from original data
- **Feature Transform:** feature processing, i.e. dimensionality reduction
  - feature extraction: creat a subset of new features by combinations of the existing features. i.e. PCA, LDA
  - feature selection: choose a subset of all the features(the ones more informative). i.e. mutual information, TF-IDF

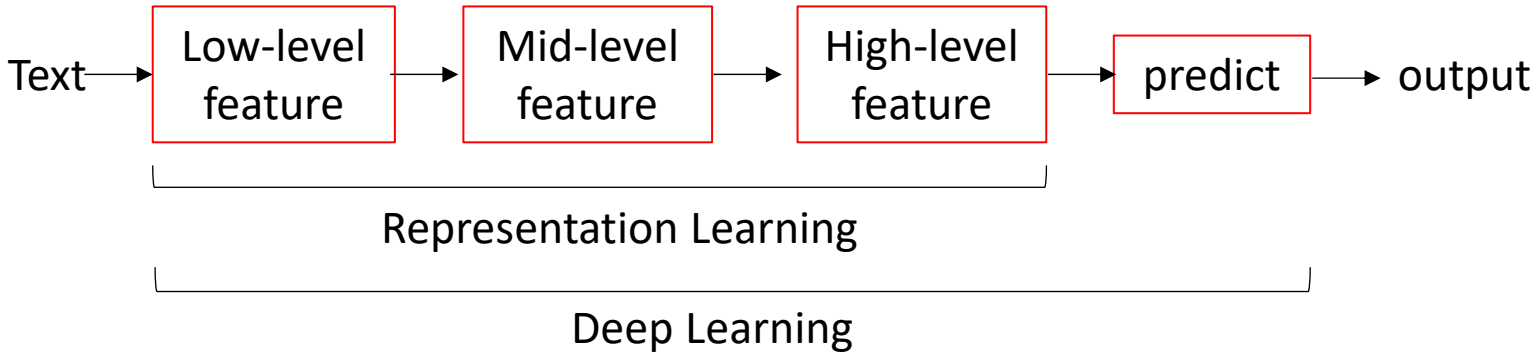
From ML to DL



- **The performance of ML model tends to be similar.**
- **Feature Engineering is more important!**

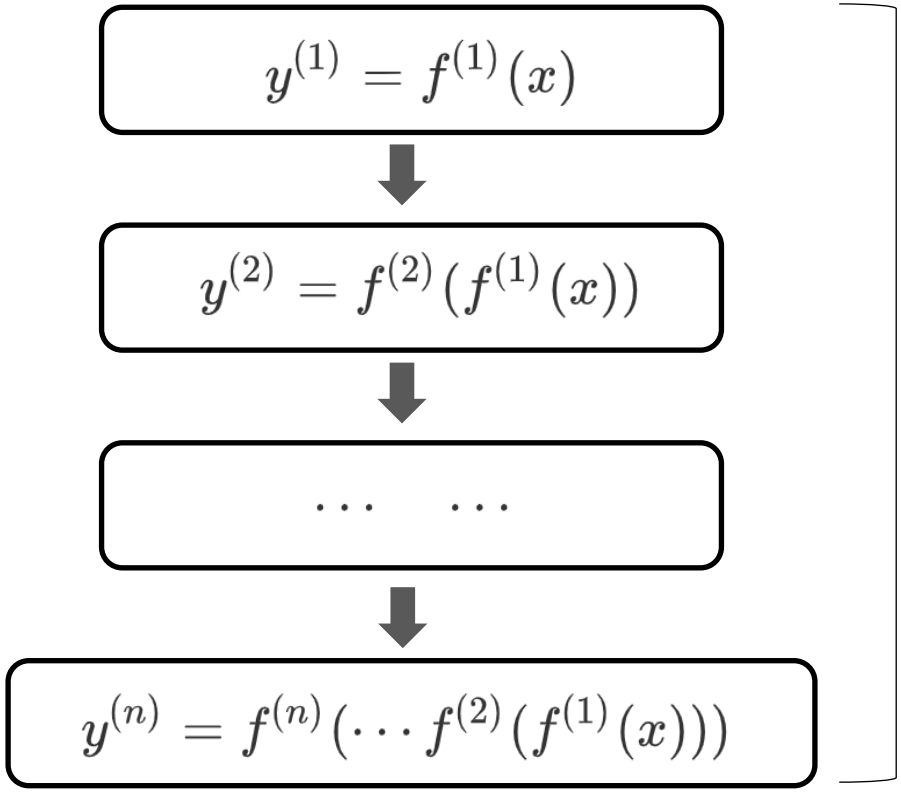
From ML to DL


Text
preprocess
Feature Creation
Feature Transform
predict
output
Feature engineering
Shallow Learning


Text
Low-level feature
Mid-level feature
High-level feature
predict
output
Representation Learning
Deep Learning

From ML to DL

Deep Learning



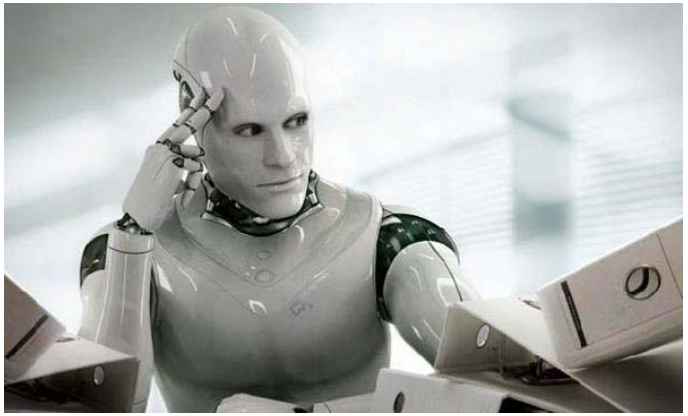
- stronger expression ability
  - Bengio and LeCun 2017
- layer by layer feature learning
  - Deeper Model -> Better non-linear expression capability

$$f(x) = \sigma(\mathbf{w}^\top \mathbf{x})$$

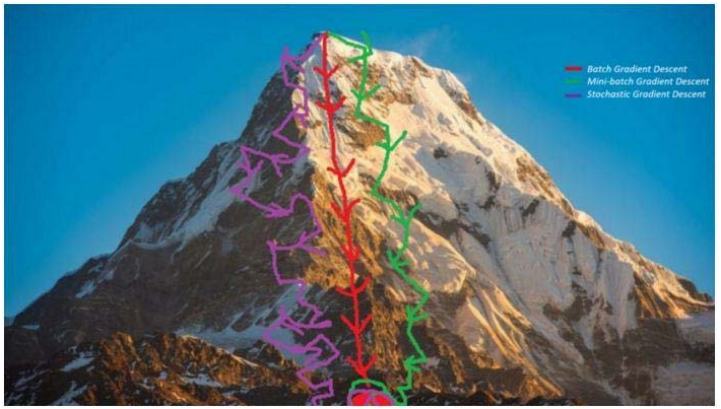
From ML to DL

- Three components
  - **Representation:** neural network.
  - **Evaluation:** cross-entropy loss, etc.
  - **Optimization:** stochastic gradient descent.

What others think DL is…

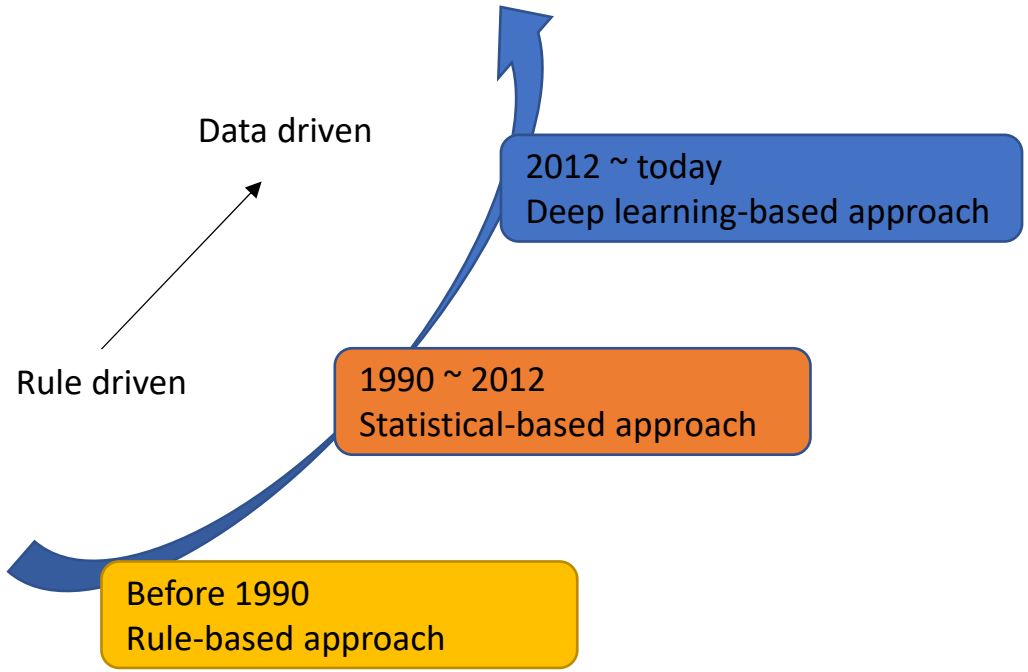


What it actually is …



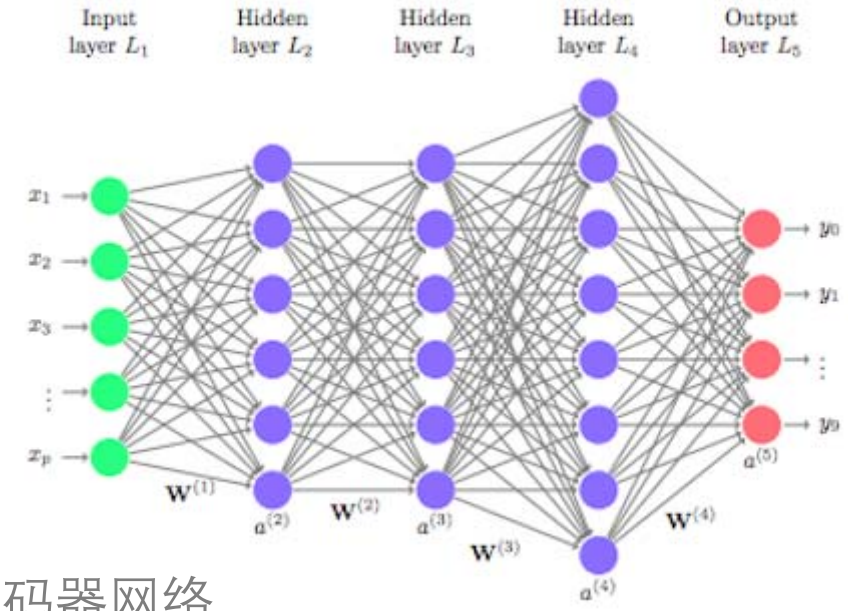
From ML to DL

- Why DL didn't work before?
  - large amounts of data
  - cheaper and faster computers
  - advanced ML techniques



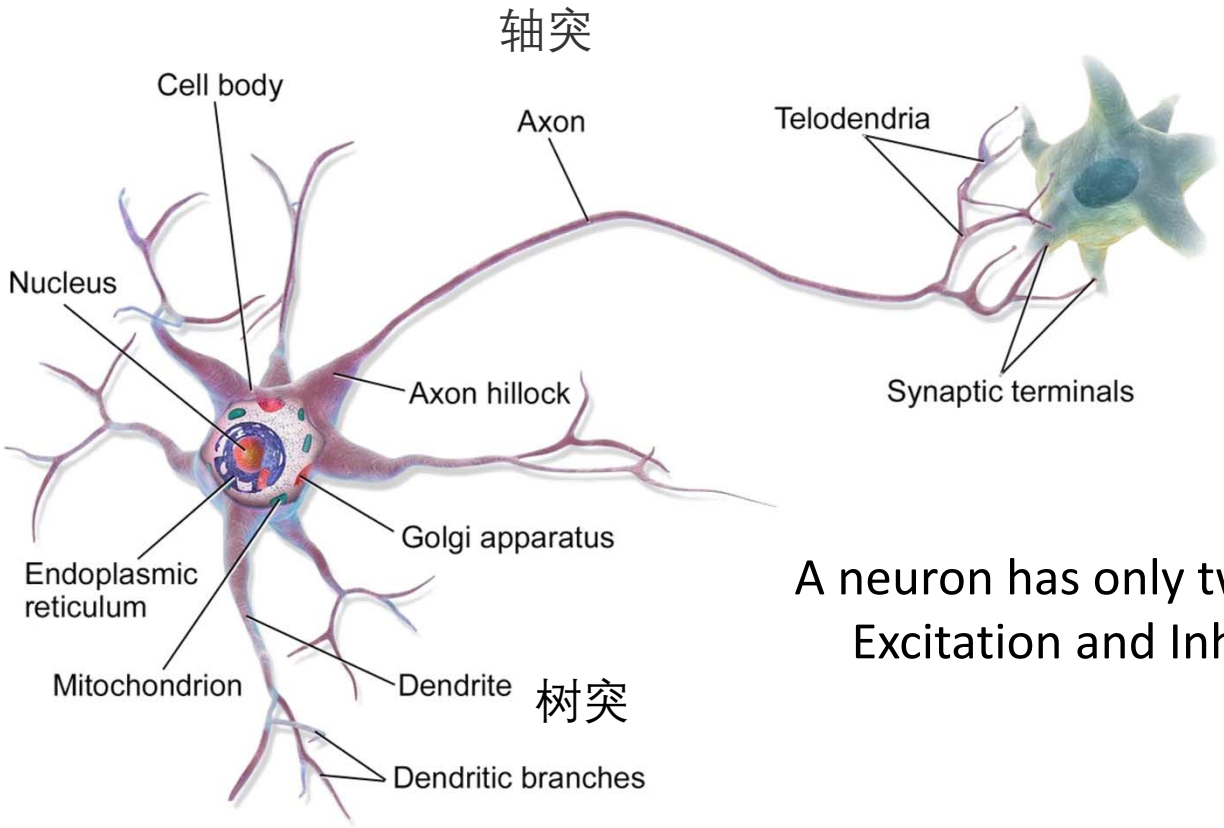
Today's class

- From ML to DL
- Neuron and ANN
- DL models
  - RNN 循环神经网络
  - CNN 卷积神经网络
  - Attention 注意力机制
  - Encoder-Decoder 编码器-解码器网络



Biological Neuron

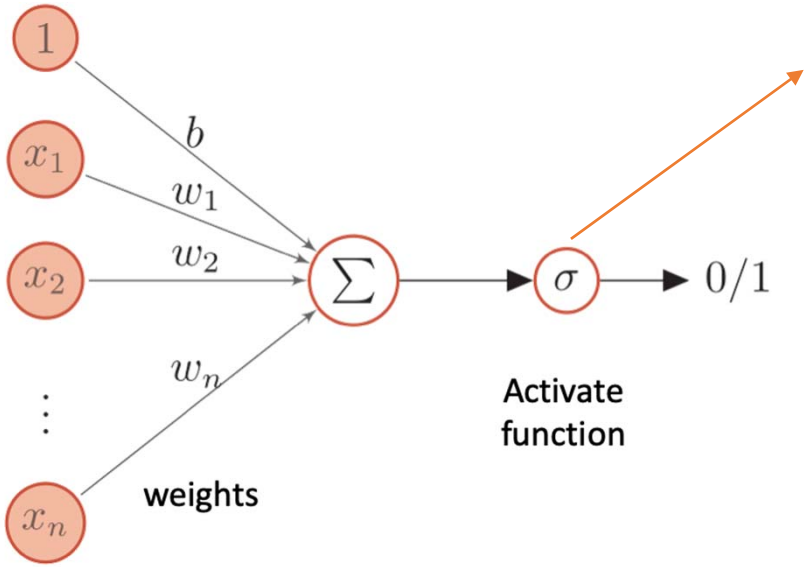
- 生物神经元

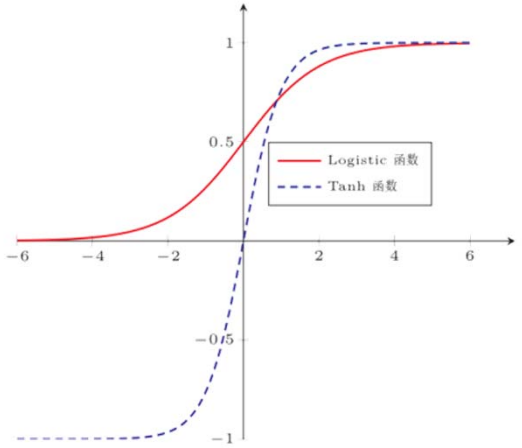


A neuron has only two states:
Excitation and Inhibition

Artificial Neuron

- 人工神经元





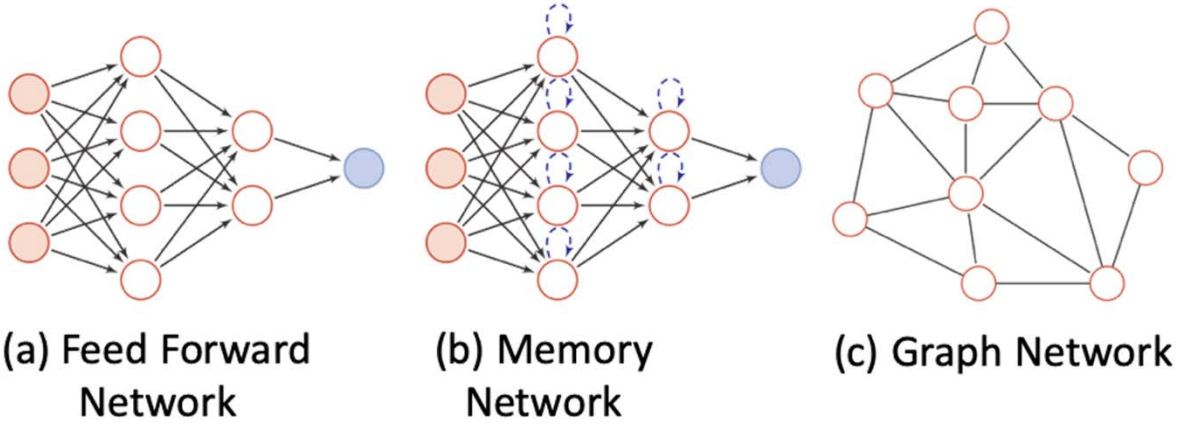
$$\sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x} + b)}$$

Artificial Neural Network

- An ANN is based on a collection of directed connected neurons.
- Three problems:
  - **neuron activations rules:** mapping from input to output, usually non-linear function
  - **network topology:** Connection between different neurons
  - **learning algorithm:** Learning appropriate parameters by training

Artificial Neural Network

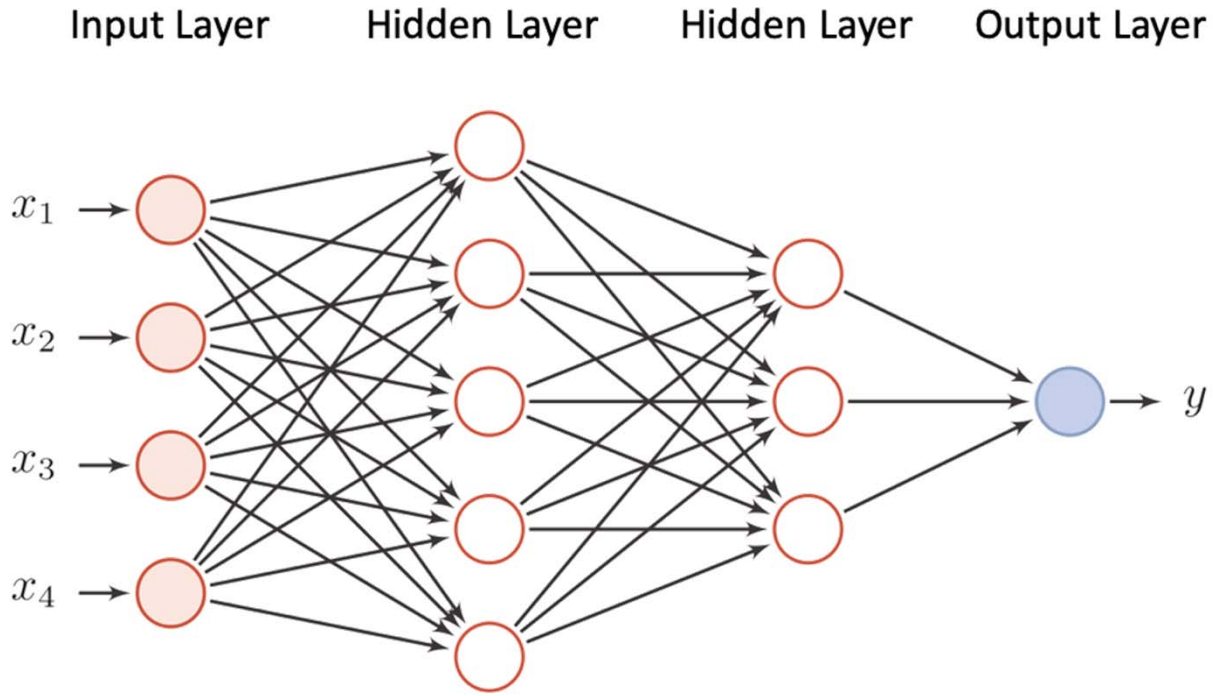
- Parallel distributed structure
  - three basic types of neural network



  - but actually, most networks are composite network that consists of different type of networks.

Feed Forward Neural Network

- A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle.



Universal Approximation Theorem

- A feed forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbf{R}^n$, under mild assumptions on the activation function.
- The theorem thus states that simple neural networks can represent a wide variety of interesting functions when given appropriate parameters.

A two-layer neural network can simulate any function

Popular Deep Learning Framework

- Easy and fast prototyping
- Automatic gradient computation
- Support CPUs and GPUs

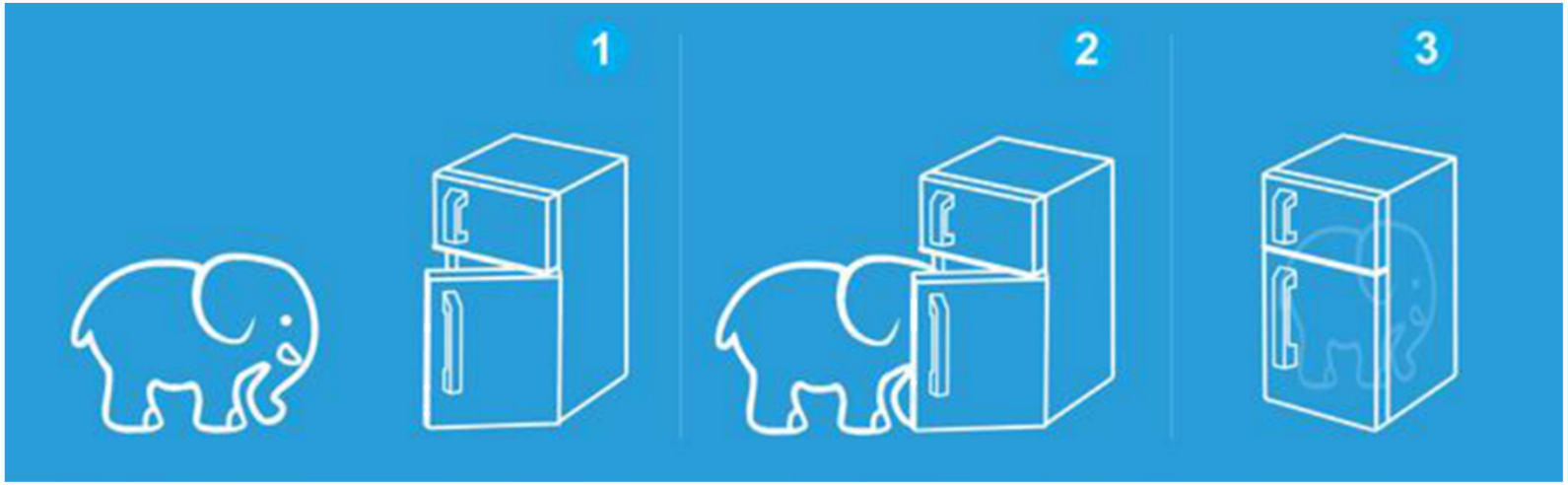
Three Steps of Deep Learning

- Deep Learning is so simple…

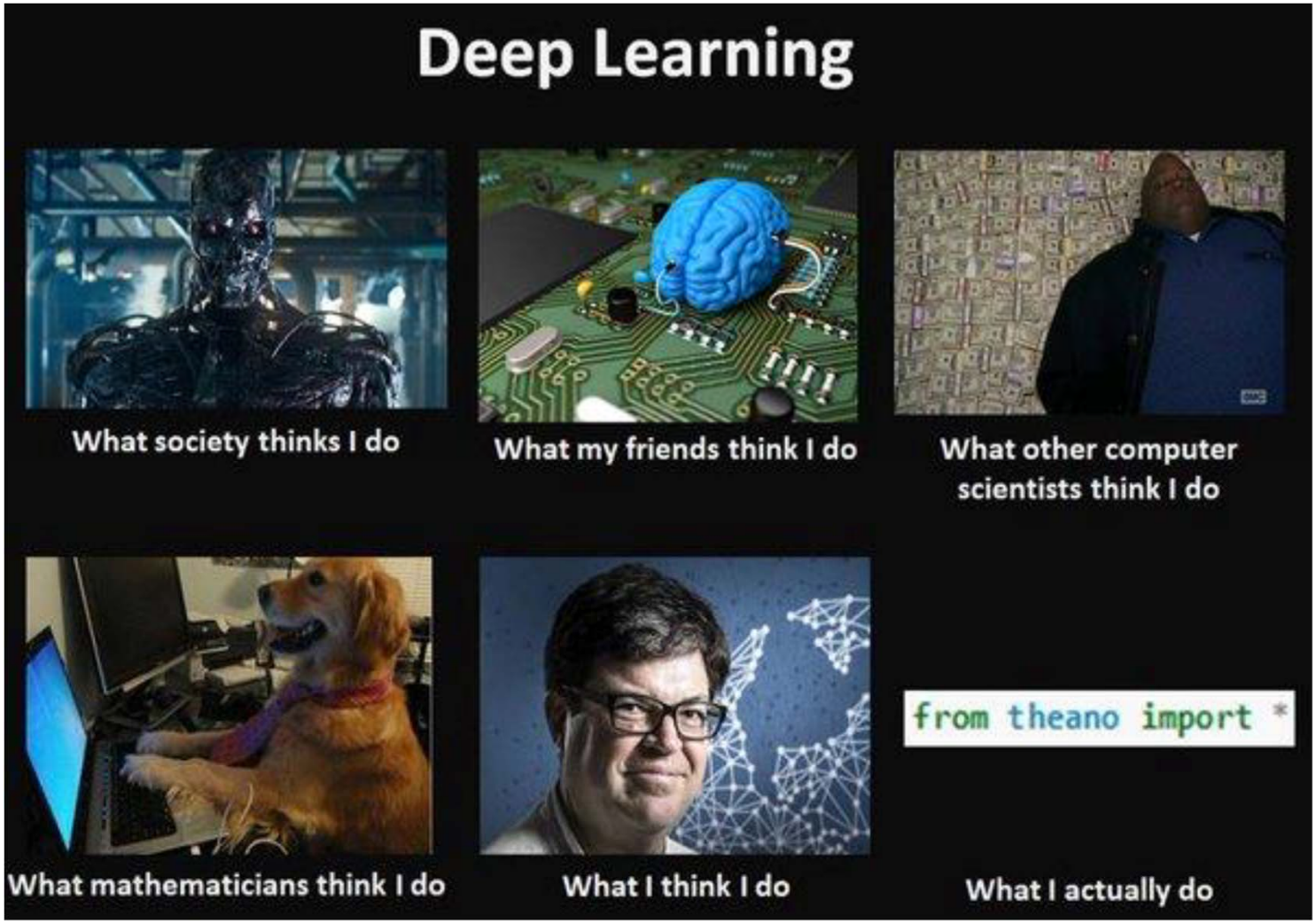
  ① design a network

  ② design a loss function

  ③ optimize the parameters

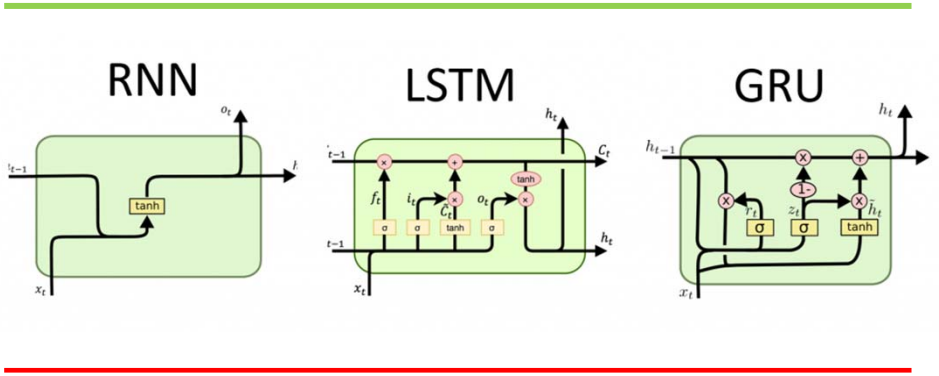


Deep Learning is easy :-)



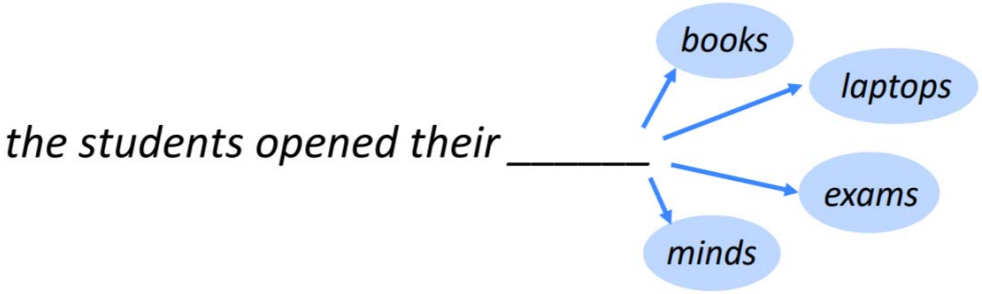
Today's class

- From ML to DL
- Neuron and ANN
- DL models
  - RNN 循环神经网络
  - CNN 卷积神经网络
  - Attention 注意力机制
  - Encoder-Decoder 编码器-解码器网络



Language Modeling

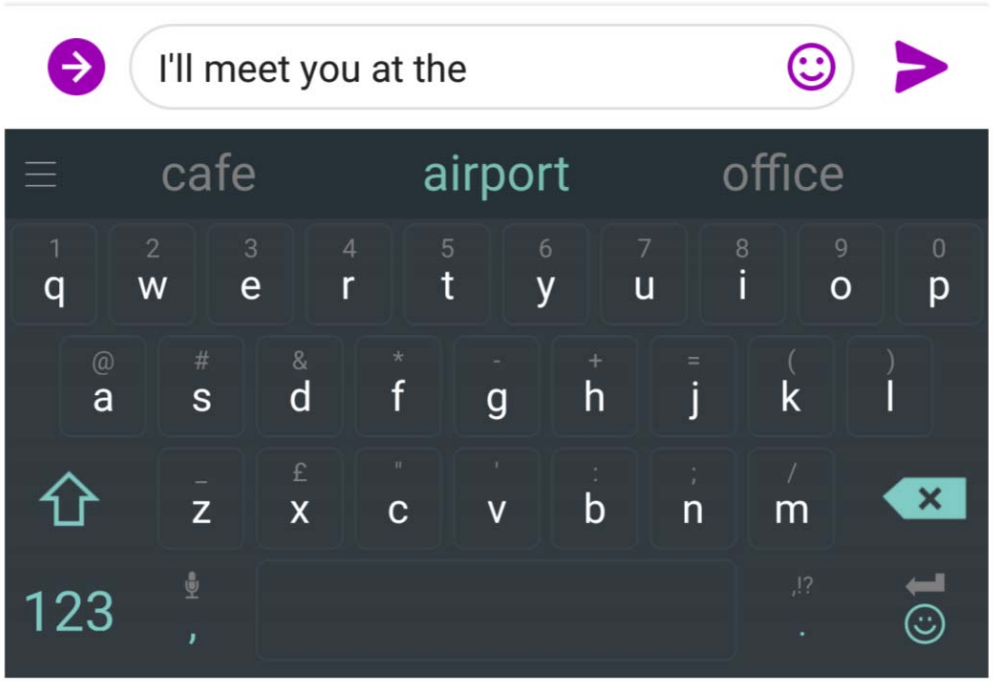
- Task: Language Modeling
  - to predict what word comes next.



  - Formally, given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability distribution of the next words $x^{(t+1)}$.

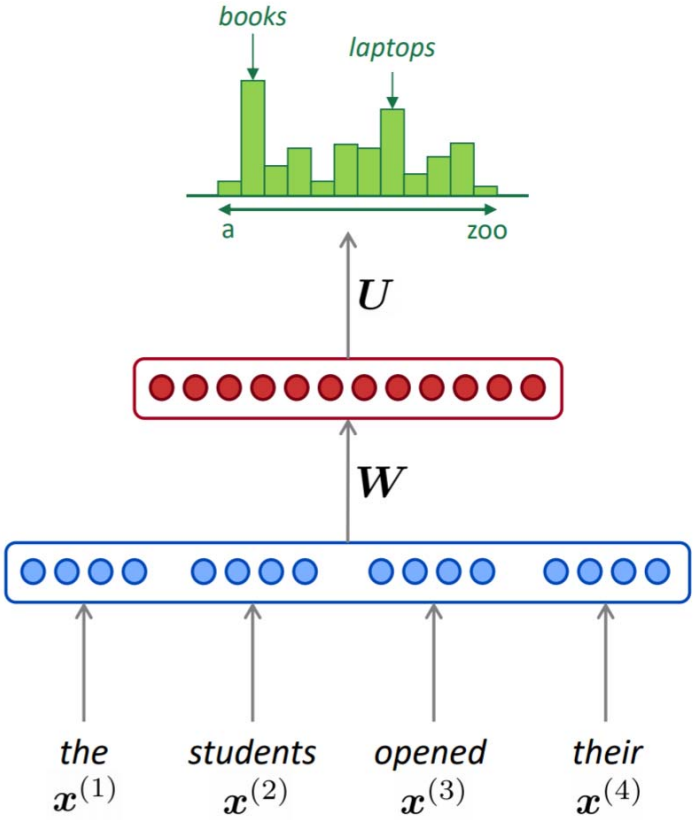
$$P(x^{(t+1)}|x^{(t)}, \cdots, x^{(1)})$$

Language Modeling

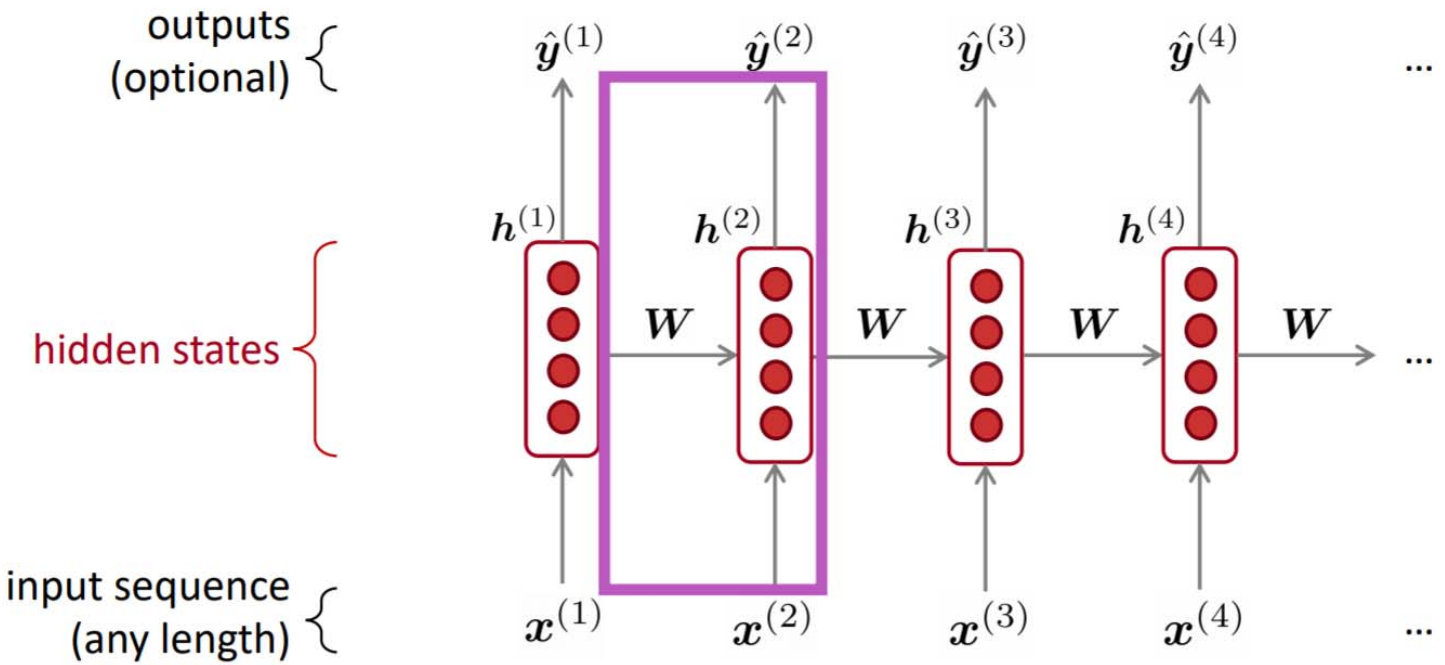


Language Modeling

- A fixed-window neural Language Model
  - fixed window, e.g. $n_{window} = 4$
  - parameters: W, U
- Weakness
  - fixed window is too small
  - enlarging window enlarges W
  - the interaction between $x^{(1)}$ and $x^{(2)}$ is not modeled.



Recurrent Neural Networks

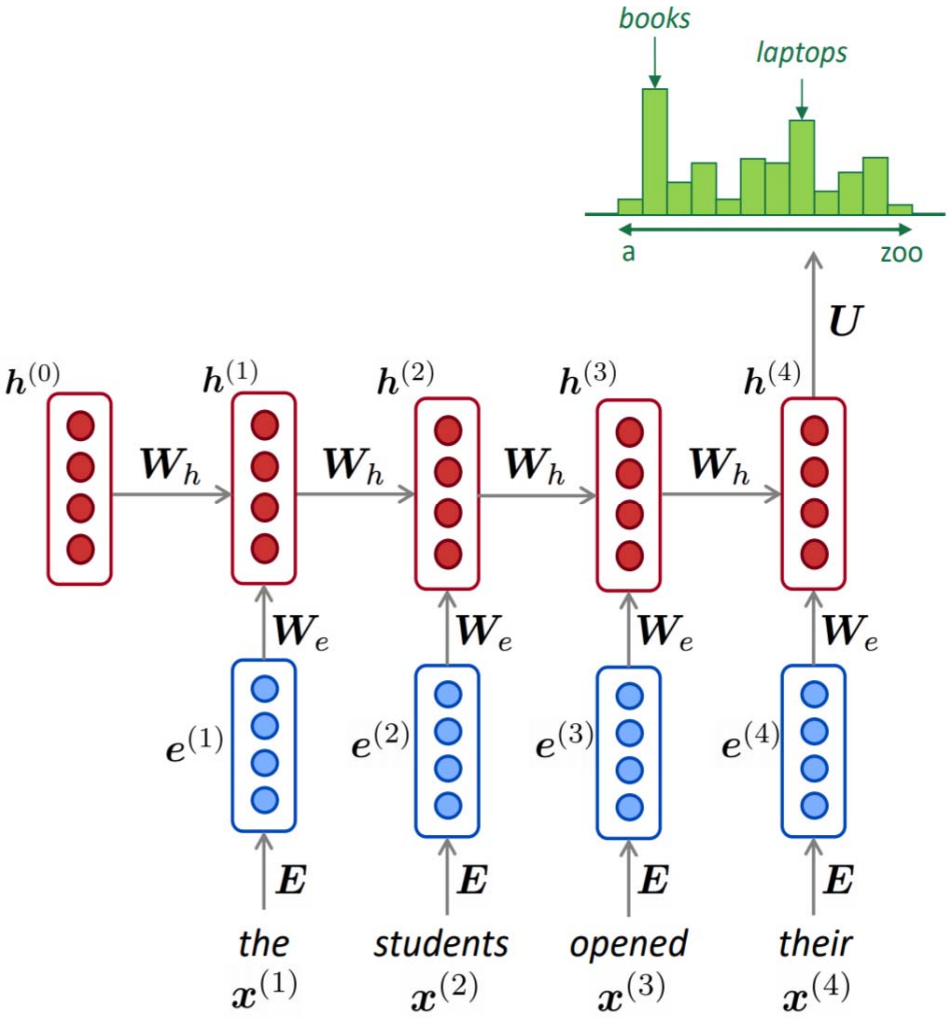


Core idea: Apply the same weights W repeatedly.

Recurrent Neural Networks

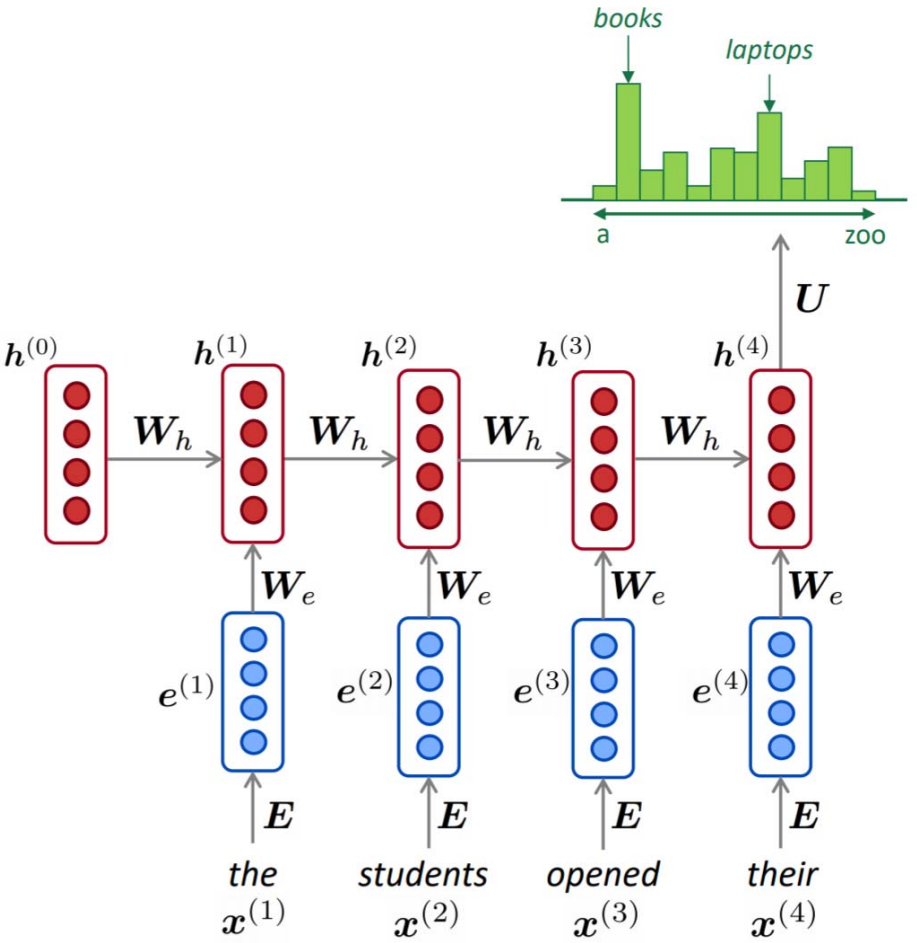
hidden states

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$



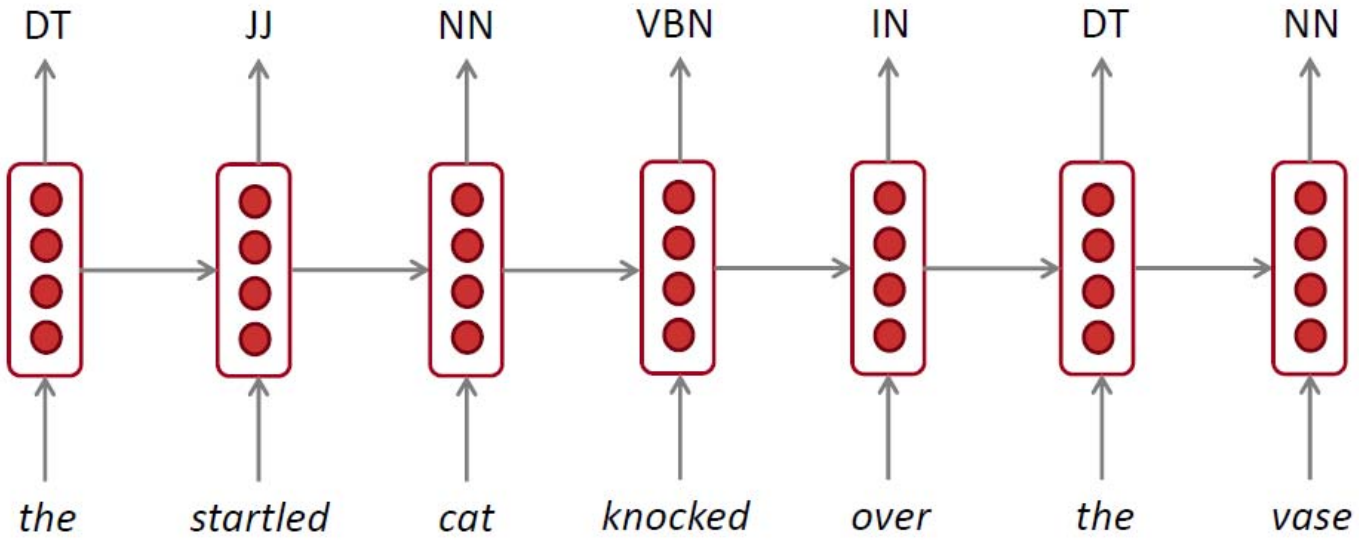
Recurrent Neural Networks

- Advantages:
  - can process any length input
  - computation for step t can (in theory) use information from many steps back
  - model size doesn't increase for longer input
  - same weights applied on every timestep, so there is symmetry in how inputs



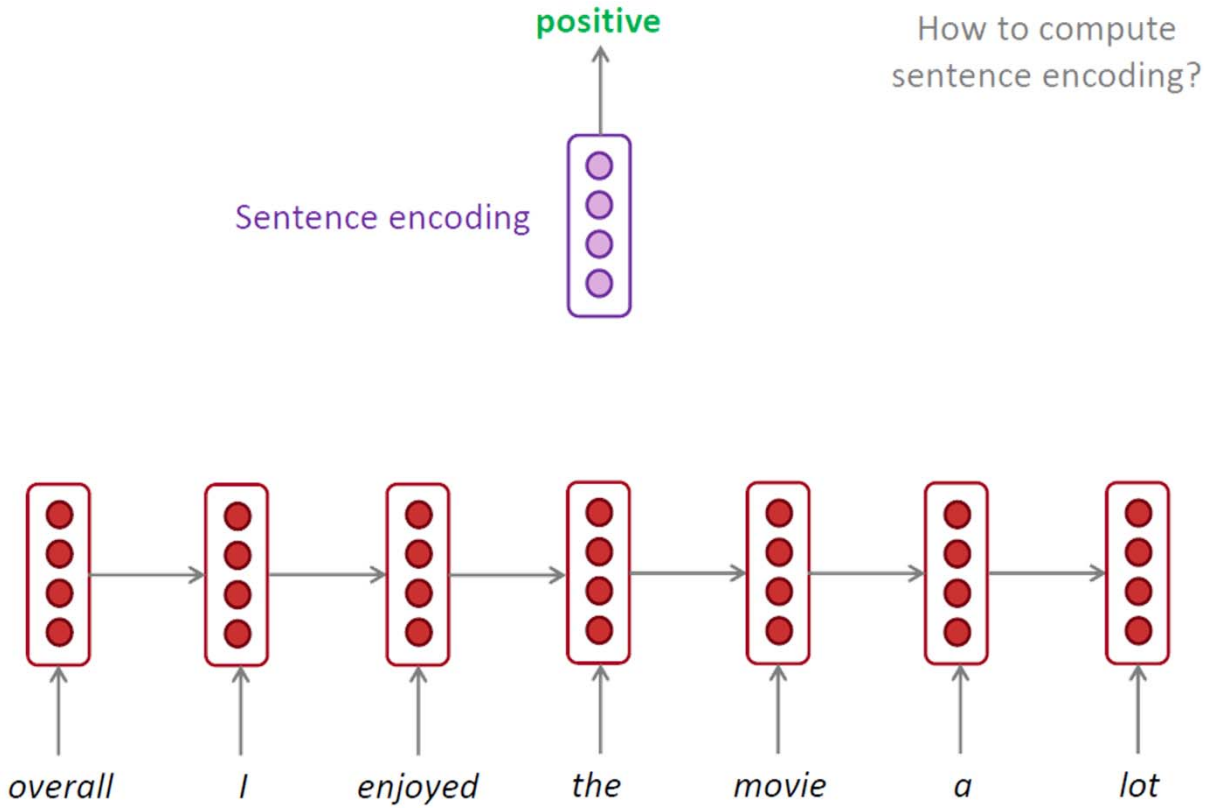
RNN can be used for tagging

- e.g. Part-of-speech tagging



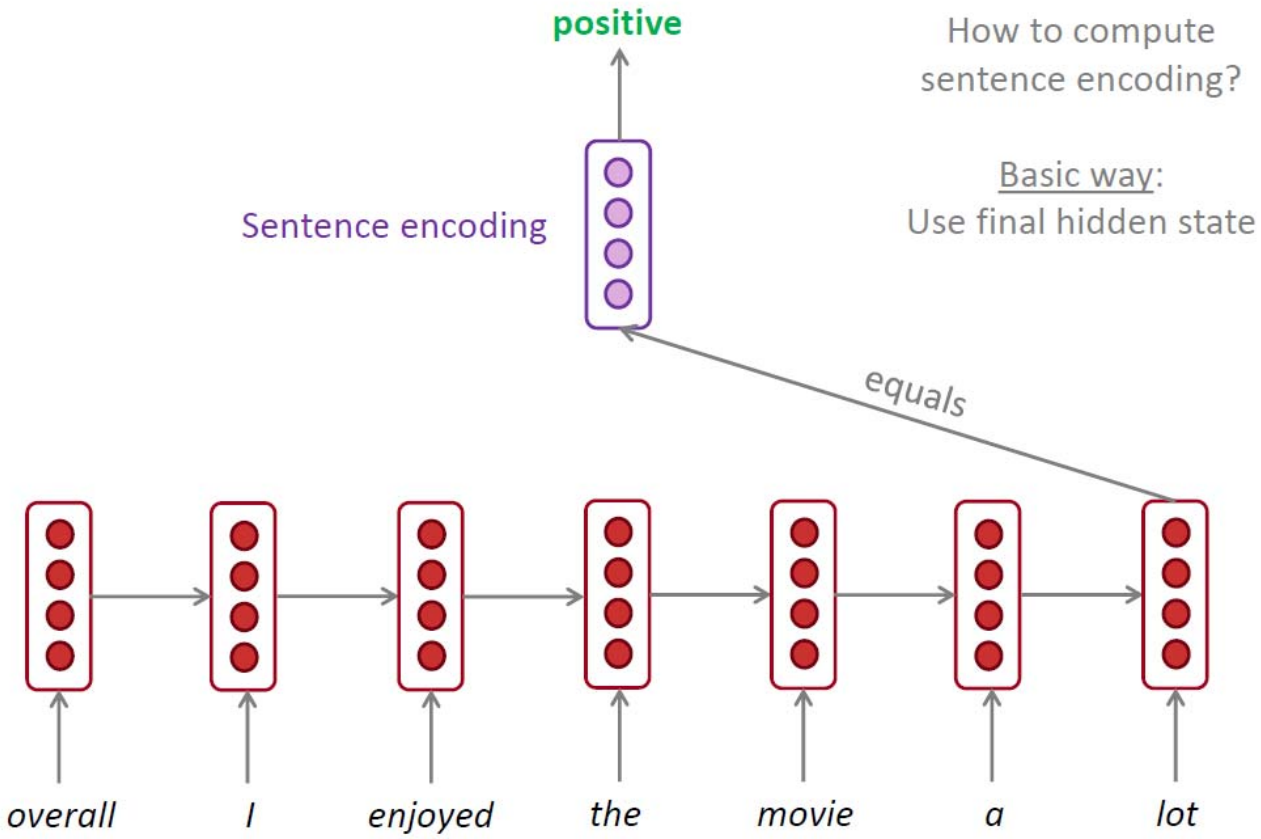
RNN can be used for sentence classification

- e.g. sentiment classification



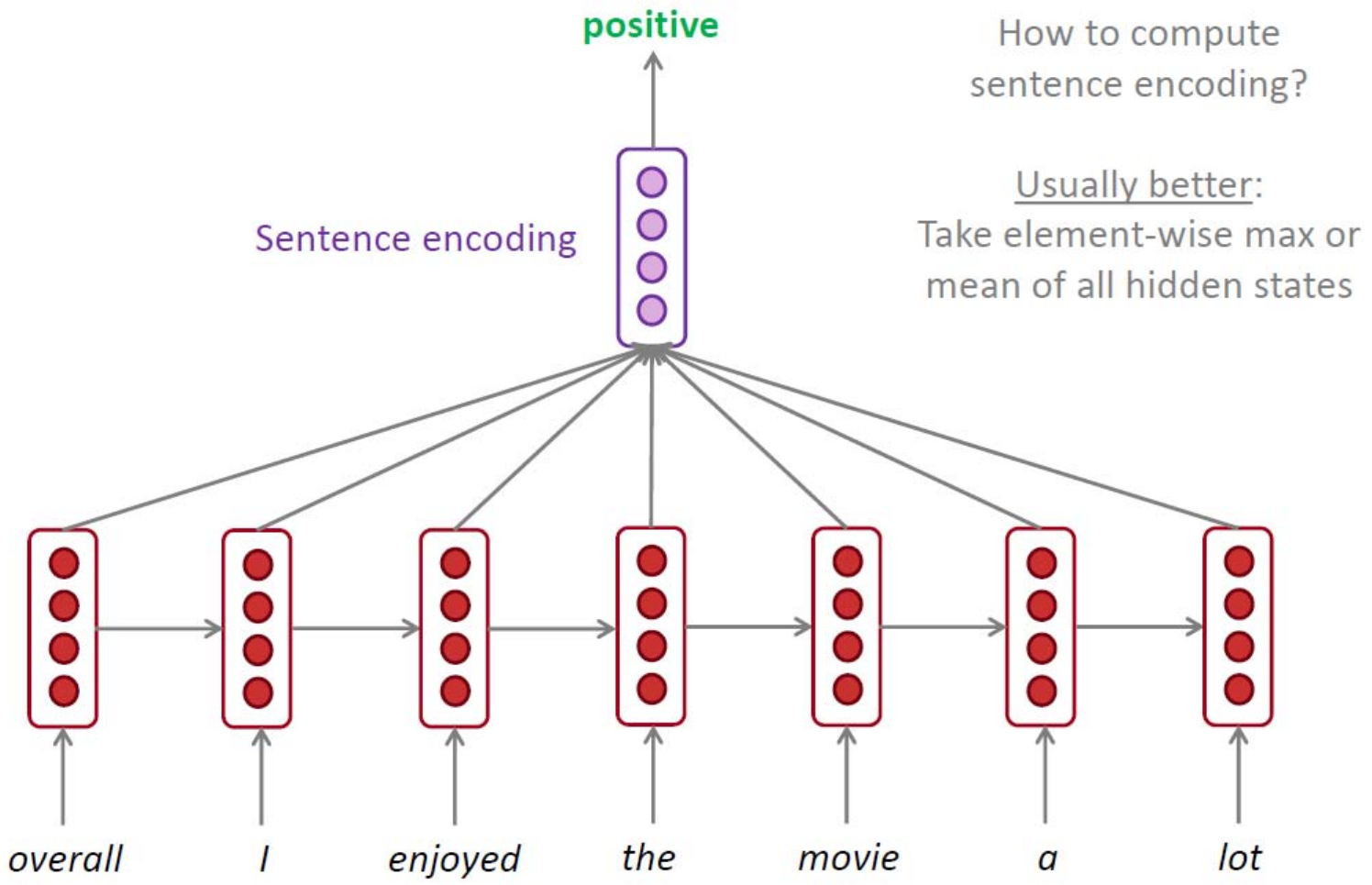
RNN can be used for sentence classification

- e.g. sentiment classification



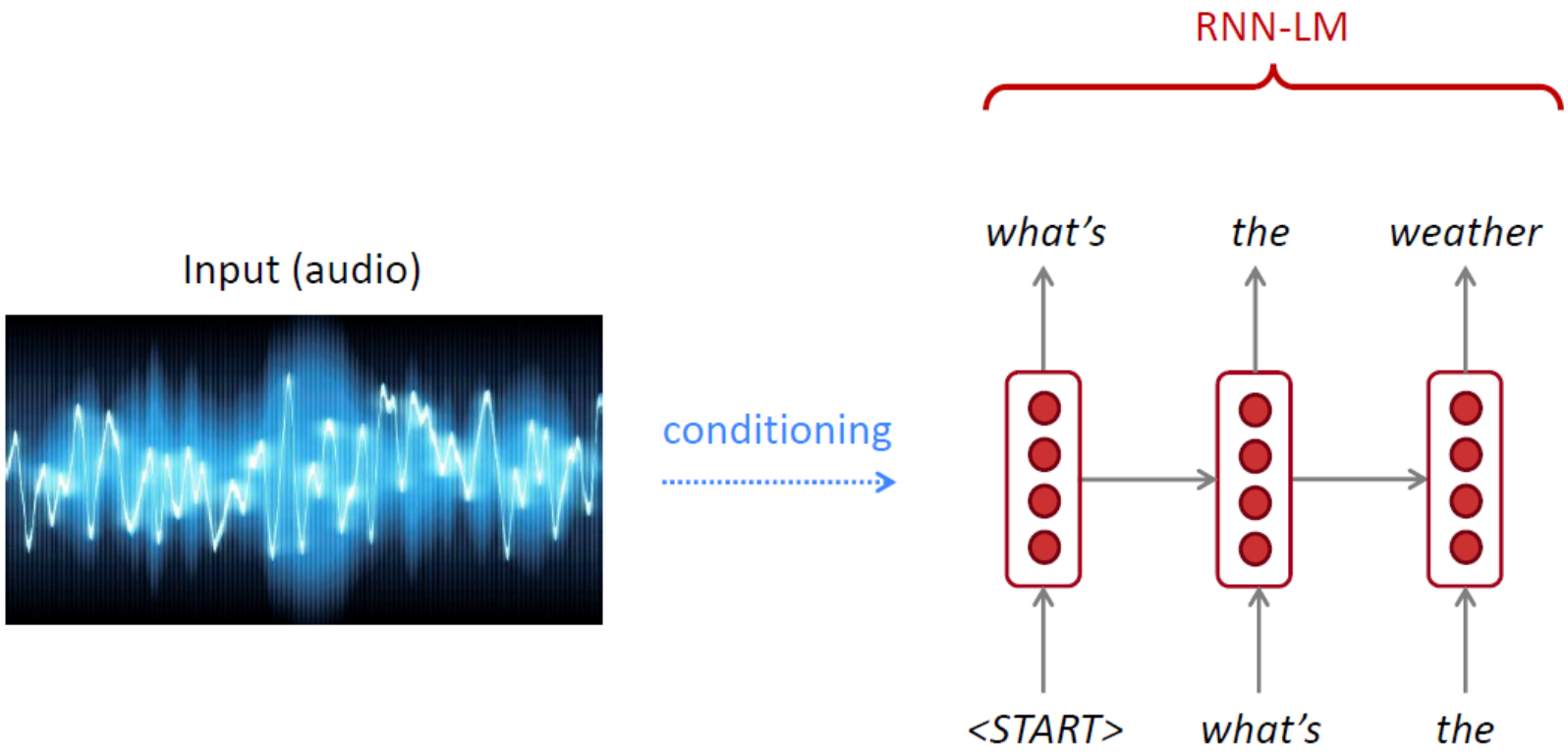
RNN can be used for sentence classification

- e.g. sentiment classification



RNN can be used to generate text

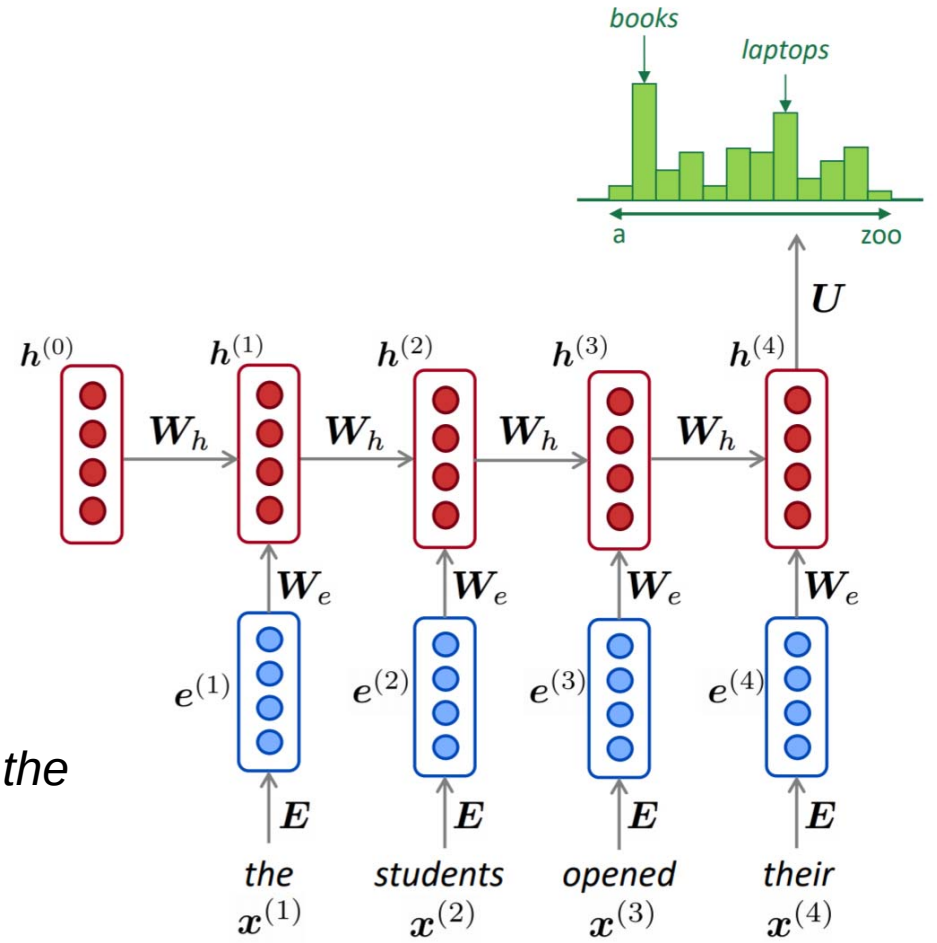
- e.g. speech recognition, machine translation, summarization



Long-Term Dependencies Problem

- Disadvantages of RNN:
  - **memory capacity:** as $h^{(t)}$ continues to accumulate new input information, saturation will occur.

as the proctor started the clock, the students opened their exam.



Long Short Term Memory (LSTM)

- On step *t*, there is a hidden state $\boldsymbol{h}^{(t)}$ *and* a cell state $\boldsymbol{c}^{(t)}$
  - Both are vectors length *n*
  - The cell stores long-term information
  - The LSTM can erase, write and read information from the cell

- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors length *n*
  - On each timestep, each element of the gates can be open (1), closed (0), or somewhere in-between.
  - The gates are dynamic: their value is computed based on the current context

Long Short Term Memory (LSTM)

We have a sequence of inputs $\boldsymbol{x}^{(t)}$, and we will compute a sequence of hidden states $\boldsymbol{h}^{(t)}$ and cell states $\boldsymbol{c}^{(t)}$. On timestep t:

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

Sigmoid function: all gate values are between 0 and 1

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f\boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i\boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o\boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c\boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

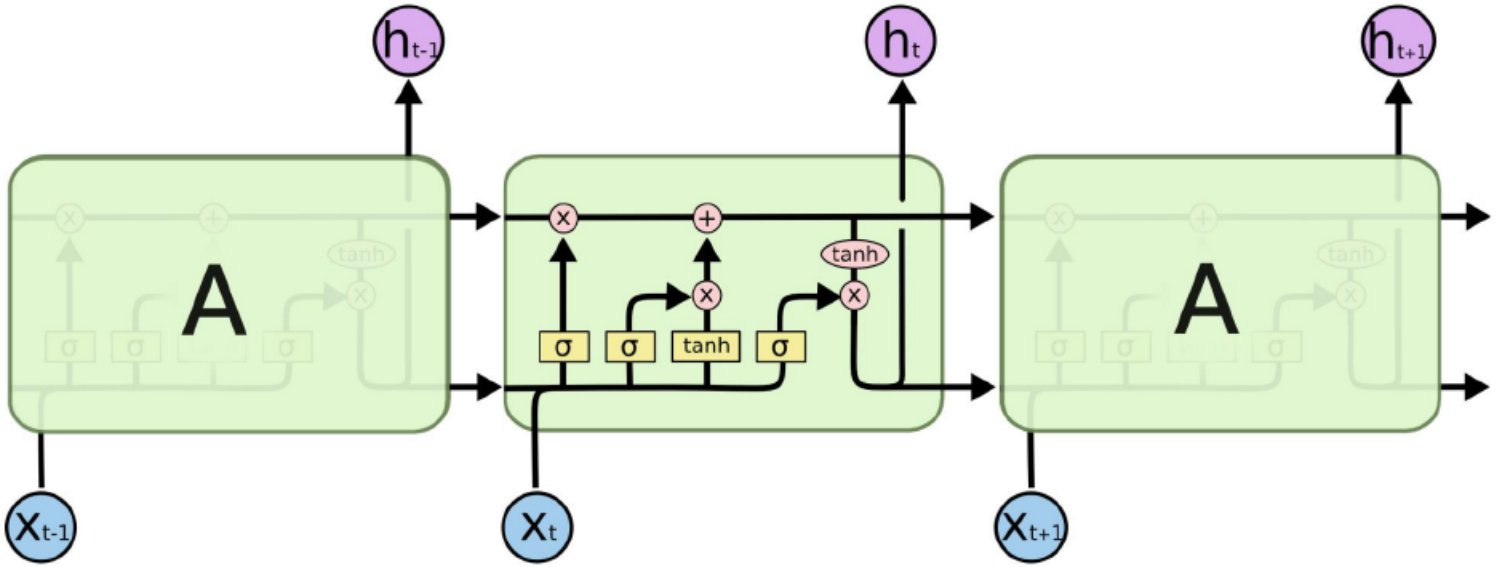
$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}$$

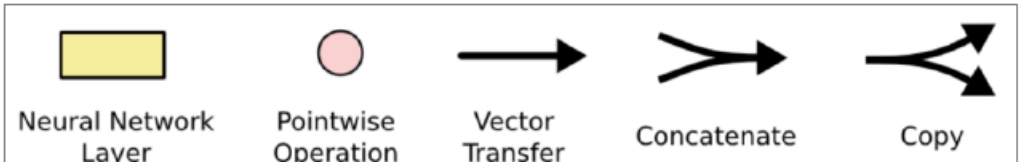
Gates are applied using element-wise product

All these are vectors of same length n

Long Short Term Memory (LSTM)

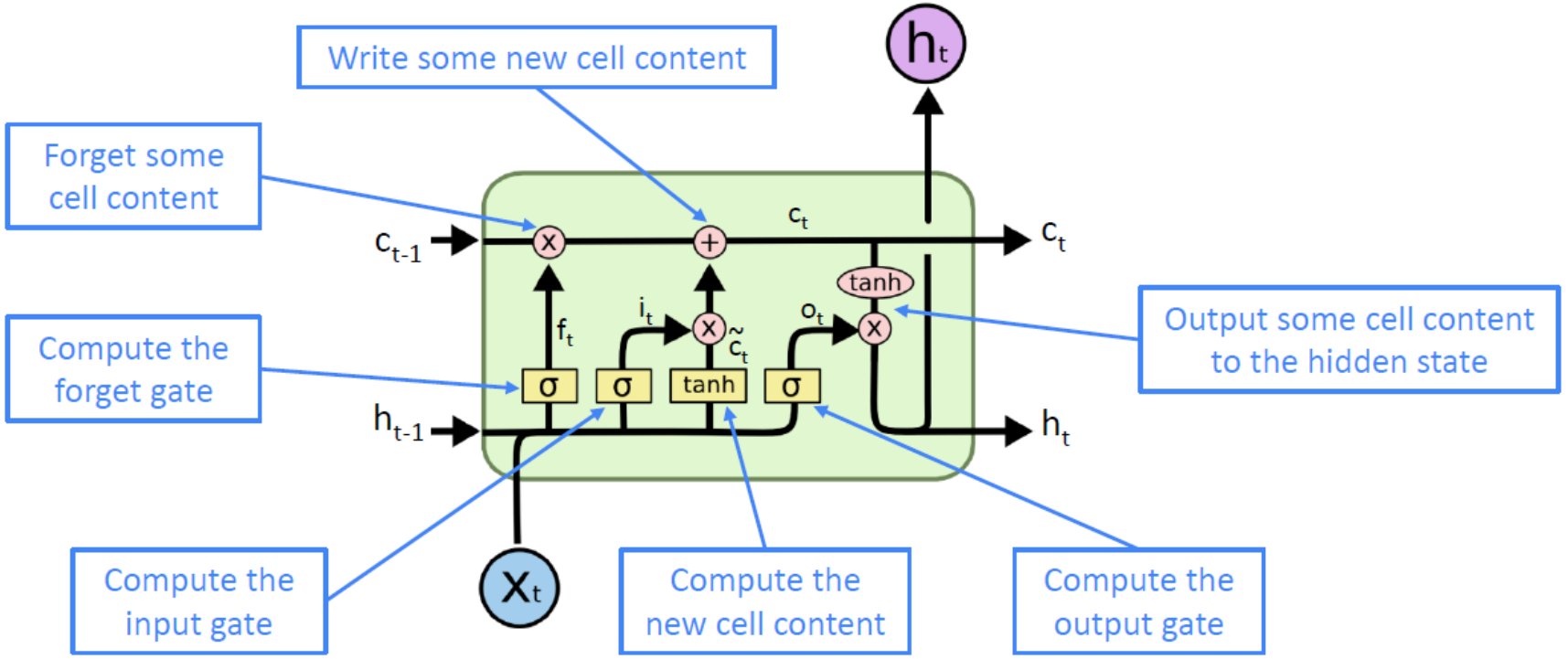
You can think of the LSTM equations visually like this:





Long Short Term Memory (LSTM)

You can think of the LSTM equations visually like this:



Long Short Term Memory (LSTM)

- The LSTM architecture makes it easier for the RNN to preserve information over many timesteps.
- In 2013-2015, LSTMs started achieving state-of-the-art results
  - successful tasks include: handwriting recognition, speech recognition, machine translation, parsing, image captioning
  - LSTM became the dominant approach

Gated Recurrent Units (GRU)

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep *t* we have input $\boldsymbol{x}^{(t)}$ and hidden state $\boldsymbol{h}^{(t)}$ (no cell state).

Update gate: controls what parts of hidden state are updated vs preserved

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_u \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u \boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

Reset gate: controls what parts of previous hidden state are used to compute new content

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r \boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$

New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_h(\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{U}_h \boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{u}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$$

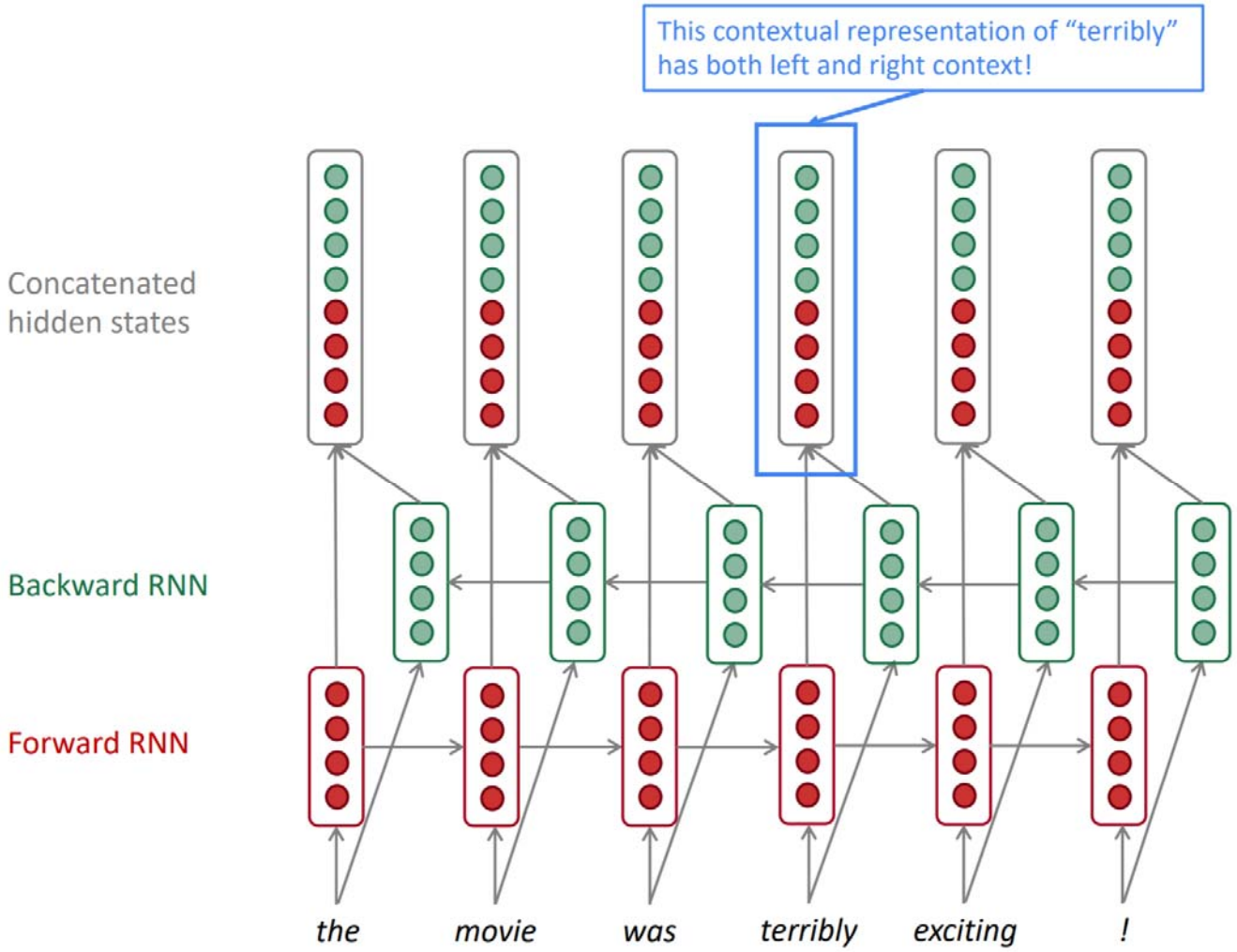
How does this solve vanishing gradient?
Like LSTM, GRU makes it easier to retain info long-term (e.g. by setting update gate to 0)

LSTM vs GRU

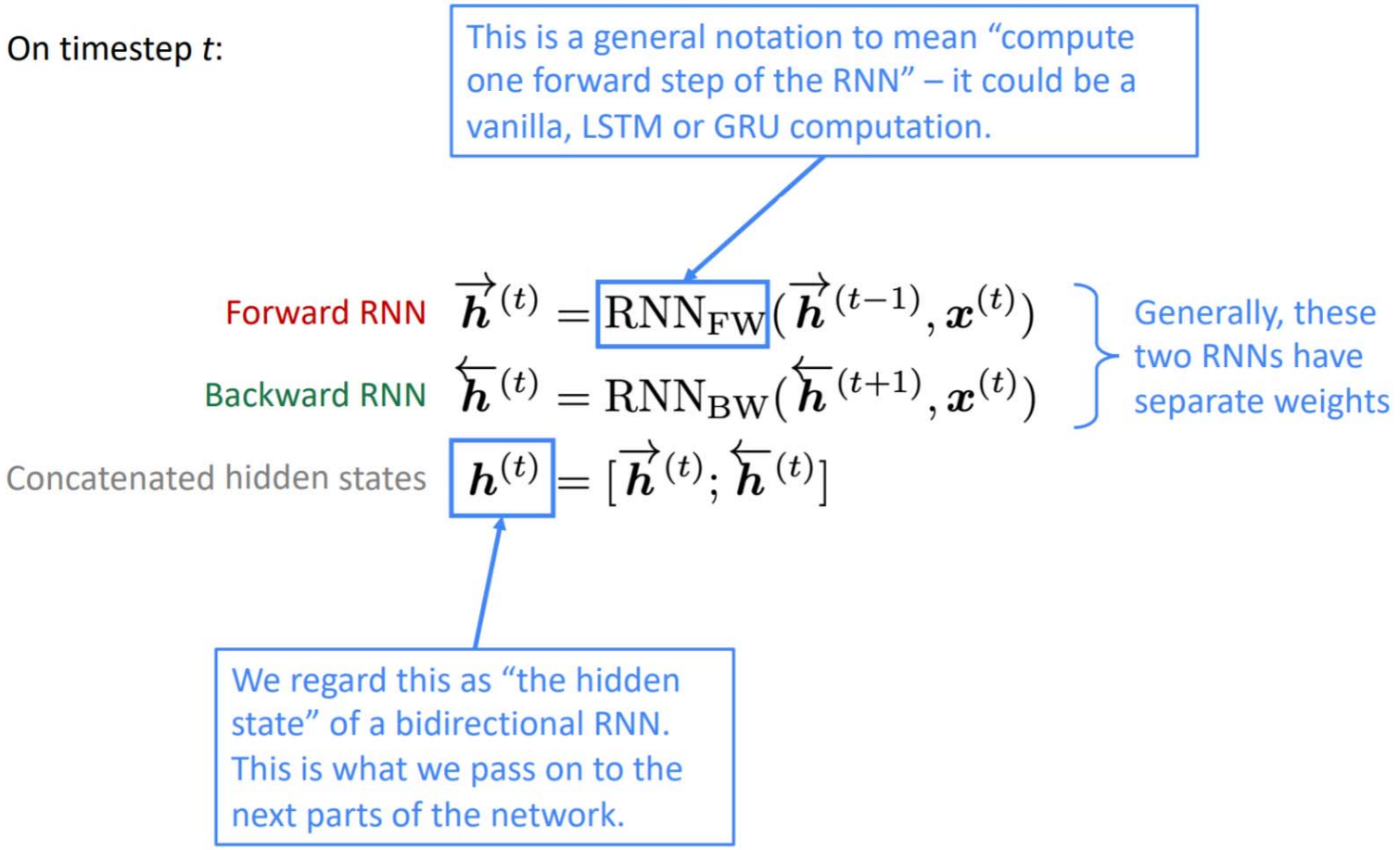
- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used

- Rule of thumb: LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data); Switch to GRUs for speed and fewer parameters.

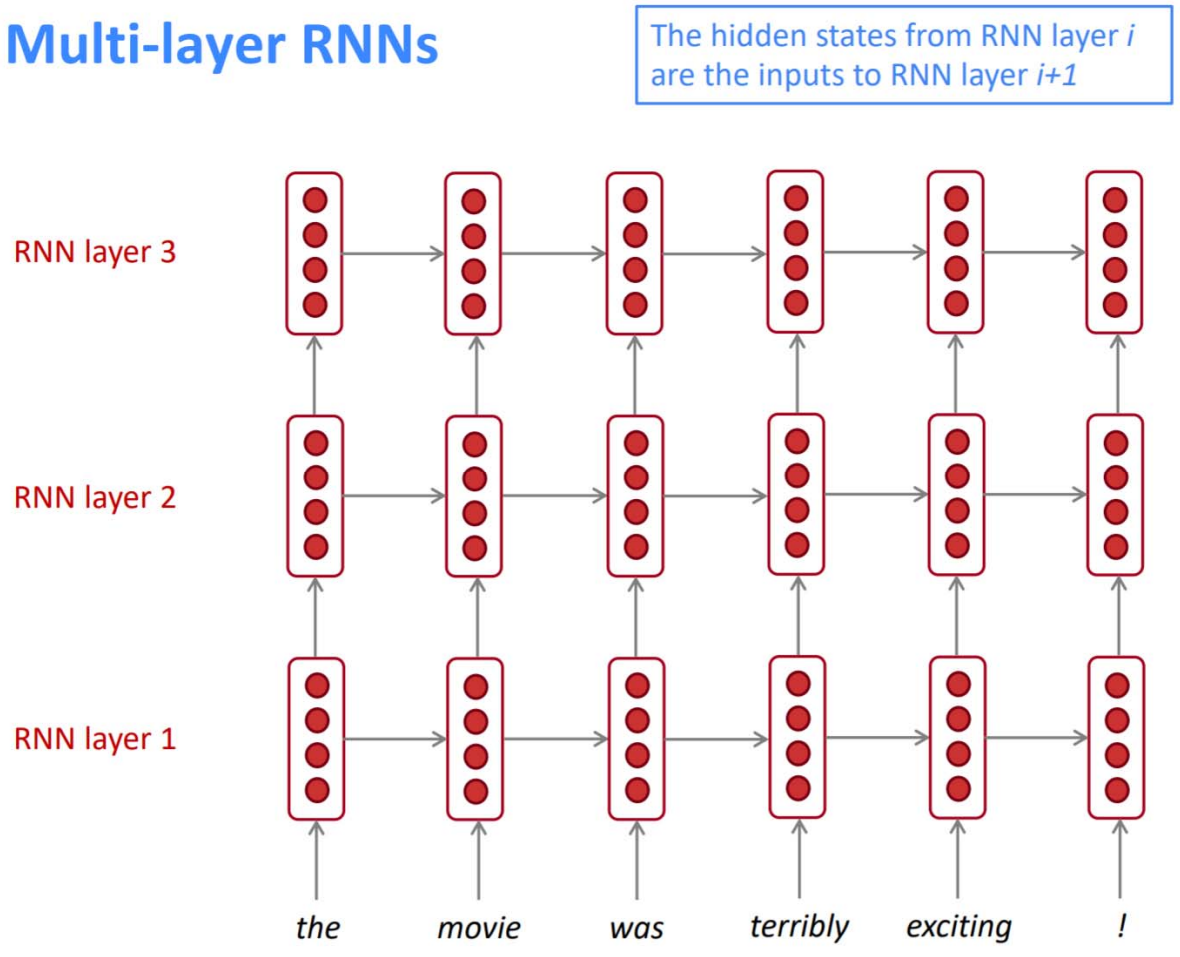
Bidirectional RNNs



Bidirectional RNNs

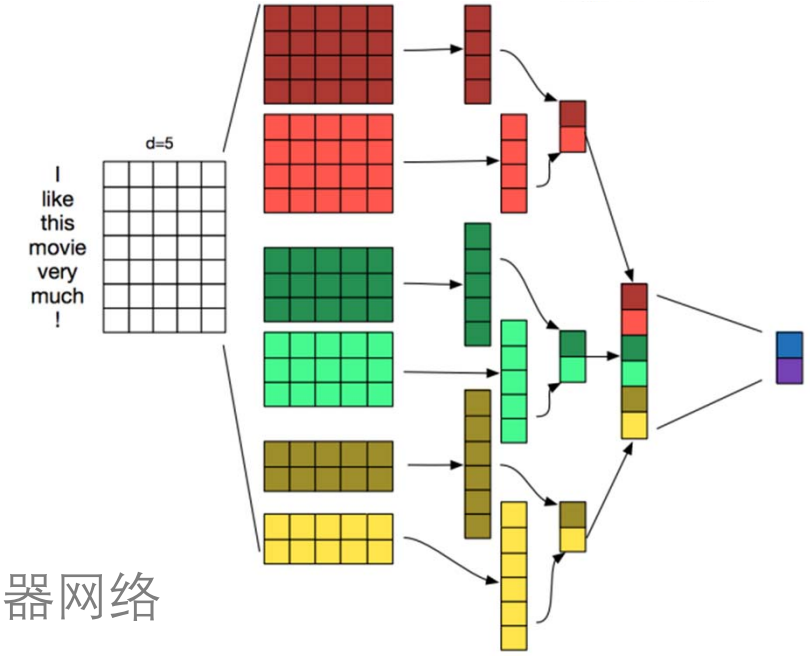


Bidirectional RNNs



Today's class

- From ML to DL
- Neuron and ANN
- DL models
  - RNN 循环神经网络
  - CNN 卷积神经网络
  - Attention 注意力机制
  - Encoder-Decoder 编码器-解码器网络



Convolutional Neural Models

- Image Classification

Image Classification

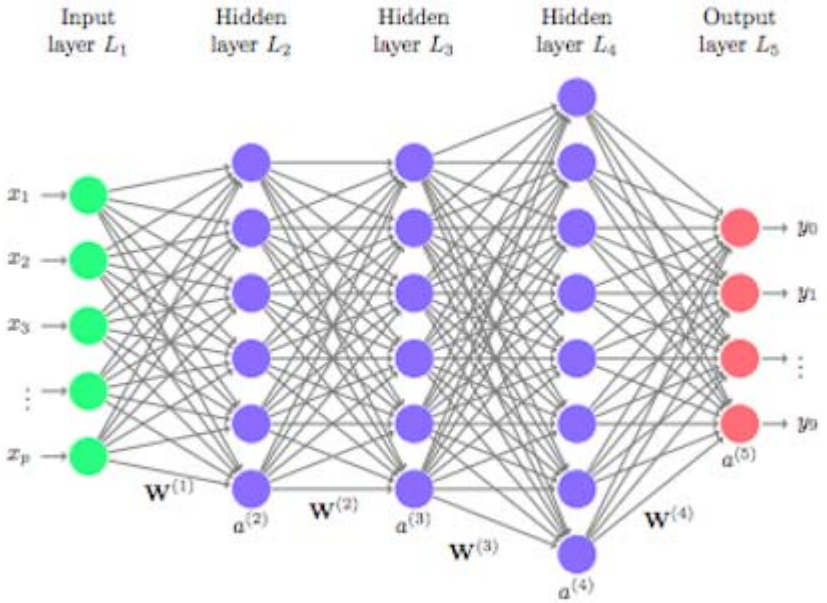
64x64

→ Cat? (0/1)

Convolutional Neural Models

- Image Classification by FNN
  - input size=64*64*3=12288
  - let hidden size=1024
  - $W \in R^{12288\times1024}$
  - two much parameters!

64x64



Convolutional Neural Models

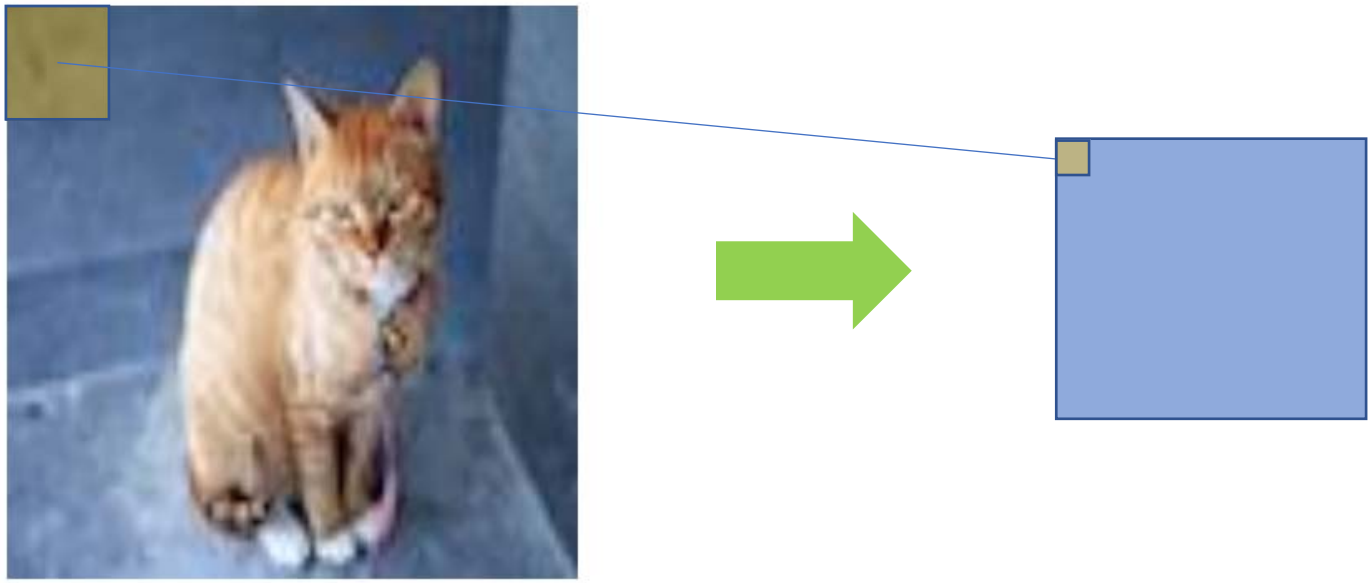
- Motivation

Convolutional Neural Models

- Motivation

Convolutional Neural Models

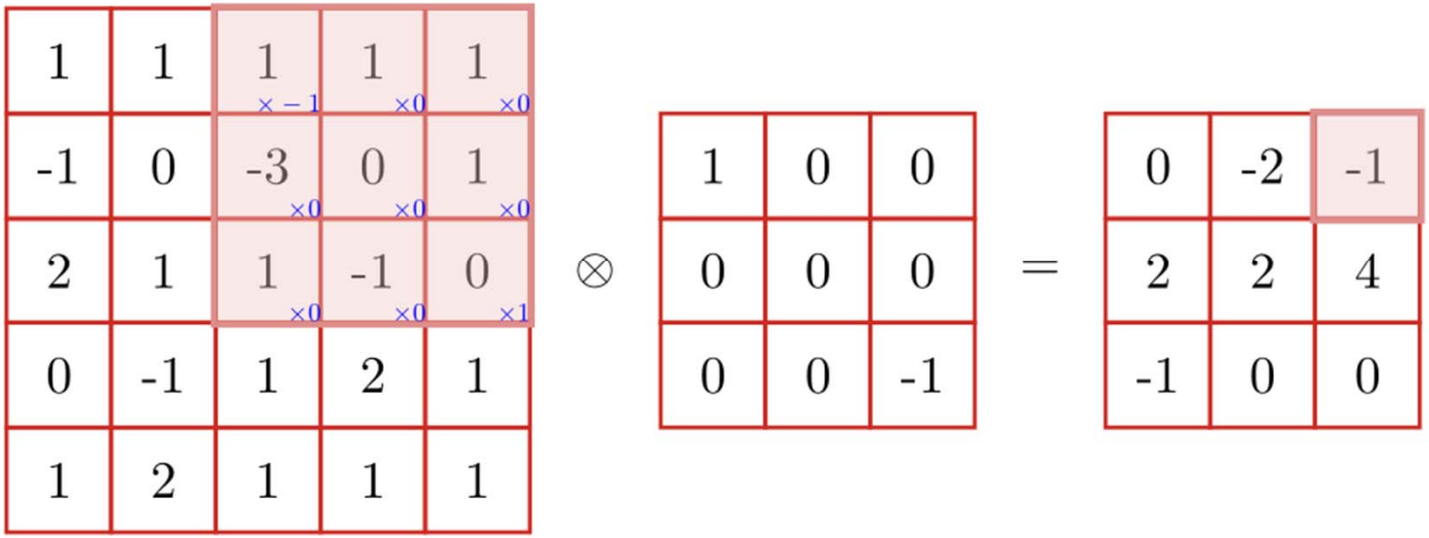
- Motivation



64x64

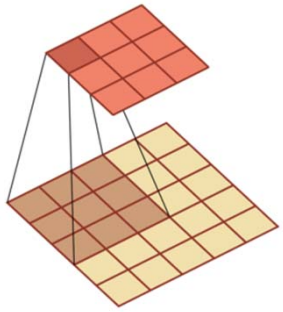
Convolutional Neural Models

- Convolution

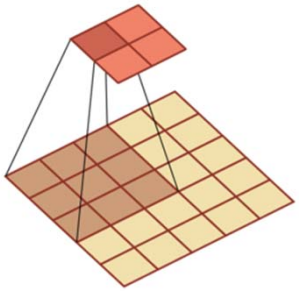


Convolutional Neural Models

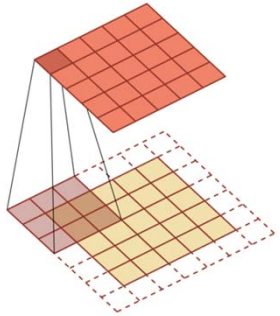
- Convolution



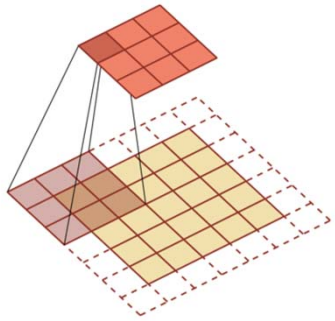
stride(步长)=1，padding(填充)=0



stride=2，padding=0



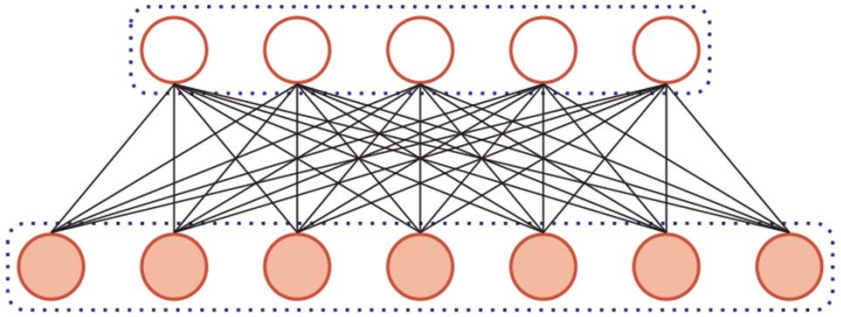
stride=1，padding=1



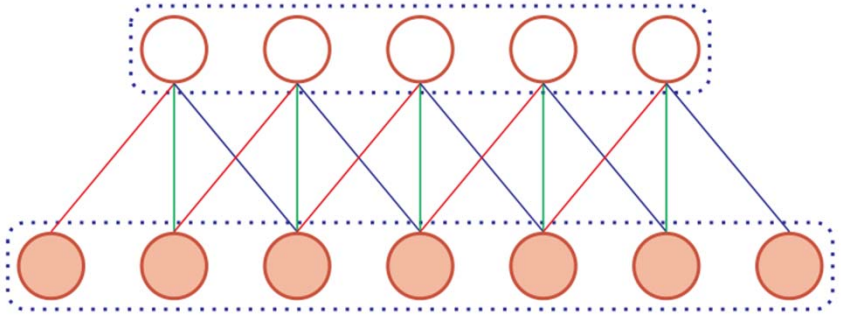
stride=2，padding=1

Convolutional Neural Models

- Replace fully-connected layer with convolution layer



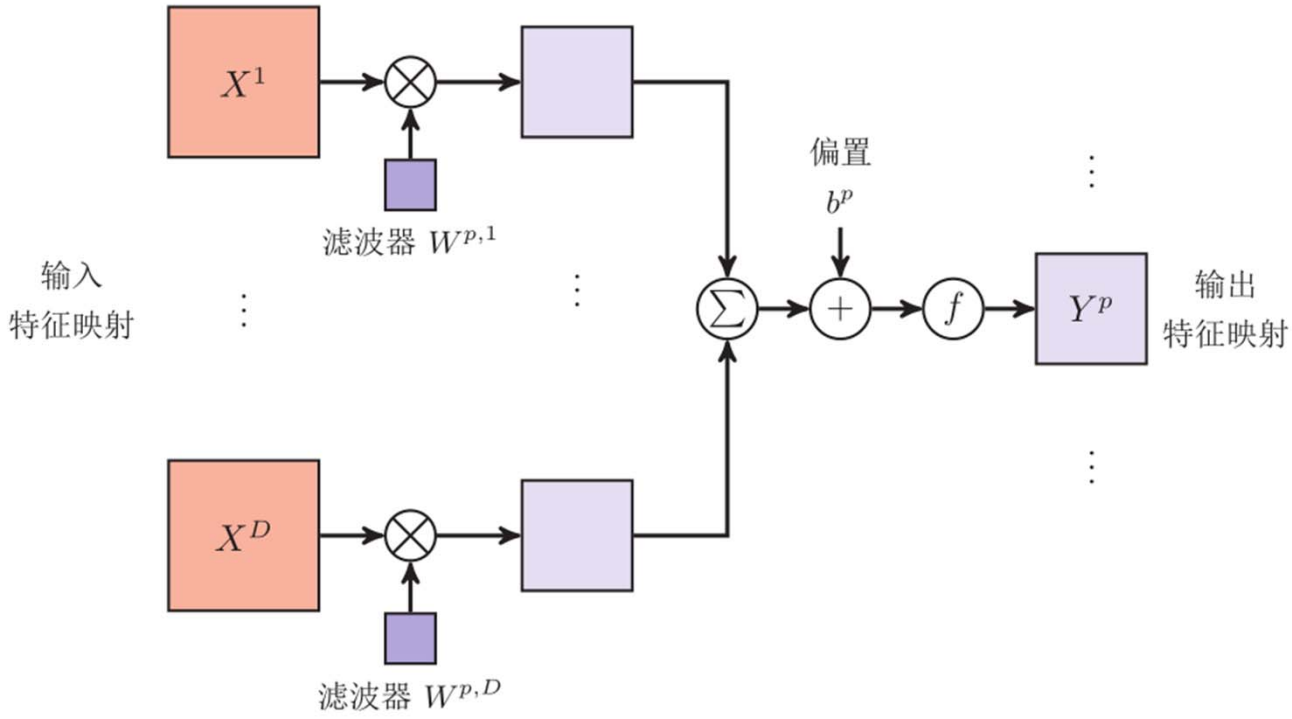
(a) 全连接层



(b) 卷积层

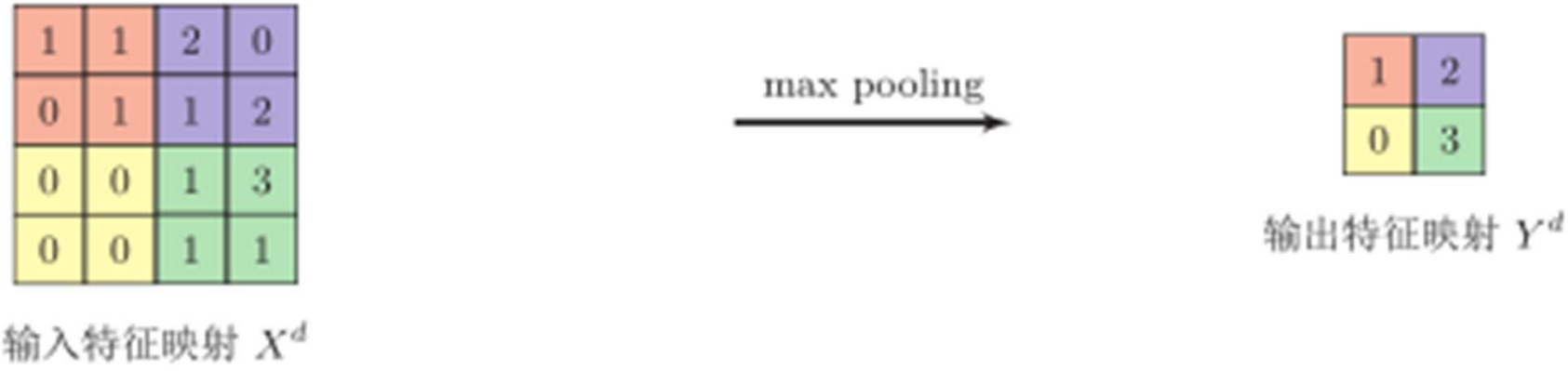
Convolutional Neural Models

- Multiple filter



Convolutional Neural Models

- Pooling Layer

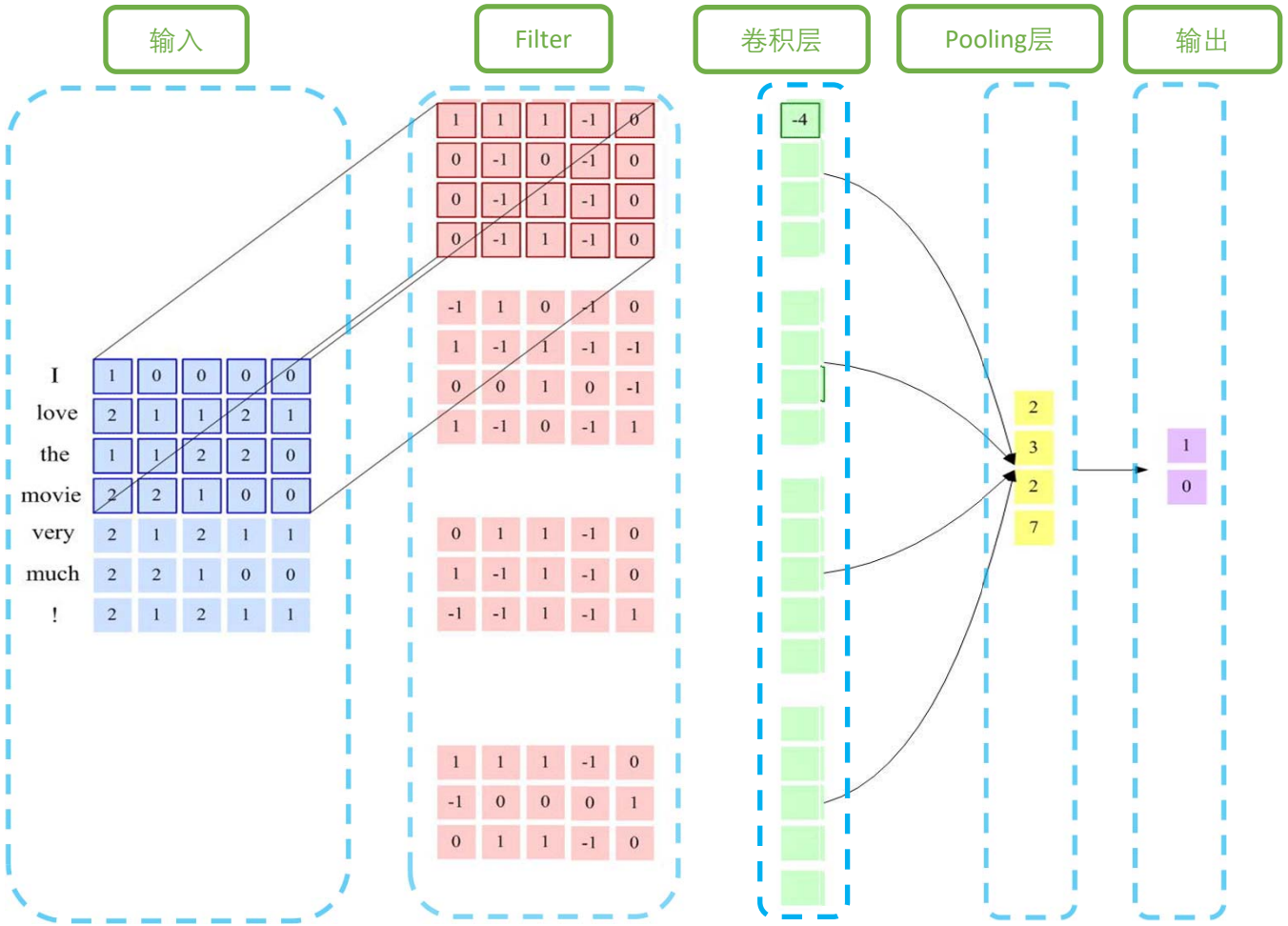


输入特征映射 X^d

输出特征映射 Y^d

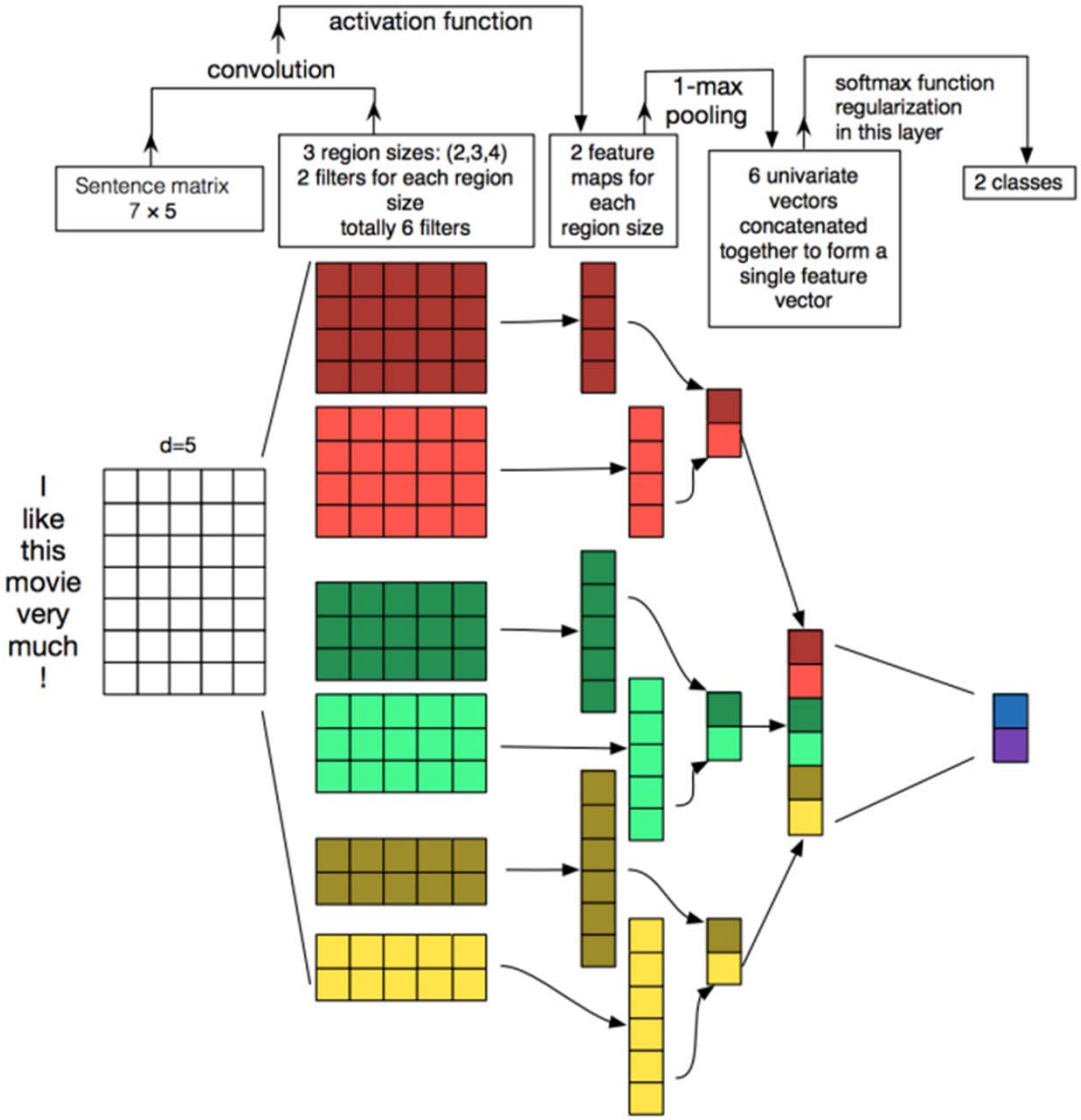
Convolutional Neural Models

- CNN for NLP



Convolutional Neural Models

- TextCNN



Convolutional Neural Models

- TextCNN

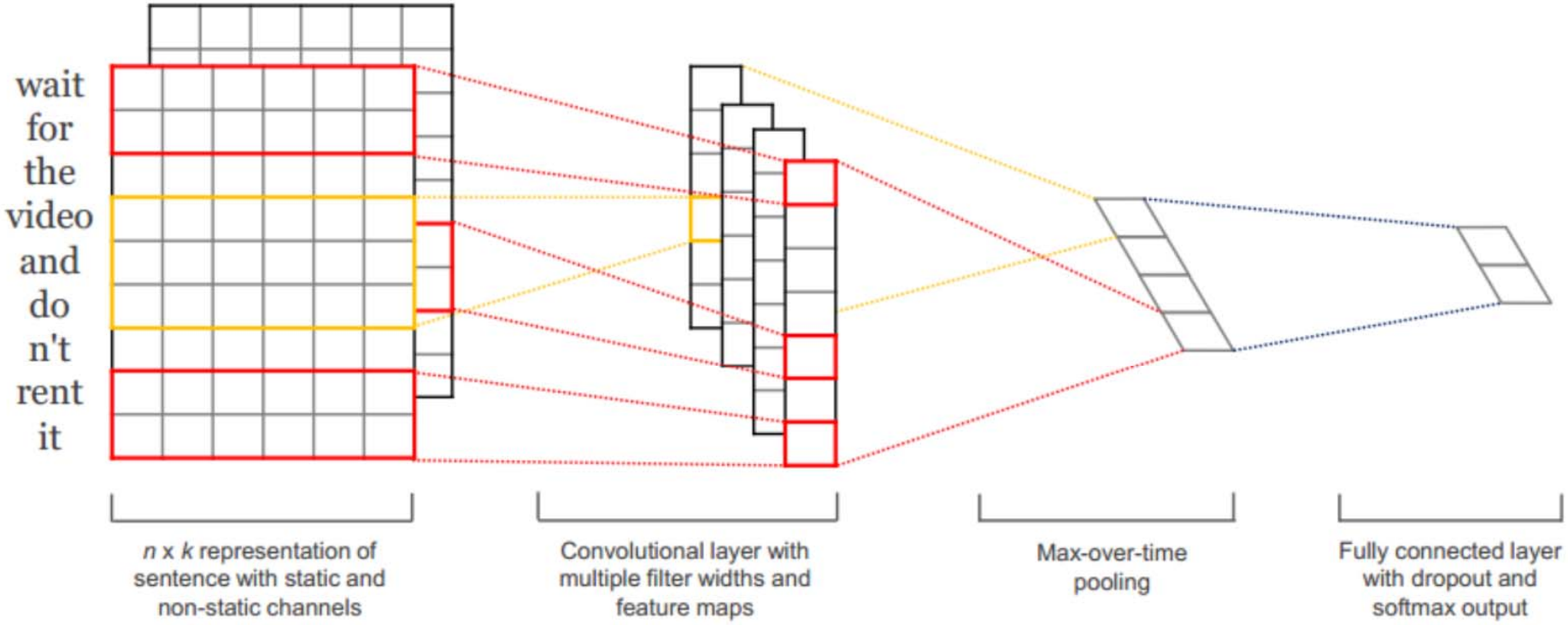


Figure 1: Model architecture with two channels for an example sentence.

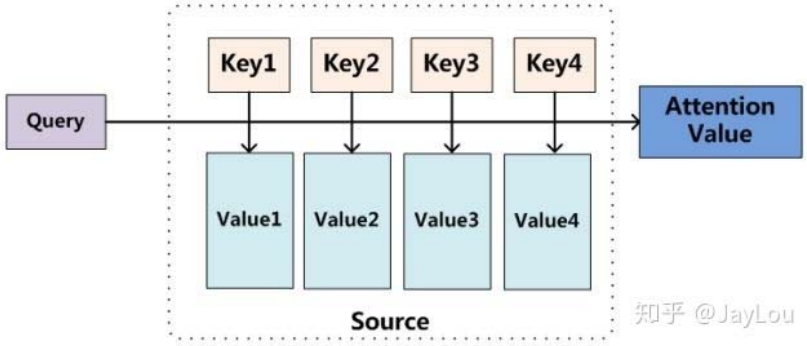
Convolutional Neural Models

- Why does textCNN work?

not recommend"), but not at the document level. Indeed, in trying to determine the polarity of a movie review, we don't really care whether "not bad, quite good" is found at the start or at the end of the document. We just need to capture the fact that "not" precedes "bad", and so forth. Note that CNNs are not able to encode long-range dependencies, and therefore, for some tasks like language modeling, where long-distance dependence matters, recurrent architectures such as LSTMs are preferred.

Today's class

- From ML to DL
- Neuron and ANN
- DL models
  - RNN 循环神经网络
  - CNN 卷积神经网络
  - Attention 注意力机制
  - Encoder-Decoder 编码器-解码器网络



Sentiment Classification

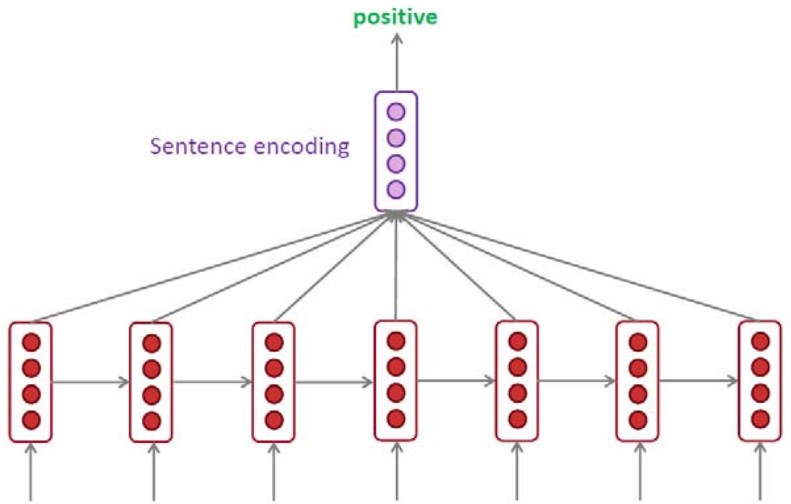
- Task: Aspect-based Sentiment classification
  - given a review and a aspect term, to determine the sentiment polarity expressed by the review on this aspect term.

"The restaurant was expensive, but the menu was great" → **{price: negative, food: positive}**

Sentiment Classification

- Baseline: LSTM
  - construct a new sentence
    - *"Aspect food. The restaurant was expensive, but the menu was great"*
  - input this sentence to LSTM
  - perform max-pooling

$$\mathbf{h} = \text{maxpooling}(\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(T)})$$



"The restaurant was expensive, but the menu was great" → **{price: negative, food: positive}**

Sentiment Classification

- Attention-based LSTM
  - input original sentence to LSTM
    - *"The restaurant was expensive, but the menu was great"*
  - weighted sum

$$\mathbf{h} = \text{maxpooling}(\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(T)}) \qquad \mathbf{h} = \sum_{i=1}^{T} \alpha_i \mathbf{h}^{(t)}$$

    - large weight for "***menu***" and "***great***"
    - Small weight for "***restaurant***", "***expensive***" and "***the***"

Sentiment Classification

- Attention-based LSTM
  - weighted sum

~~$\mathbf{h} = \text{maxpooling}(\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(T)})$~~ $\quad \mathbf{h} = \sum_{i=1}^{T} \alpha_i \mathbf{h}^{(t)}$

  - e.g., a good weight vector

The	*restaurant*	*was*	*expensive*	*,*	*but*	*the*	*menu*	*was*	*great*
0.01	*0.01*	*0.01*	*0.01*	*0.01*	*0.01*	*0.02*	*0.45*	*0.02*	*0.45*

Sentiment Classification

- Attention-based LSTM
  - how to obtain the weight vector?
    - calculate the correlation between each word and the given aspect and then normalize

$$s_t = \boldsymbol{h}^{(t)\top} W \boldsymbol{h}_{food}$$

$$\alpha_i = \mathrm{softmax}(a_t) = \frac{\exp(s_t)}{\sum_i \exp(s_i)}$$

Attention

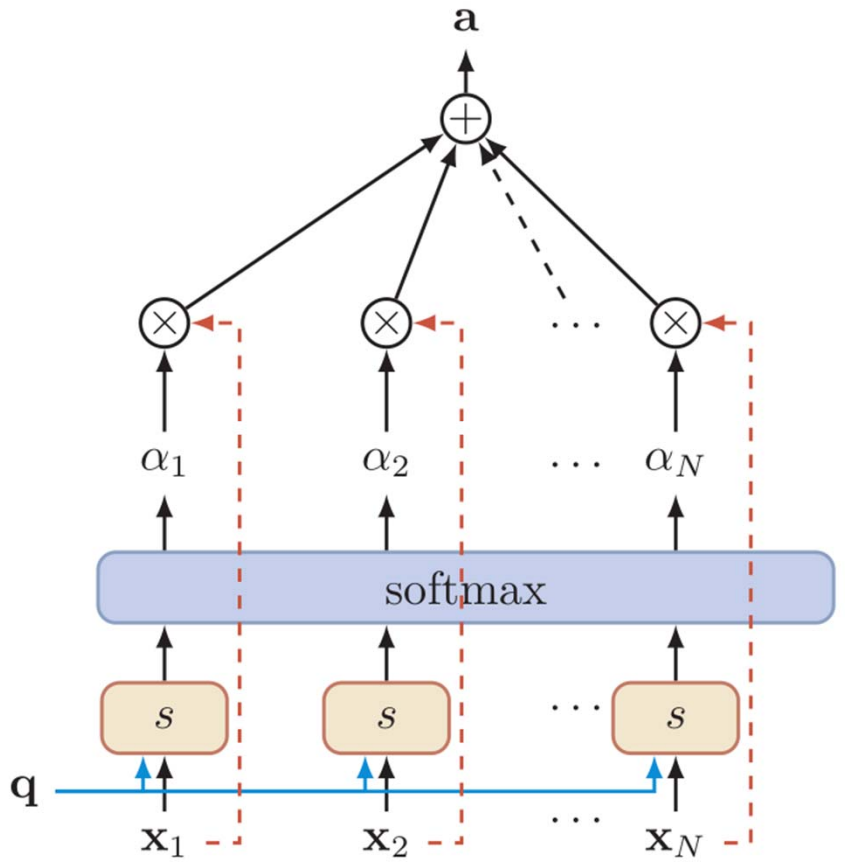
- Formal definition: (K-Q-V).
  - given query $\boldsymbol{q}$ and keys $\mathbf{x}_{1:N}$

$$\begin{aligned}\alpha_i &= p(z=i|\mathbf{x}_{1:N},\mathbf{q}) \\ &= \text{softmax}\left(s(\mathbf{x}_i,\mathbf{q})\right) \\ &= \frac{\exp\left(s(\mathbf{x}_i,\mathbf{q})\right)}{\sum_{j=1}^{N}\exp\left(s(\mathbf{x}_j,\mathbf{q})\right)},\end{aligned}$$

  - $s(\mathbf{x}_i, \boldsymbol{q})$ is the score function

$$s(\mathbf{x}_i,\mathbf{q}) = \mathbf{x}_i^{\mathrm{T}}\mathbf{q},$$

$$s(\mathbf{x}_i,\mathbf{q}) = \mathbf{x}_i^{\mathrm{T}}W\mathbf{q},$$

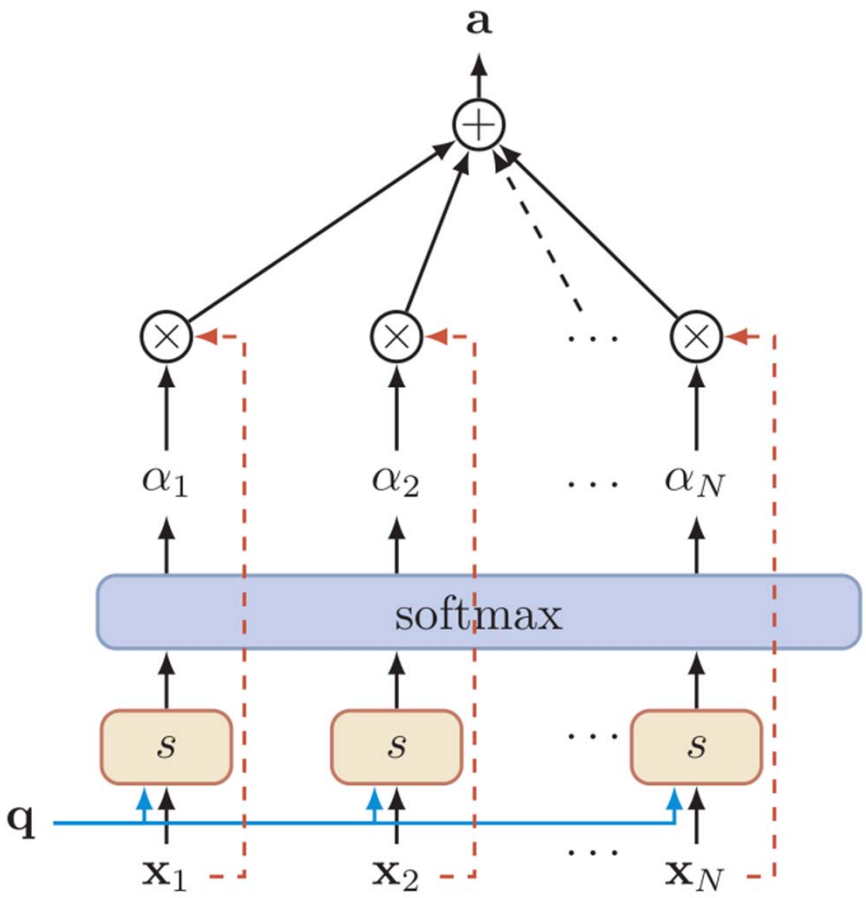


Attention

- Formal definition: (K-Q-V).
  - given value $\mathbf{v}_{1:N}$

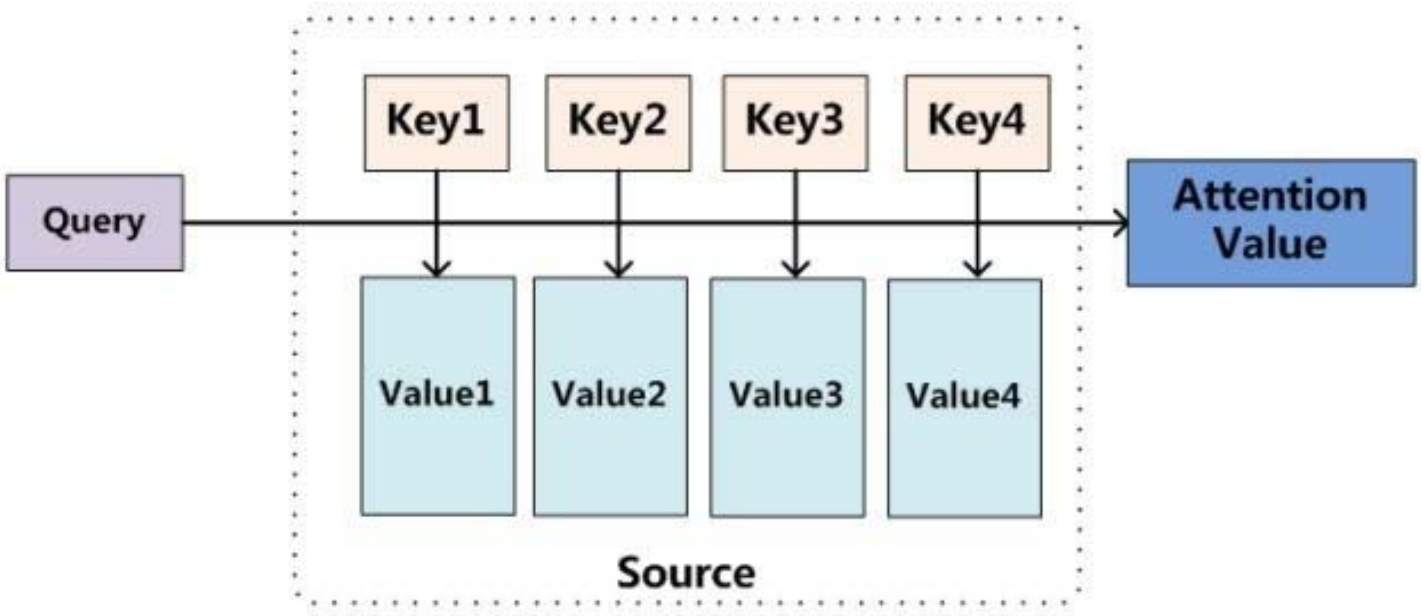
$$a = \sum_i \alpha_i \mathbf{v}_i$$

  - a is the output.



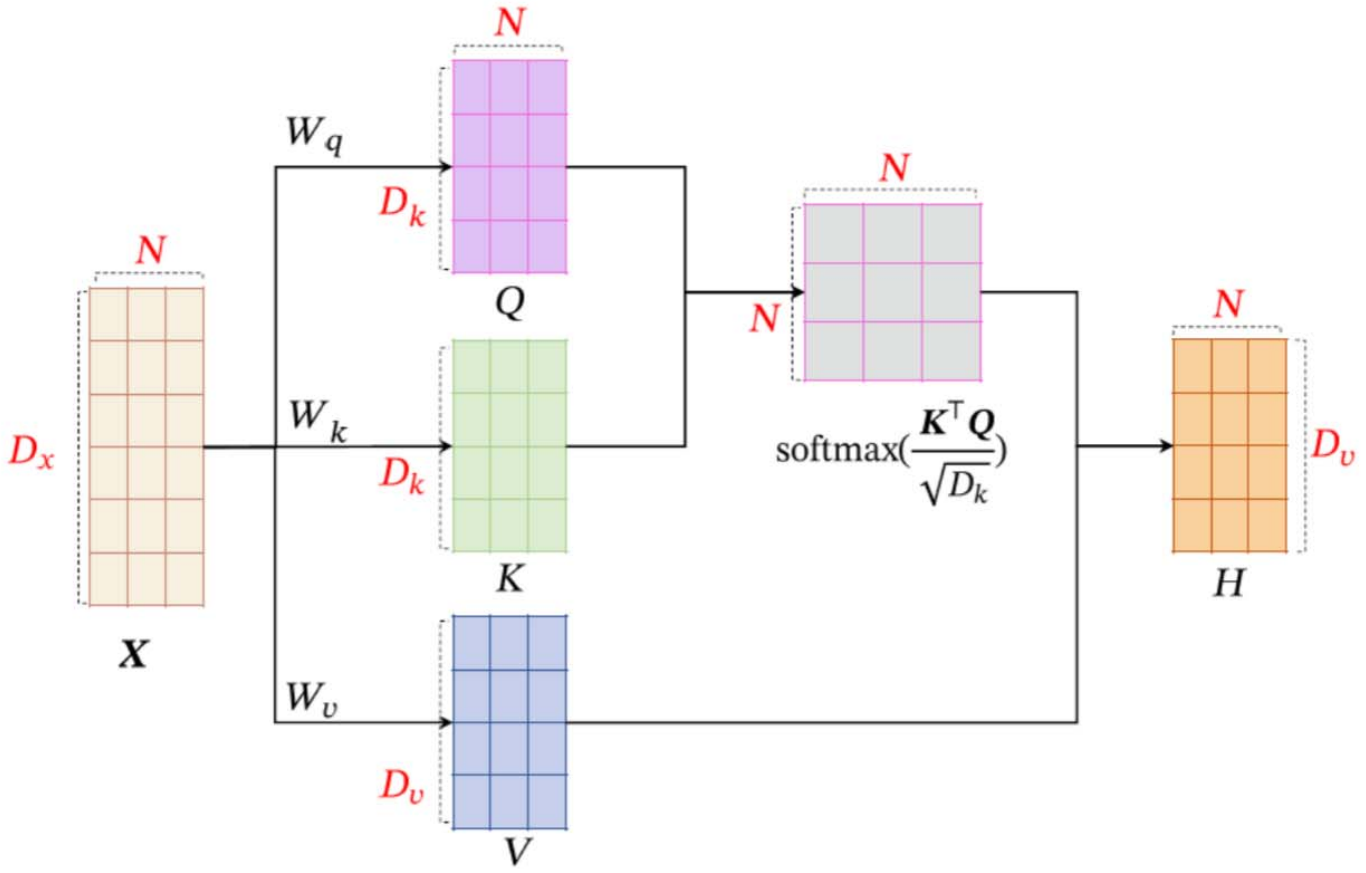
Attention

- Formal definition: (K-Q-V).



Self-Attention

- Self-Attention

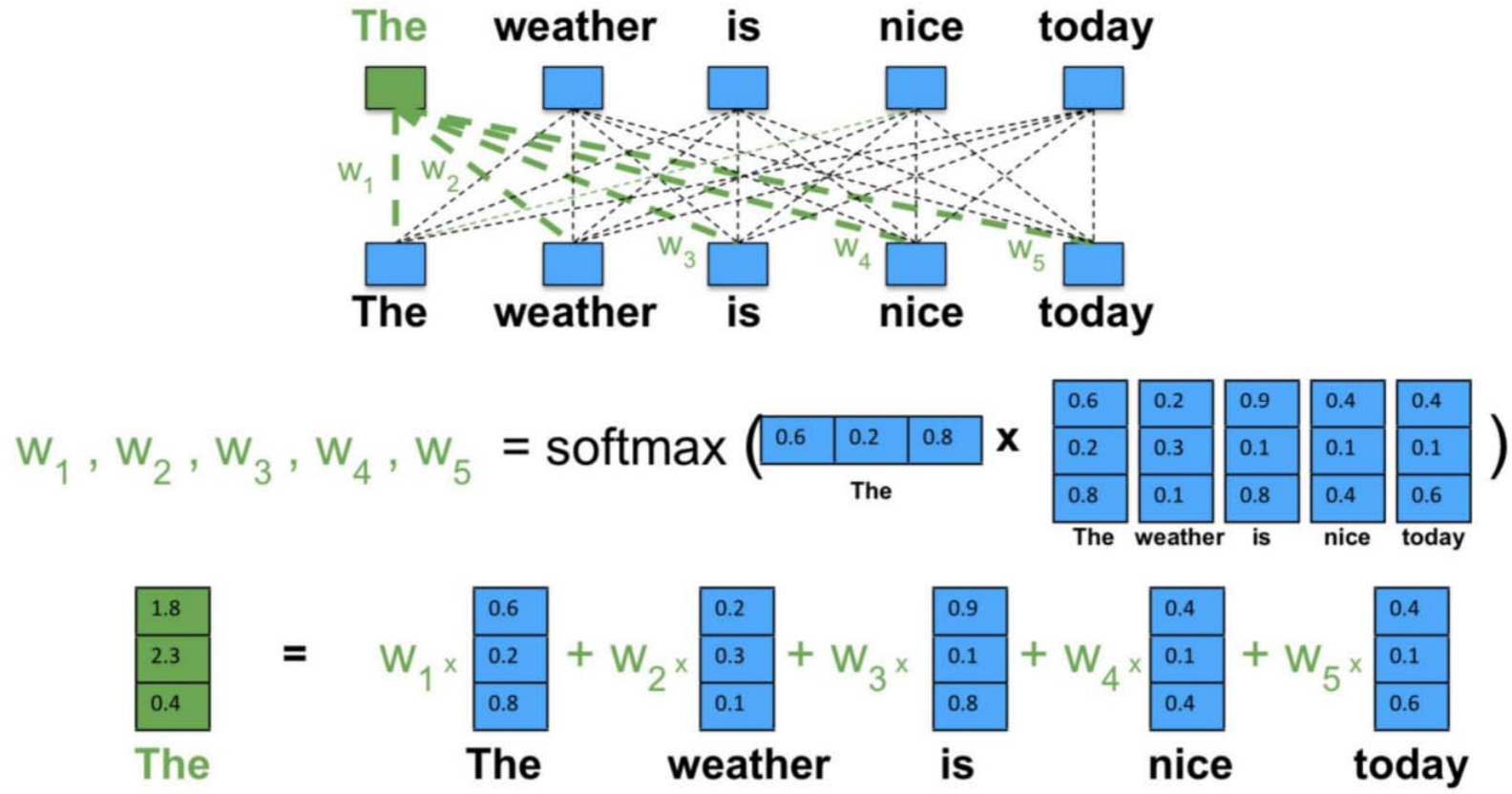


Attention in RNN

- Global Attention
- Local Attention
- Bi-directional Attention
- Self Attention
- Key Value Attention

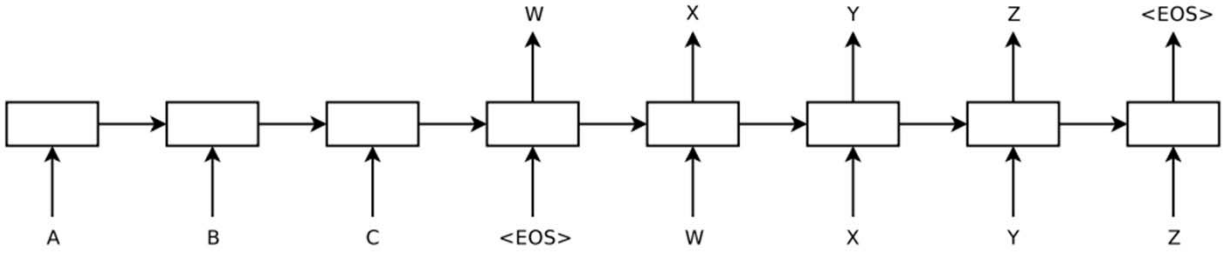
Self-Attention

- Self-Attention



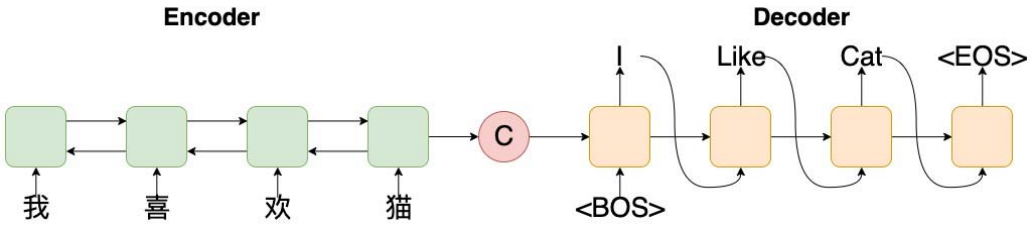
Today's class

- From ML to DL
- Neuron and ANN
- DL models
  - RNN 循环神经网络
  - CNN 卷积神经网络
  - Attention 注意力机制
  - Encoder-Decoder 编码器-解码器网络



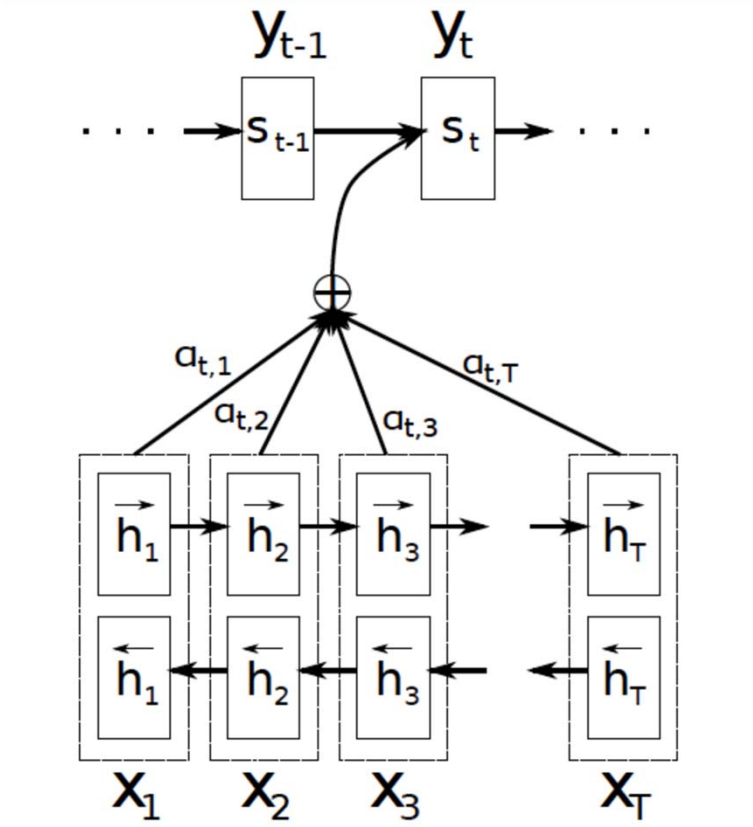
Encoder-Decoder

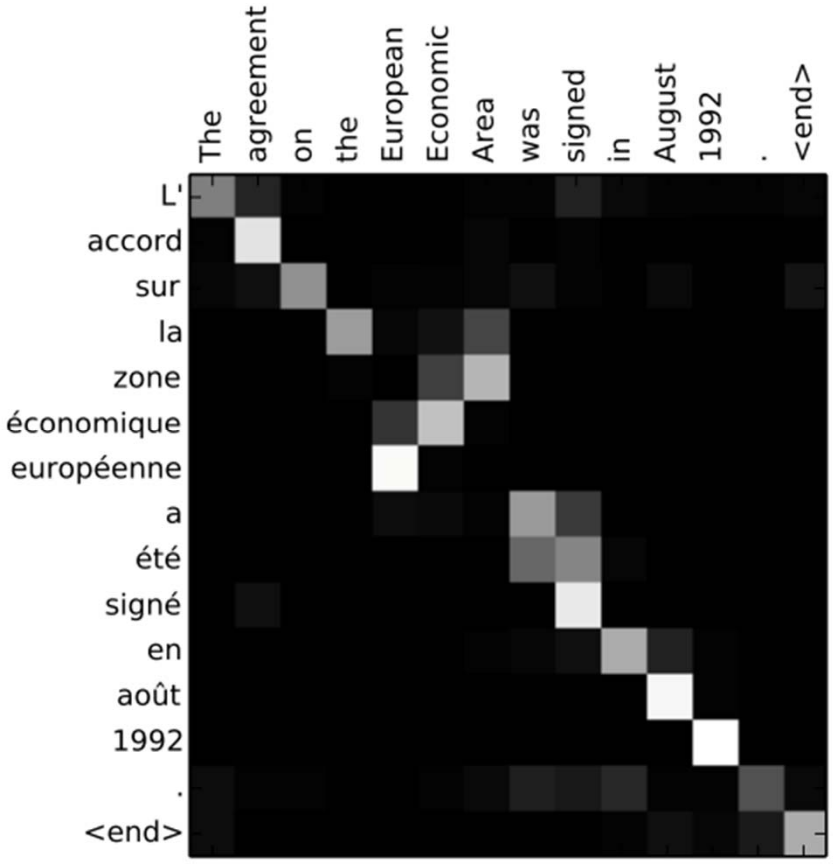
- Sequence to sequence
  - core component
    - RNN (GRU、LSTM)
  - extension
    - with attention mechanism
    - with copy mechanism
    - ……
  - decoding strategy
    - Greedy decoding
    - Beam search
    - ……



Encoder-Decoder

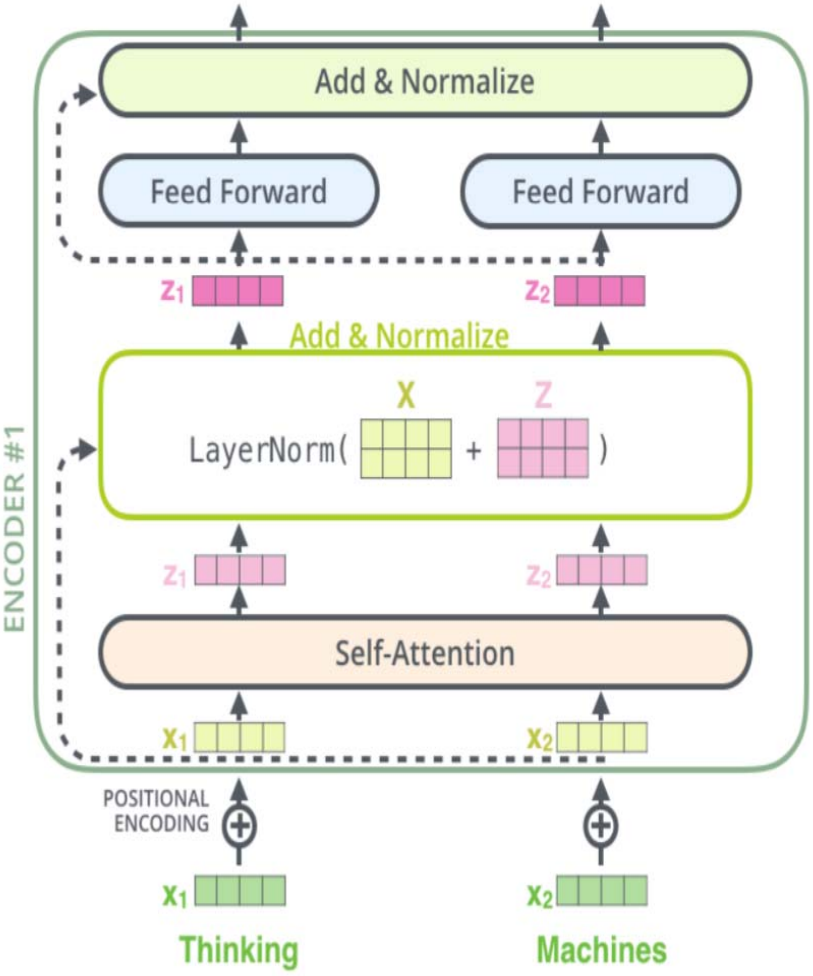
- Sequence to sequence



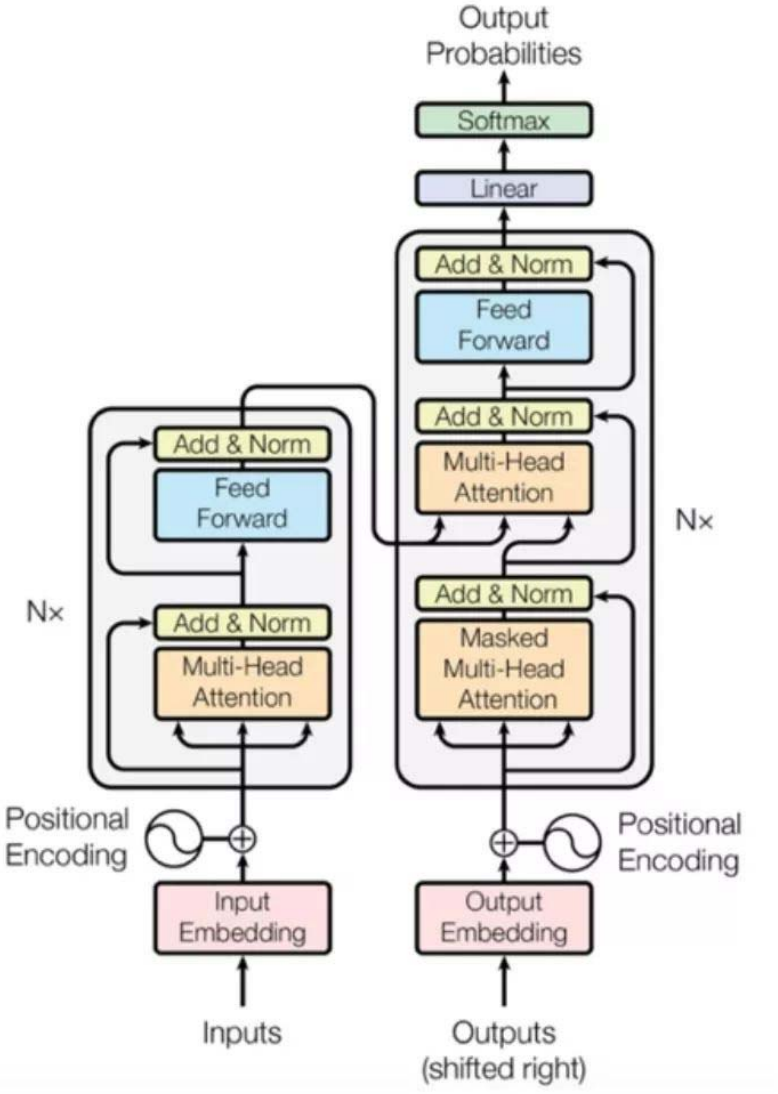


Transformer

- Transformer
  - New framework
  - core component
    - Self-attention
    - Multi-head attention
  - other components
    - Position embedding 位置编码
    - Layer norm 层归一化
    - Residual connection 残差连接
    - FNN
  - Fit to parallel training



Transformer



- Transformer apply masks to the input of first multi-head attention module to avoid seeing potential “future” sequence.
- Transformer may be applied to sequence transformation, and different input/output modal

Reference

1. 邱锡鹏. 神经网络与深度学习.
2. 自然语言处理中的自注意力模型 https://xpqiu.github.io/slides/20200613-CAAI-%E8%87%AA%E7%84%B6%E8%AF%AD%E8%A8%80%E5%A4%84%E7%90%86%E4%B8%AD%E7%9A%84%E8%87%AA%E6%B3%A8%E6%84%8F%E5%8A%9B%E6%A8%A1%E5%9E%8B.pdf
3. Recurrent Neural Networks and Language Models. http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture06-rnnlm.pdf
4. Vanishing Gradients and Fancy RNNs. http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture07-fancy-rnn.pdf
5. Tixier A J P. Notes on deep learning for nlp[J]. arXiv preprint arXiv:1808.09772, 2018.

The next lecture

- Lecture 5: Language Model