社区说

Community Talks

Introduction on Deep Learning
深度学习基础

Xihan Li 李锡涵
Department of Computer Science, University College London
Google Developers Expert in Machine Learning
xihan.li@cs.ucl.ac.uk
https://snowkylin.github.io

Mar 2024

Outline 大纲

在进入细节之前……

- 概述
- 深度学习历史

深度学习的基石

- 神经元的计算模型
- 梯度下降
- 多层神经网络和反向传播

简单而有用的模型

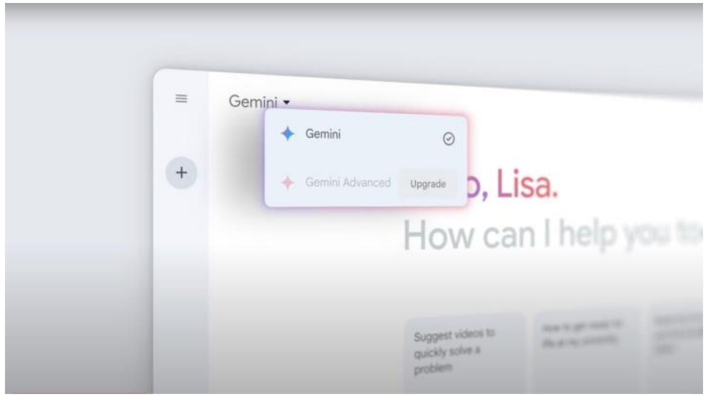
- 用于图像的卷积神经网络(CNN)
- 用于文本的循环神经网络(RNN

当前的最佳实践

- Transformer模型(Gemini等对话生成产品背后的模型)

?

Deep Neural Networks are everywhere
深度神经网络无处不在



Text (audio)
(Dialogue generation
via Gemini)

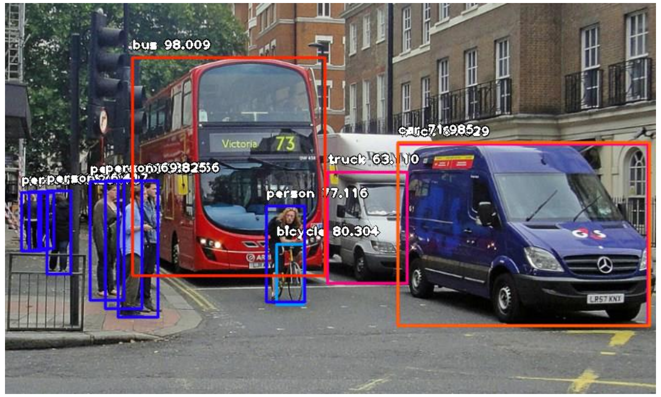
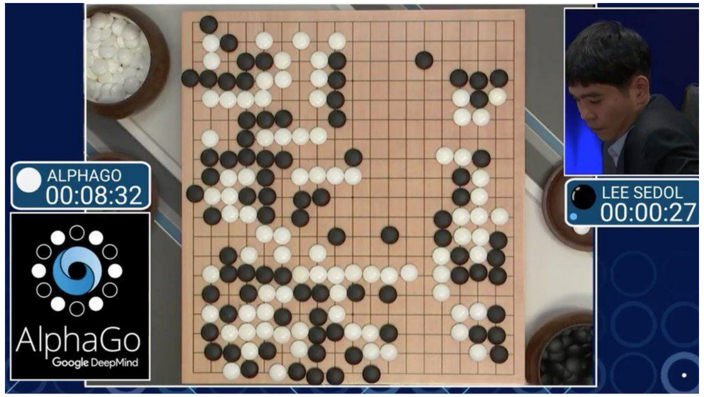
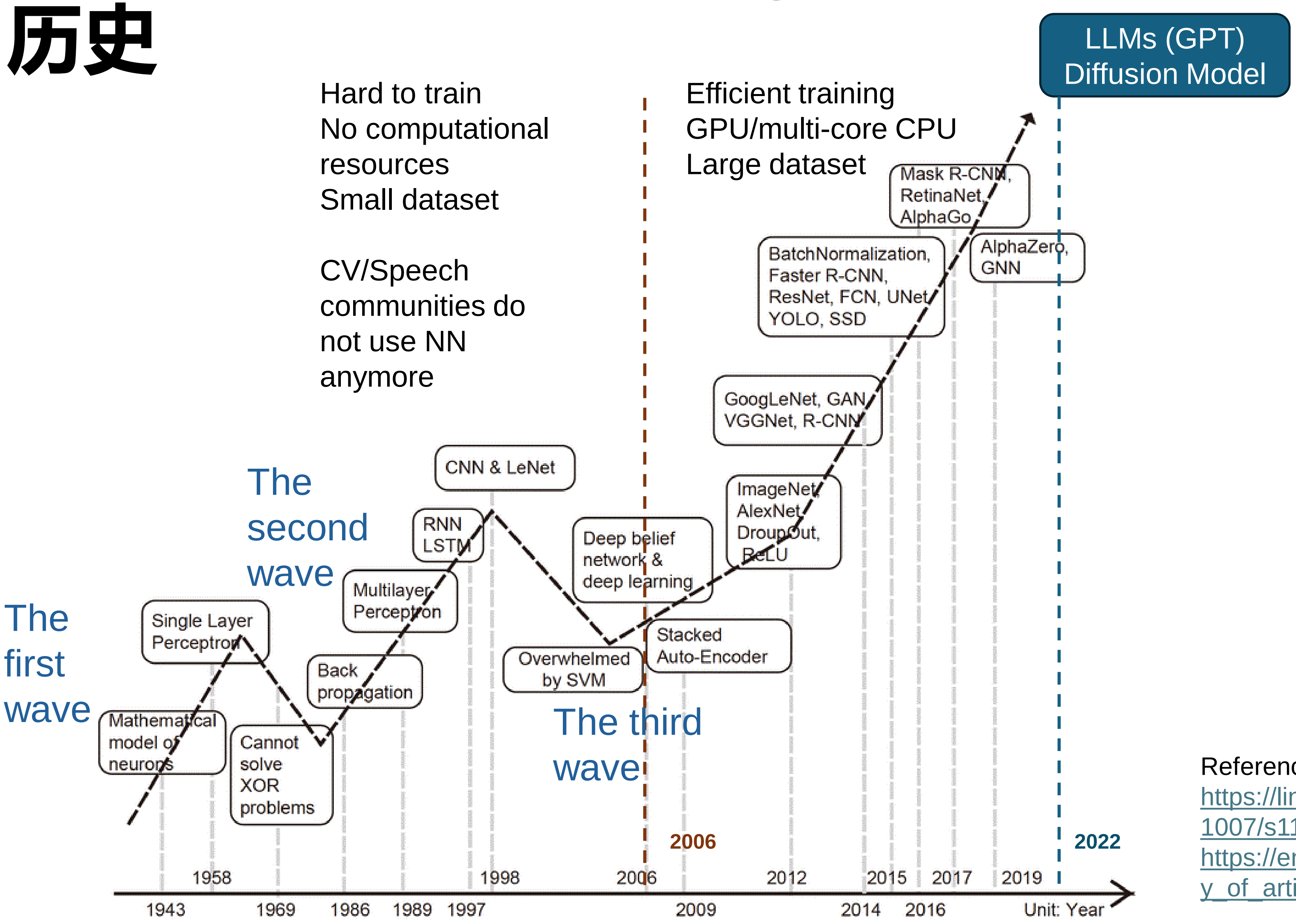


Image (video)
(Object Detection
via YOLO)



Decision Making
(Playing Go via AlphaGo)

History of Deep Learning
深度学习历史



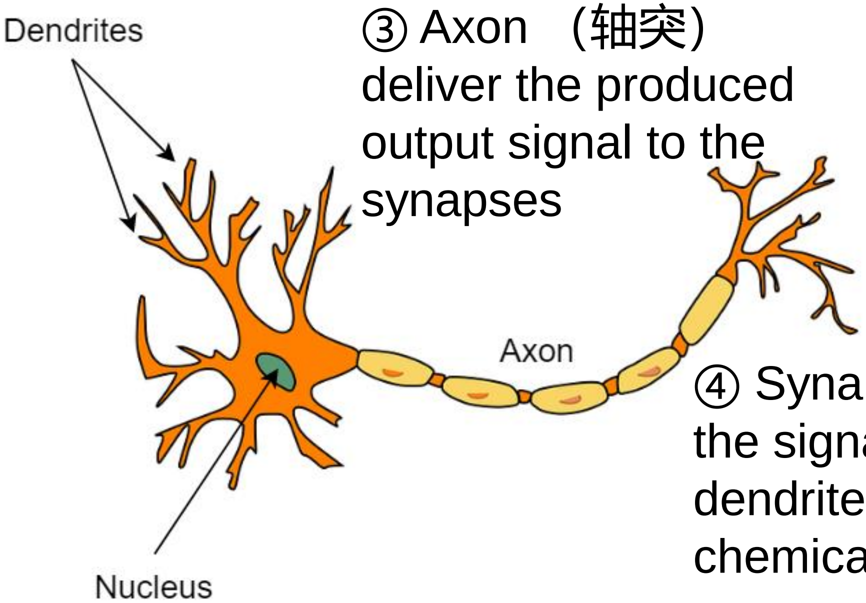
Reference:
https://link.springer.com/article/10.1007/s11430-019-9584-9
https://en.wikipedia.org/wiki/History_of_artificial_intelligence

Cornerstones 深度学习的基石

- Computational model of a neuron 神经元的计算模型
- Gradient Descent 梯度下降
- Multilayer Neural Networks and Backpropagation 多层神经网络与反向传播

Computational model of a neuron
神经元的计算模型

① Dendrites （树突） receive input signals from other neurons

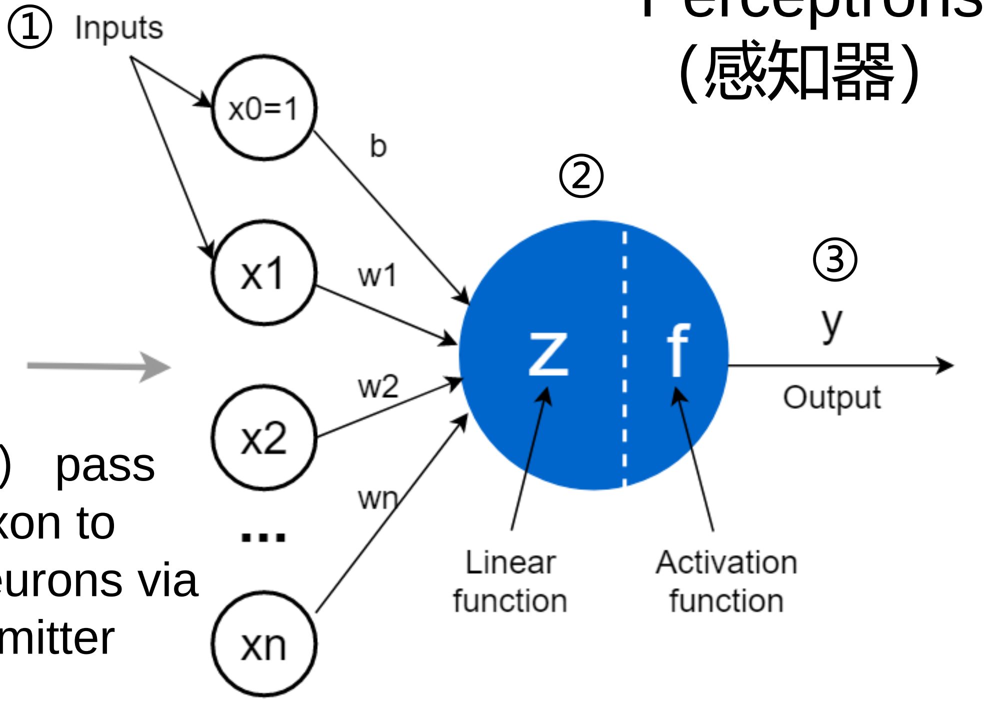


③ Axon （轴突） deliver the produced output signal to the synapses

④ Synapses （突触） pass the signal from an axon to dendrites of other neurons via chemical neurotransmitter

② Nucleus （细胞核） process the input signals (different inputs have different importance) and produce an electrical output signal

Perceptrons
(感知器)



Input signals: $x = (x_1, \dots, x_n)$
Output signal: y
Parameters of the neuron model:
$w = (w_1, \dots, w_n)$ and b

Computational process:

$$z = w_1 x_1 + \cdots + w_n x_n + b$$
$$y = f(z)$$

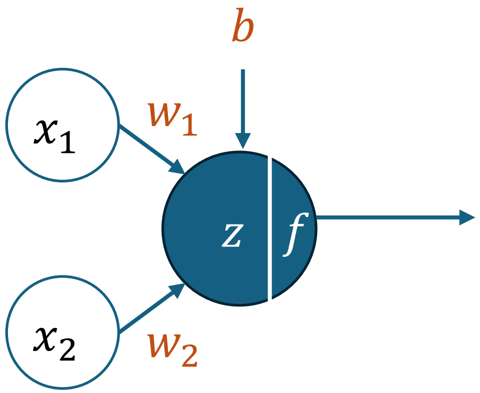
Reference:
https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc

Searching parameters
为神经元寻找合适的参数

- Given a set of inputs with their corresponding "desired" outputs, finding the value of parameters w and b, so that the behavior of the perceptron model is aligned with the given data.

Desired behavior of the neuron

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{d}$
1	1	-1
1	2	-1
2	1	-1
2	2	1



$$z = w_1x_1 + w_2x_2 + b$$

$$f(x) = \begin{cases} 1, x \geq 0 \\ -1, otherwise \end{cases}$$

$$y = f(z)$$

Finding parameters w_1, w_2, b so that the above perceptron model behaves as in the left table

Solution: transforming it into an **optimization problem**

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{d}$	Model output $\boldsymbol{y}$
1	1	-1	$f(w_1 + w_2 + b)$
1	2	-1	$f(w_1 + 2w_2 + b)$
2	1	-1	$f(2w_1 + w_2 + b)$
2	2	1	$f(2w_1 + 2w_2 + b)$

Loss function $L = \sum_i (d_i - y_i)^2$
We should find w_1, w_2, b that minimize L ! (ideally 0)

Optimization via gradient descent 使用梯度下降进行优化

	Gradient Descent	Linear Programming
Objective	Any differentiable function	Linear function
Constraint	N/A	Linear constraints

Gradient descent is another optimization technique to maximize/minimize a given function, which run two steps iteratively: ① compute gradient ② update variables guided by gradient

$y = f(x)$	$\frac{dy}{dx} = f'(x)$
k, any constant	0
x	1
x^2	$2x$
x^3	$3x^2$
x^n, any constant n	nx^{n-1}
e^x	e^x
e^{kx}	ke^{kx}
$\ln x = \log_e x$	$\frac{1}{x}$
$\sin x$	$\cos x$

Example: finding x and y that minimize $L = x^2 + y^2 + 2x - 2y$

Preparation: compute the gradient of L w.r.t. x and y

$\frac{\partial L}{\partial x} = 2x + 2$ (regard y as a constant), $\frac{\partial L}{\partial y} = 2y - 2$ (regard x as a constant)

Then initialize (x, y) randomly, and do the following iteratively:

① compute $g = \frac{\partial L}{\partial x}$ and $h = \frac{\partial L}{\partial y}$ (here they are $g = 2x + 2$ and $h = 2y - 2$)

② update x, y via $x \leftarrow x - \alpha g$ and $y \leftarrow y - \alpha h$ (α is a small learning rate)

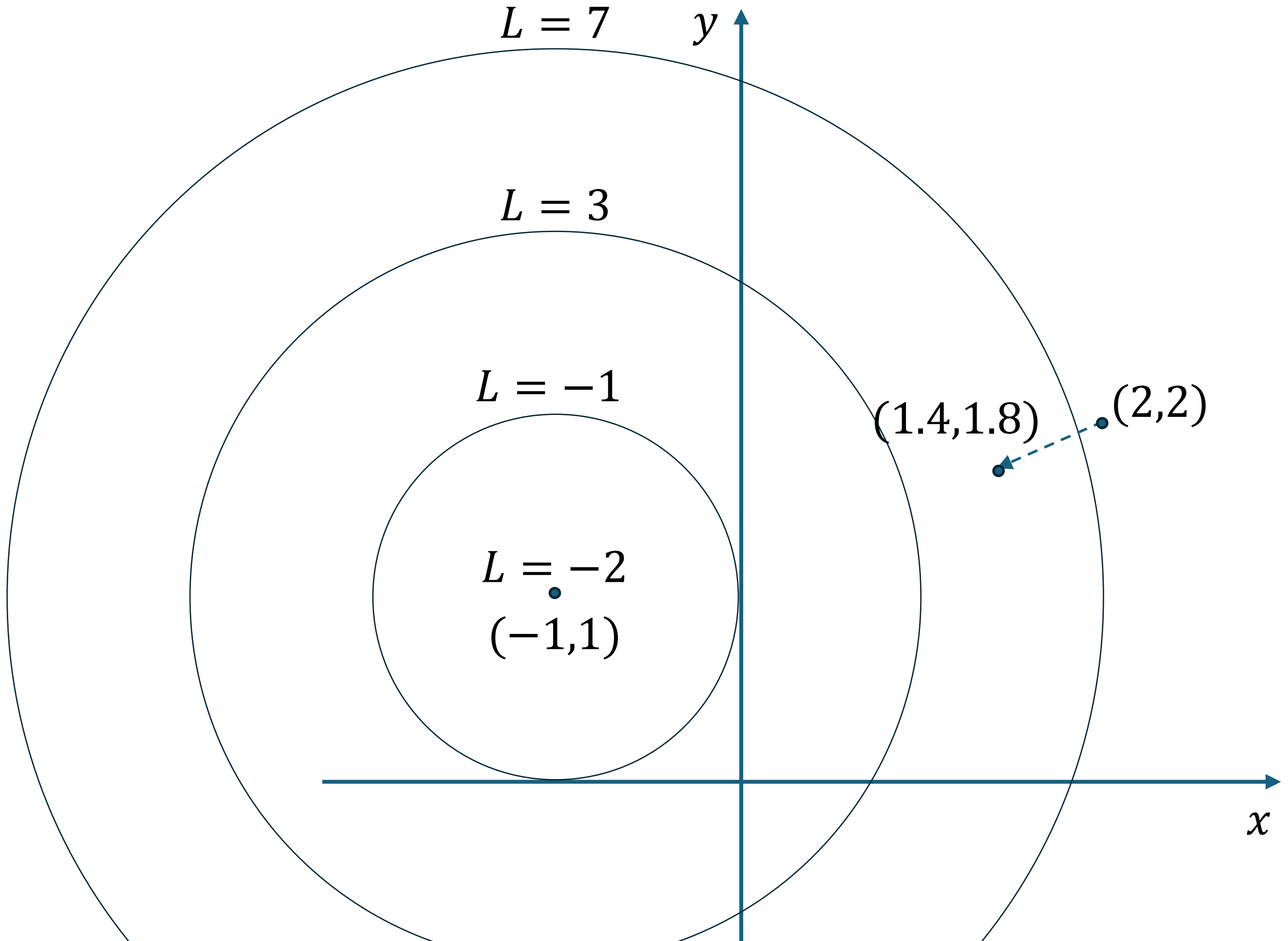
Until g and h are close to zero.

$\alpha = 0.1$

	x	y	$g = \frac{\partial L}{\partial x}$	$h = \frac{\partial L}{\partial y}$	L
0	2	2	6	2	8
1	1.4	1.8	5.6	1.6	4.4
	…	…	…	…	…
T	-1	1	0	0	-2

L is decreasing!

Optimization via gradient descent 使用梯度下降进行优化



$L = x^2 + y^2 + 2x - 2y$

$(-\frac{\partial L}{\partial x}, -\frac{\partial L}{\partial y})$ points to the direction that leads to fastest descent

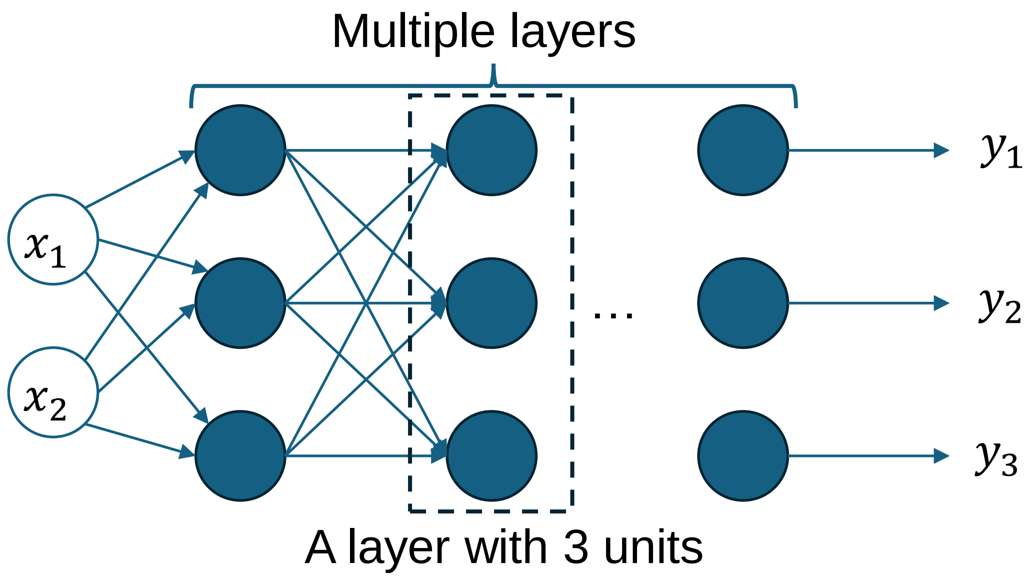
$\alpha = 0.1$

	x	y	$g = \frac{\partial L}{\partial x}$	$h = \frac{\partial L}{\partial y}$	L
0	2	2	6	2	8
1	1.4	1.8	5.6	1.6	4.4
	…	…	…	…	…
T	-1	1	0	0	-2

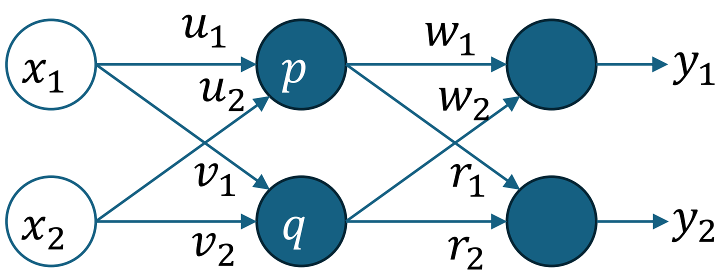
L is decreasing!

Multilayer Neural Networks
多层神经网络

- Now we can find suitable parameters for a single neuron model, to mimic given expected behaviors.
- However, the capacity of a single neuron is very limited
  - (consider the XOR logic function, why a single neuron model cannot mimic it?)
- Solution: stack multiple neuron models horizontally and vertically!



Backpropagation
反向传播



Here we omitted the bias term for simplicity

$$\frac{\partial(a+b)}{\partial x} = \frac{\partial a}{\partial x} + \frac{\partial b}{\partial x}$$
$$\frac{\partial f(y)}{\partial x} = \frac{\partial f(y)}{\partial y}\frac{\partial y}{\partial x}$$
$$\frac{\partial f(a,b)}{\partial x} = \frac{\partial f(a,b)}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial f(a,b)}{\partial b}\frac{\partial b}{\partial x}$$

- Feedforward:
  - $p = f(u_1x_1 + u_2x_2), q = f(v_1x_1 + v_2x_2)$
  - $y_1 = f(w_1p + w_2q), y_2 = f(r_1p + r_2q)$
  - $L_1 = (y_1 - d_1)^2, L_2 = (y_2 - d_2)^2$
  - $L = L_1 + L_2$

$$L = \underbrace{(\overbrace{f(w_1p + w_2q)}^{y_1} - d_1)^2}_{L_1} + \underbrace{(\overbrace{f(r_1p + r_2q)}^{y_2} \; - d_2)^2}_{L_2}$$

- Backpropagation (finding the gradient of loss function L w.r.t variables w, r, u, v)
  - $\frac{\partial L}{\partial y_1} = 2(y_1 - d_1), \frac{\partial L}{\partial y_2} = 2(y_2 - d_2)$
  - $\frac{\partial L}{\partial w_1} = \frac{\partial L_1}{\partial w_1} + 0 = \frac{\partial L}{\partial y_1}\frac{\partial y}{\partial w_1} = \frac{\partial L}{\partial y_1} f'(w_1p + w_2q)p, \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_2}\frac{\partial y}{\partial w_2} = \frac{\partial L}{\partial y_2} f'(r_1p + r_2q)q$ (similar for $\frac{\partial L}{\partial r_1}$ and $\frac{\partial L}{\partial r_2}$)
  - $\frac{\partial L}{\partial p} = \frac{\partial L_1}{\partial p} + \frac{\partial L_2}{\partial p} = \frac{\partial L_1}{\partial y_1}\frac{\partial y_1}{p} + \frac{\partial L_2}{\partial y_2}\frac{\partial y_2}{p} = \frac{\partial L}{\partial y_1} f'(w_1p + w_2q)w_1 + \frac{\partial L}{\partial y_2} f'(r_1p + r_2q)r_1$ (similar for $\frac{\partial L}{\partial q}$)
  - $\frac{\partial L}{\partial u_1} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial u_1} = \frac{\partial L}{\partial p} f'(u_1x_1 + u_2x_2)x_1, \frac{\partial L}{\partial u_2} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial u_2} = \frac{\partial L}{\partial p} f'(u_1x_1 + u_2x_2)x_2$ (similar for $\frac{\partial L}{\partial v_1}$ and $\frac{\partial L}{\partial v_2}$)

Feedforward network in matrix form
反向传播的矩阵形式

Here we omitted the bias term for simplicity

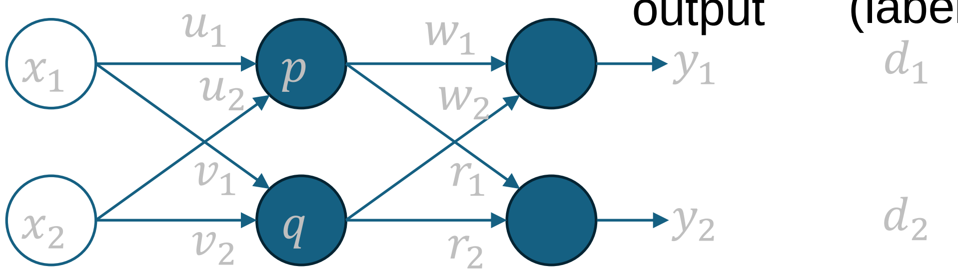
$$V = \begin{pmatrix} u_1 & u_2 \\ v_1 & v_2 \end{pmatrix} \quad W = \begin{pmatrix} w_1 & w_2 \\ r_1 & r_2 \end{pmatrix}$$

Model output

Expected Output (label)

inputs

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$



$$z = \begin{pmatrix} p \\ q \end{pmatrix} = f(Vx) \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = f(Wz)$$

In such a way we can simply write the feedforward process as

$$y = f\big(Wf(Vx)\big)$$

with parameters W and V

Training a feedforward neural network:

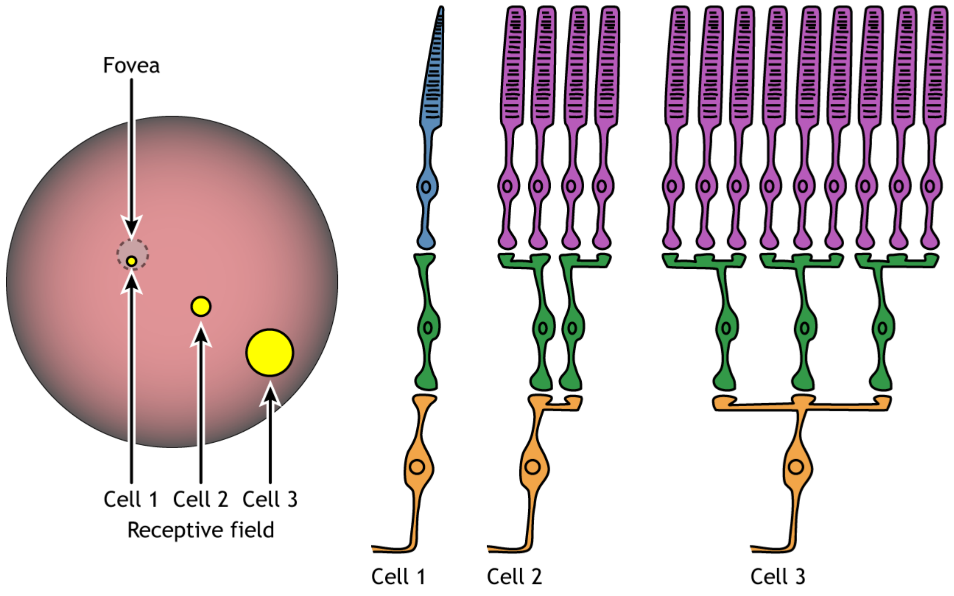
```
Given dataset (X,D), initialize parameters W,V
While not converged:
  sample data x,d from (X,D)
  compute model output y = f(Wf(Vx))
  compute loss function L = ||y − d||^2
  compute gradients ∂L/∂W, ∂L/∂V via backpropagation
  update parameters via gradient descent
    W ← W − α ∂L/∂W,  V ← V − α ∂L/∂V
```

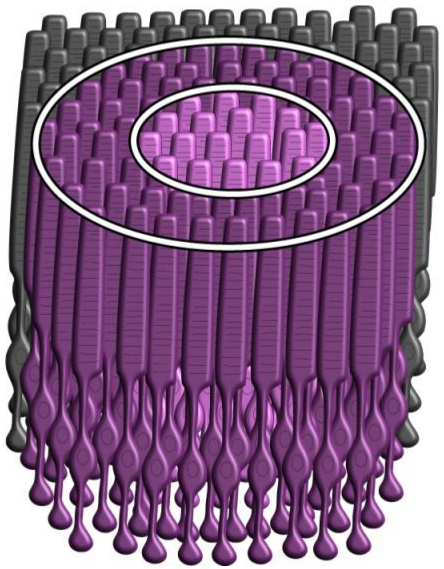
Basic Neural networks for image and text 面向图像和文本的基础模型

- Convolutional Neural Networks (CNN) – spatial connection 卷积神经网络
- Recurrent Neural Networks (RNN) – temporal connection 循环神经网络

Receptive field 感受野

- Different from the fully-connected case, neurons in the retina （视网膜） respond to light stimulus in **restricted regions** of the visual field

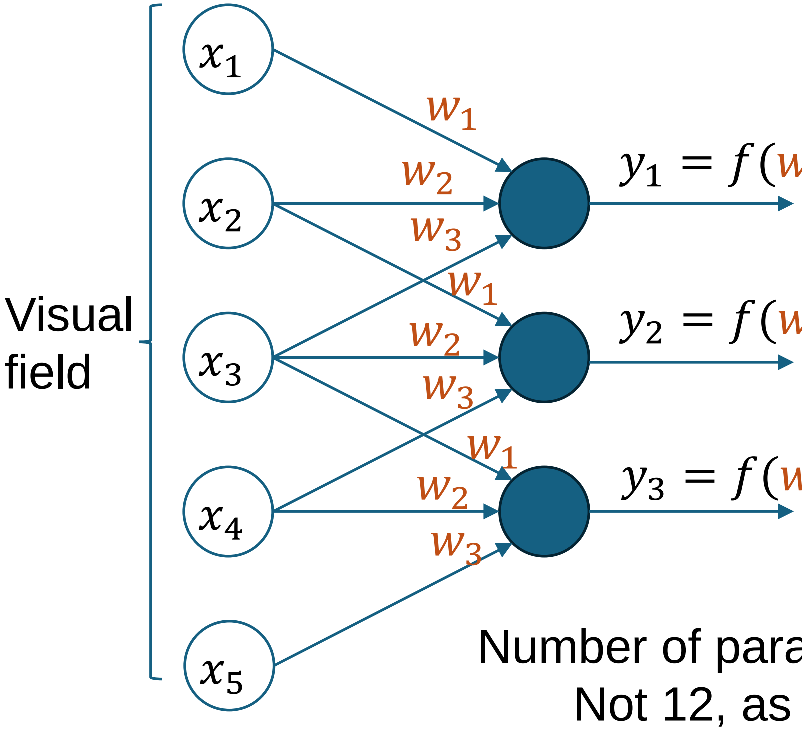




Reference: https://openbooks.lib.msu.edu/neuroscience/chapter/vision-the-retina/

Convolutional layer (1D)
一维卷积层

- To mimic the characteristic of retina neurons, we design a special way of connection that is
  - **Sparsely, local connected**: each output only connects to its nearest k inputs
  - **Shared weight**: the weight is replicated across the entire visual field
- We named it as a “filter”



$y_1 = f(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$

$y_2 = f(w_1 x_2 + w_2 x_3 + w_3 x_4 + b)$

$y_3 = f(w_1 x_3 + w_2 x_4 + w_3 x_5 + b)$

Number of parameters: 4 (w_1, w_2, w_3, b)

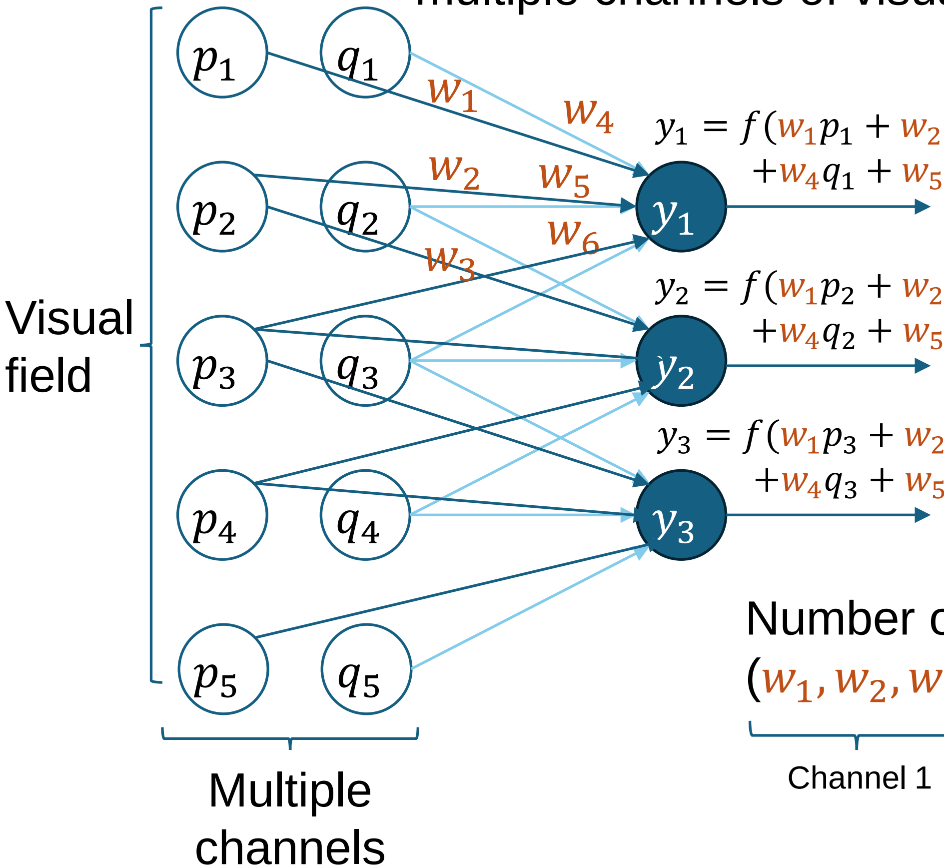
Not 12, as the weights are shared across the visual field

Size of the receptive field: 3

Each output connects to 3 inputs, not 5

Convolutional layer (1D)
一维卷积层

① One filter can process multiple channels of visual field



$y_1 = f(w_1 p_1 + w_2 p_2 + w_3 p_3 + w_4 q_1 + w_5 q_2 + w_6 q_3 + b)$

$y_2 = f(w_1 p_2 + w_2 p_3 + w_3 p_4 + w_4 q_2 + w_5 q_3 + w_6 q_4 + b)$

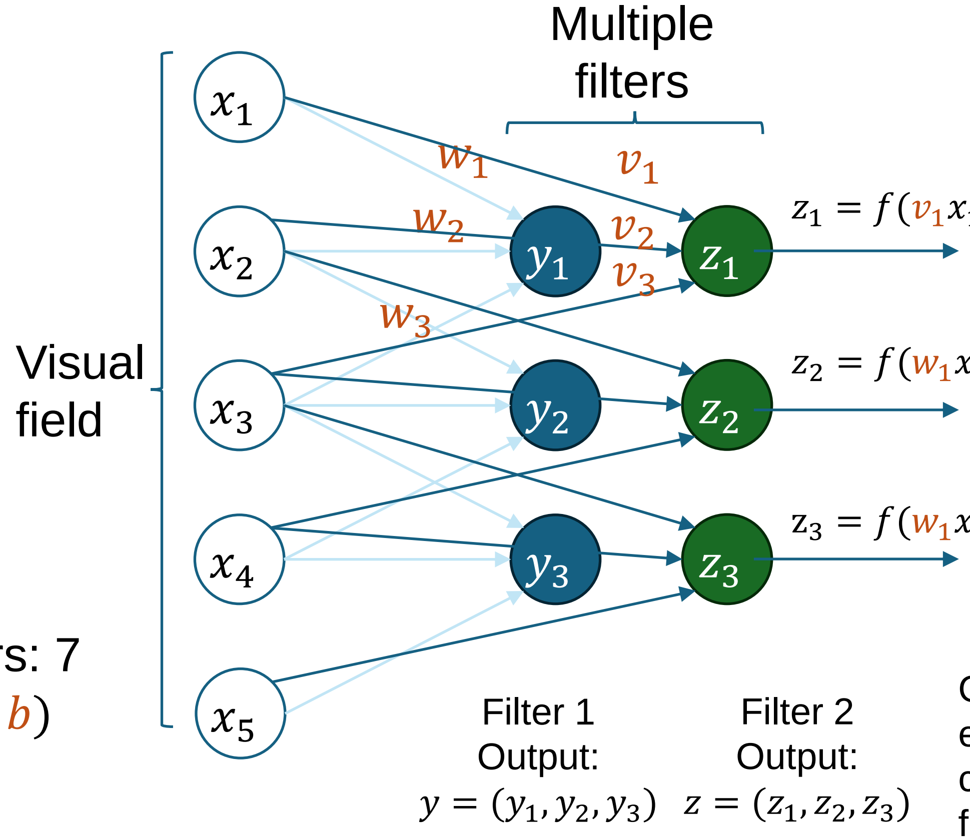
$y_3 = f(w_1 p_3 + w_2 p_4 + w_3 p_5 + w_4 q_3 + w_5 q_4 + w_6 q_5 + b)$

Number of parameters: 7

$(\underbrace{w_1, w_2, w_3}_{\text{Channel 1}}, \underbrace{w_4, w_5, w_6, b}_{\text{Channel 2}})$

② Multiple filters can work simultaneously on the same visual field

A convolutional layer usually consists of multiple filters



$z_1 = f(v_1 x_1 + v_2 x_2 + v_3 x_3 + b)$

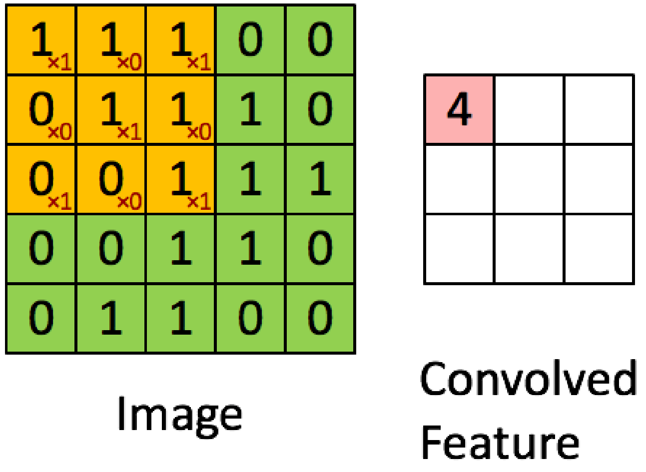
$z_2 = f(w_1 x_2 + w_2 x_3 + w_3 x_4 + b)$

$z_3 = f(w_1 x_3 + w_2 x_4 + w_3 x_5 + b)$

Output: 2 channels, each channel consists of a visual field of length 3

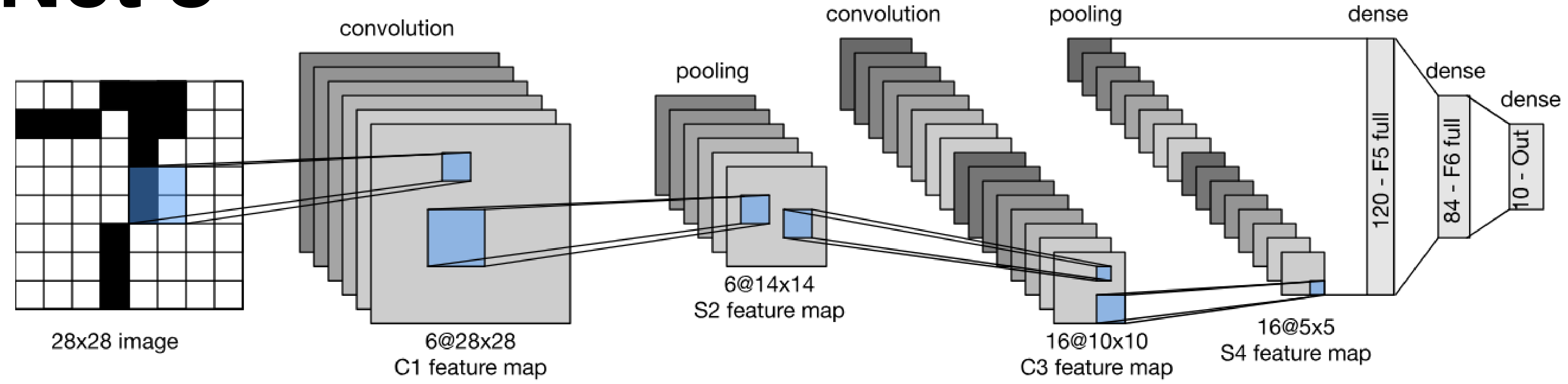
Convolutional layer (2D)
二维卷积层

- A direct extension of the previous discussed filter, from 1D to 2D visual fields.
  - Input: from an 1D vector (size 5) to a 2D matrix (size 5×5)
  - Output: from an 1D vector (size 3) to a 2D matrix (size 3×3)
  - Receptive field: from an 1D sub-range (size 3) to a 2D sub-range (size 3×3)
- Other things are generally the same!



Visual field = feature map
Size of receptive field = kernel size

LeNet 5

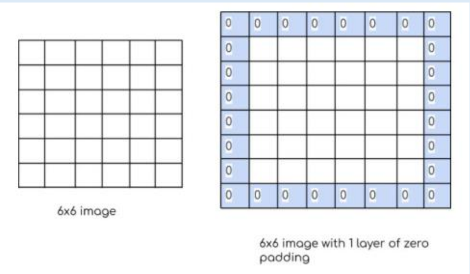


Receptive Field: 5×5
Number of filters: 6
Number of parameters:
$(5 \times 5 \times 1) \times 6$
With 2 paddings

Receptive Field: 5×5
Number of filters: 16
Number of parameters:
$(5 \times 5 \times 6) \times 16$
No padding

Reference:
https://d2l.ai/chapter_convolutional-neural-networks/lenet.html
Number of parameters excludes the bias term.

Padding 填充
The size of the visual field will “shrink” after convolution
To recover the size, we add padding at the border of the visual field.



Pooling 池化
A pooling layer slides a two-dimensional filter over each channel of visual field, and summarizes the value lying within the region covered by the filter.

1	2	2	3
2	1	3	2
2	3	1	2
3	2	2	1

1.5	2.5
2.5	1.5

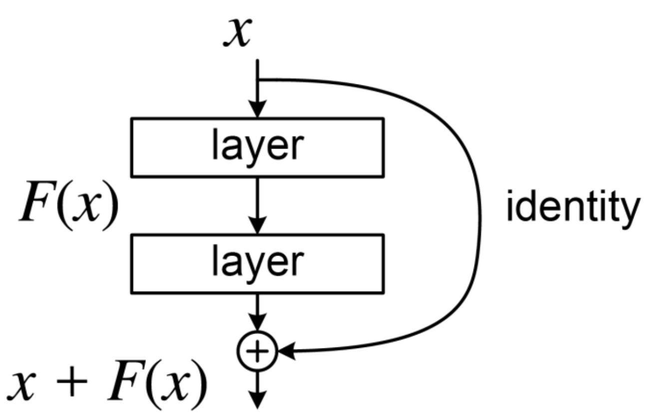
Average pooling

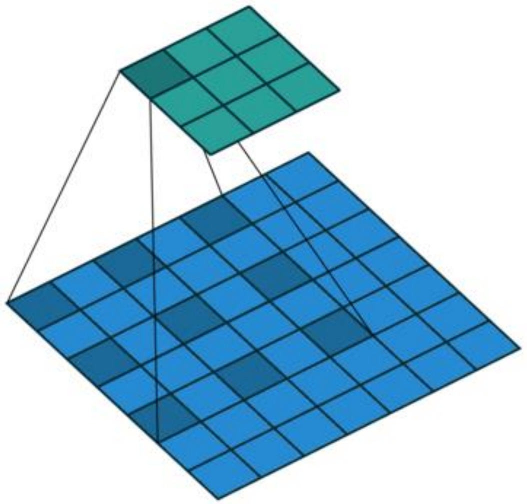
2	3
3	2

Max pooling

More about CNN

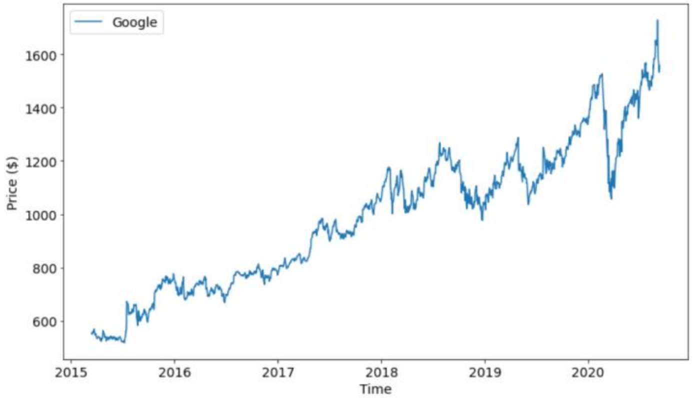
- Modern CNN (e.g., ResNet): https://d2l.ai/chapter_convolutional-modern/index.html
- Different types of convolution https://github.com/vdumoulin/conv_arithmetic

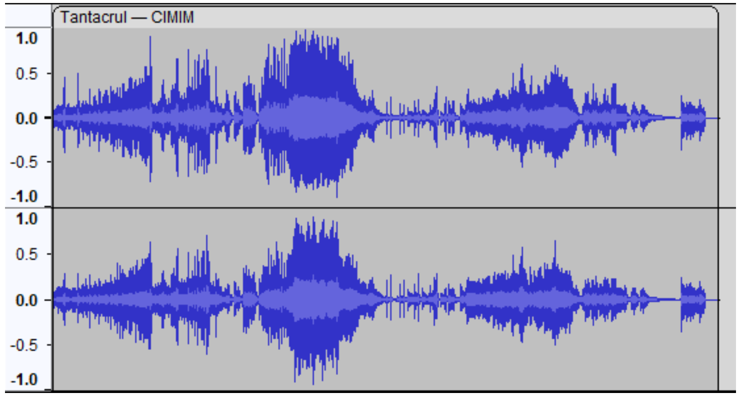


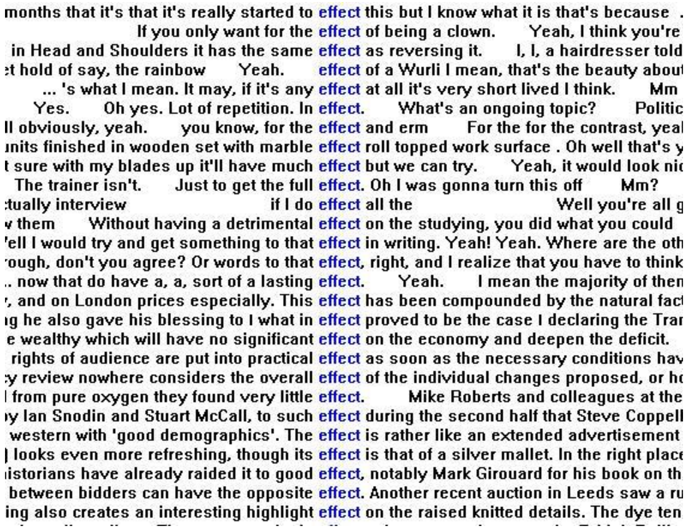


Sequential Data with temporal connections 时间序列数据

- Time-series data (stock price)
- Audio
- …
- And the most common one, **text （文本）**



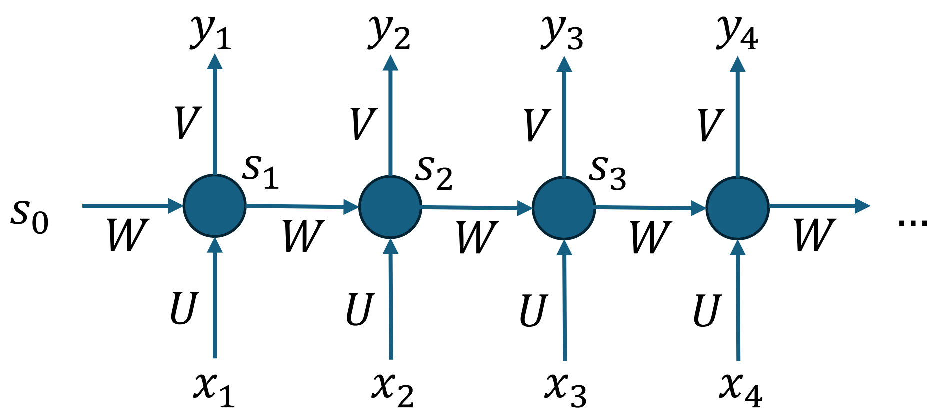



months that it's that it's really started to effect this but I know what it is that's because .
If you only want for the effect of being a clown. Yeah, I think you're
in Head and Shoulders it has the same effect as reversing it. I, I, a hairdresser told
:t hold of say, the rainbow Yeah. effect of a Wurli I mean, that's the beauty about
... 's what I mean. It may, if it's any effect at all it's very short lived I think. Mm
Yes. Oh yes. Lot of repetition. In effect. What's an ongoing topic? Politic
ll obviously, yeah. you know, for the effect and erm For the for the contrast, yeal
ınits finished in wooden set with marble effect roll topped work surface . Oh well that's y
t sure with my blades up it'll have much effect but we can try. Yeah, it would look nic
The trainer isn't. Just to get the full effect. Oh I was gonna turn this off Mm?
:tually interview if I do effect all the Well you're all g
v them Without having a detrimental effect on the studying, you did what you could
'ell I would try and get something to that effect in writing. Yeah! Yeah. Where are the oth
ough, don't you agree? Or words to that effect, right, and I realize that you have to think
.. now that do have a, a, sort of a lasting effect. Yeah. I mean the majority of then
, and on London prices especially. This effect has been compounded by the natural fact
ıg he also gave his blessing to I what in effect proved to be the case I declaring the Trar
e wealthy which will have no significant effect on the economy and deepen the deficit.
rights of audience are put into practical effect as soon as the necessary conditions hav
:y review nowhere considers the overall effect of the individual changes proposed, or hc
l from pure oxygen they found very little effect. Mike Roberts and colleagues at the
ıy Ian Snodin and Stuart McCall, to such effect during the second half that Steve Coppell
western with 'good demographics'. The effect is rather like an extended advertisement
] looks even more refreshing, though its effect is that of a silver mallet. In the right place
ıistorians have already raided it to good effect, notably Mark Girouard for his book on th
between bidders can have the opposite effect. Another recent auction in Leeds saw a ru
ing also creates an interesting highlight effect on the raised knitted details. The dye ten

Process sequential data with a recurrent neural network 使用循环神经网络处理序列数据

- Assuming that the data is represented as $x_1, x_2, \dots, x_T$ (each x_t is an n-dimensional vector)
- Initialize a state vector s of length h, and three parameters U, W, V (in matrix form)
- For t from 1 to T:
  - Update state: $s_t \leftarrow f(Ux_t + Ws_{t-1})$
  - Produce output: $y_t \leftarrow Vs_t$

Reference: https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-1/



U: an $h \times n$ matrix transforming input x_t
W: an $h \times h$ matrix transforming previous input s_{t-1}
V: an $n \times h$ matrix transforming current input s_t
So we have $(2n + h)h$ parameters in an RNN (excluding bias)

如何在神经网络中表示一个单词?

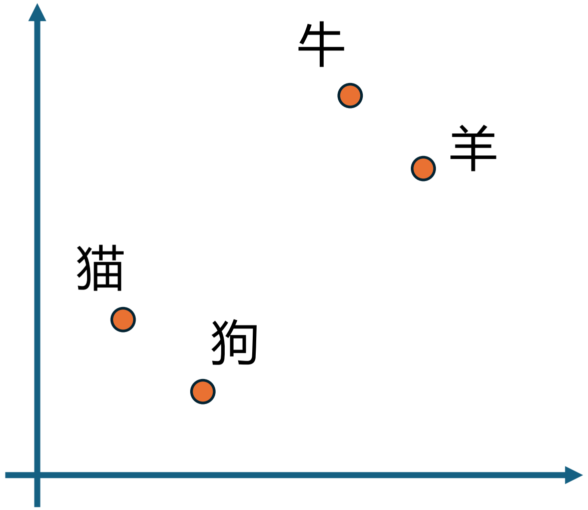
- 计算机内的文字编码（UTF8，Unicode等）
- 猫：\u732b；狗：\u72d7；牛：\u725b；羊：\u7f8a
- 整数编码（取一本词典，给其中的每个词编个号）
- 猫：1；狗：2；牛：3；羊：4

One-Hot编码

猫	1	0	0	0
狗	0	1	0	0
牛	0	0	1	0
羊	0	0	0	1

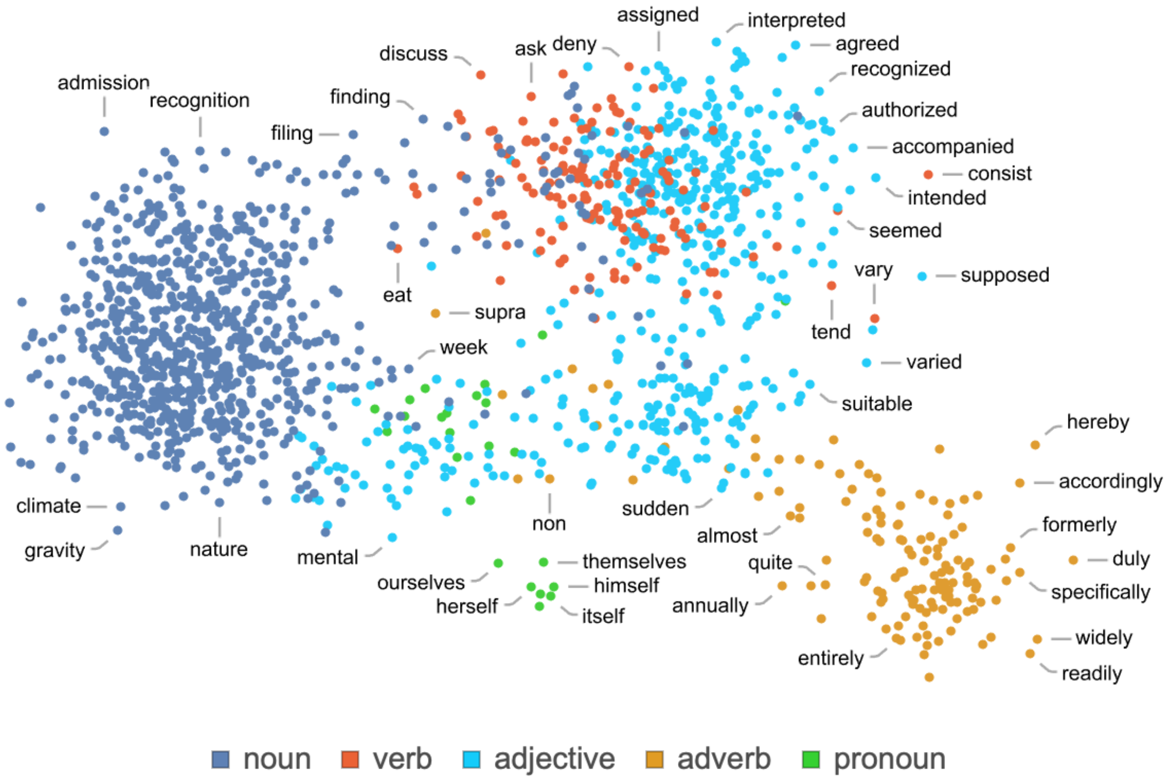
词嵌入（Word Embedding）

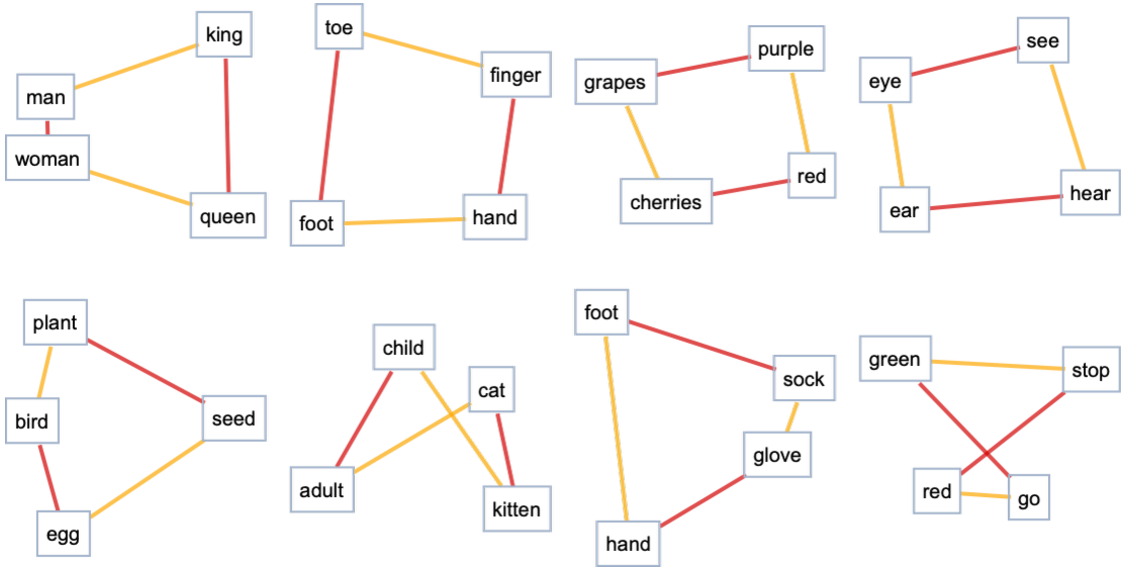
猫	0.1	0.2
狗	0.2	0.1
牛	0.8	0.7
羊	0.7	0.8



“语义空间”：语意相似的词在向量空间上也会比较相近

Word Embedding 词嵌入



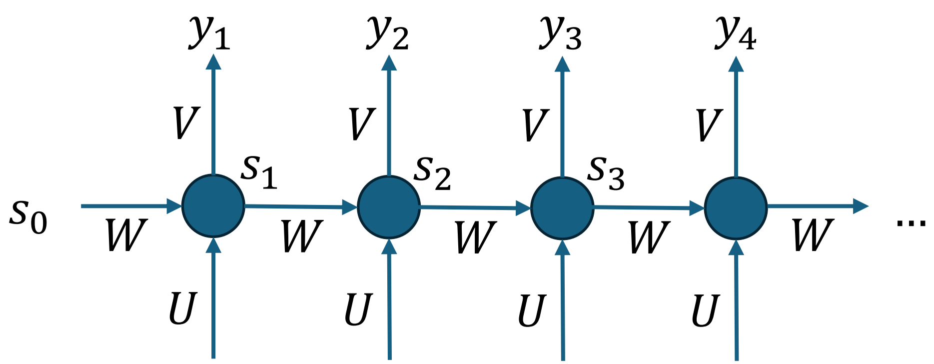


Reference：
https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

RNN Example: next word prediction
卷积神经网络示例：预测下一个单词

Input $x_1, x_2, \dots, x_T$	Expected label d_T
• An apple a day keeps ___	the
• An apple a day keeps the ___	doctor
• An apple a day keeps the doctor ___	away

Expected label: d_1 apple, d_2 a, d_3 day, d_4 keeps



Input: x_1 an, x_2 apple, x_3 a, x_4 day

Training a recurrent neural network:

Given dataset (X, D), initialize parameters W, V
While not converged:
  sample data $x_1, x_2, \dots, x_T, d$ from (X, D)
  For t from 1 to T:
    $s_t = f(Ux_t + Ws_{t-1})$, $y_t = Vs_t$
  compute loss function $L = \sum_t \|y_t - d_t\|^2$
  compute gradients $\frac{\partial L}{\partial U}, \frac{\partial L}{\partial W}, \frac{\partial L}{\partial V}$ via backpropagation
  update parameters via gradient descent
    $U \leftarrow U - \alpha \frac{\partial L}{\partial U}$, $W \leftarrow W - \alpha \frac{\partial L}{\partial W}$, $V \leftarrow V - \alpha \frac{\partial L}{\partial V}$

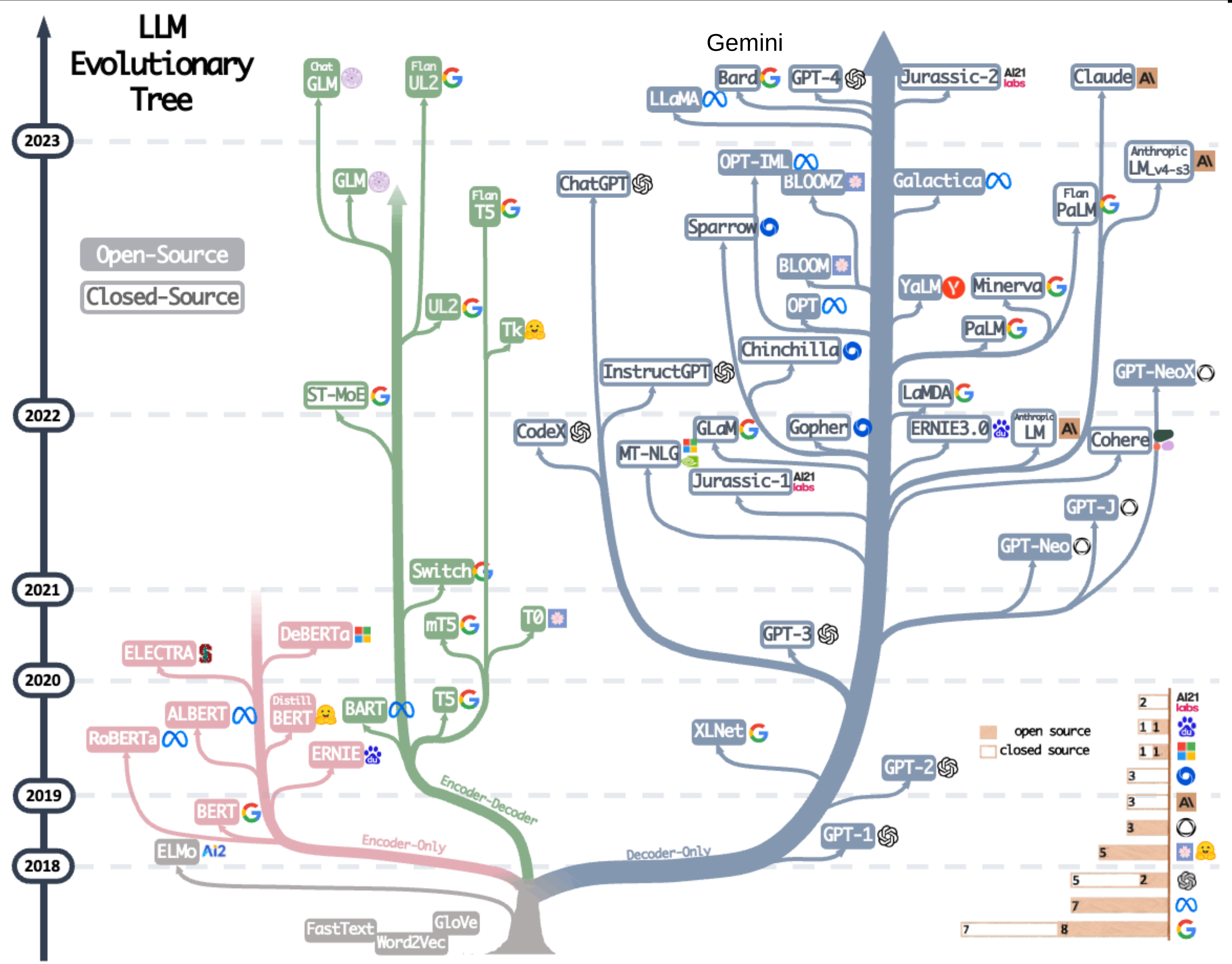
More about RNN

- Backpropagation Through Time https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-3/
- Vanishing Gradients and LSTM https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Sequence-to-Sequence Model (Seq2Seq) https://www.tensorflow.org/text/tutorials/nmt_with_attention

State of the art techniques 最佳实践

- Transformer (the technique behind Gemini)

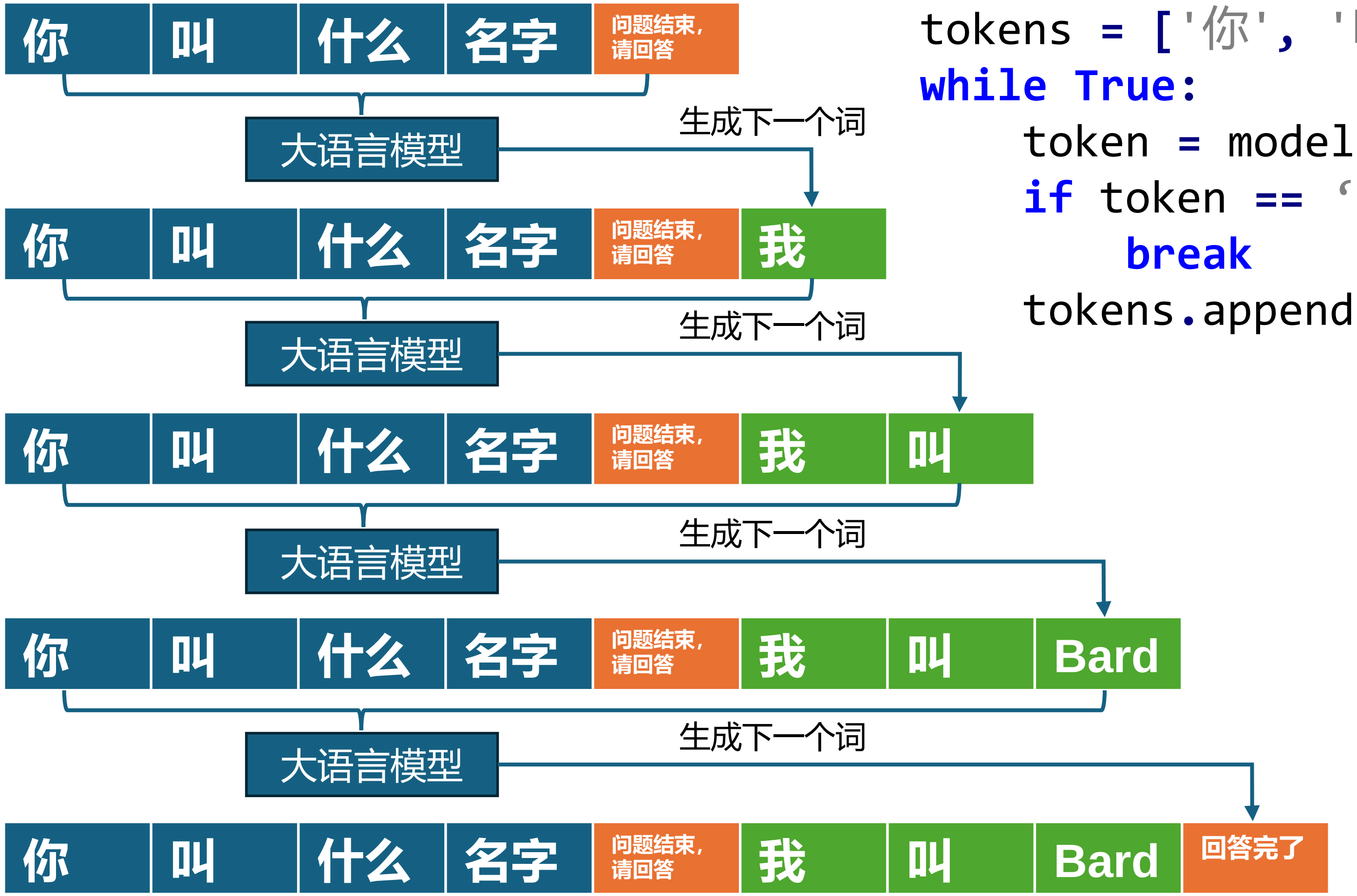
You’ve



Reference:
https://arxiv.org/abs/2304.13712

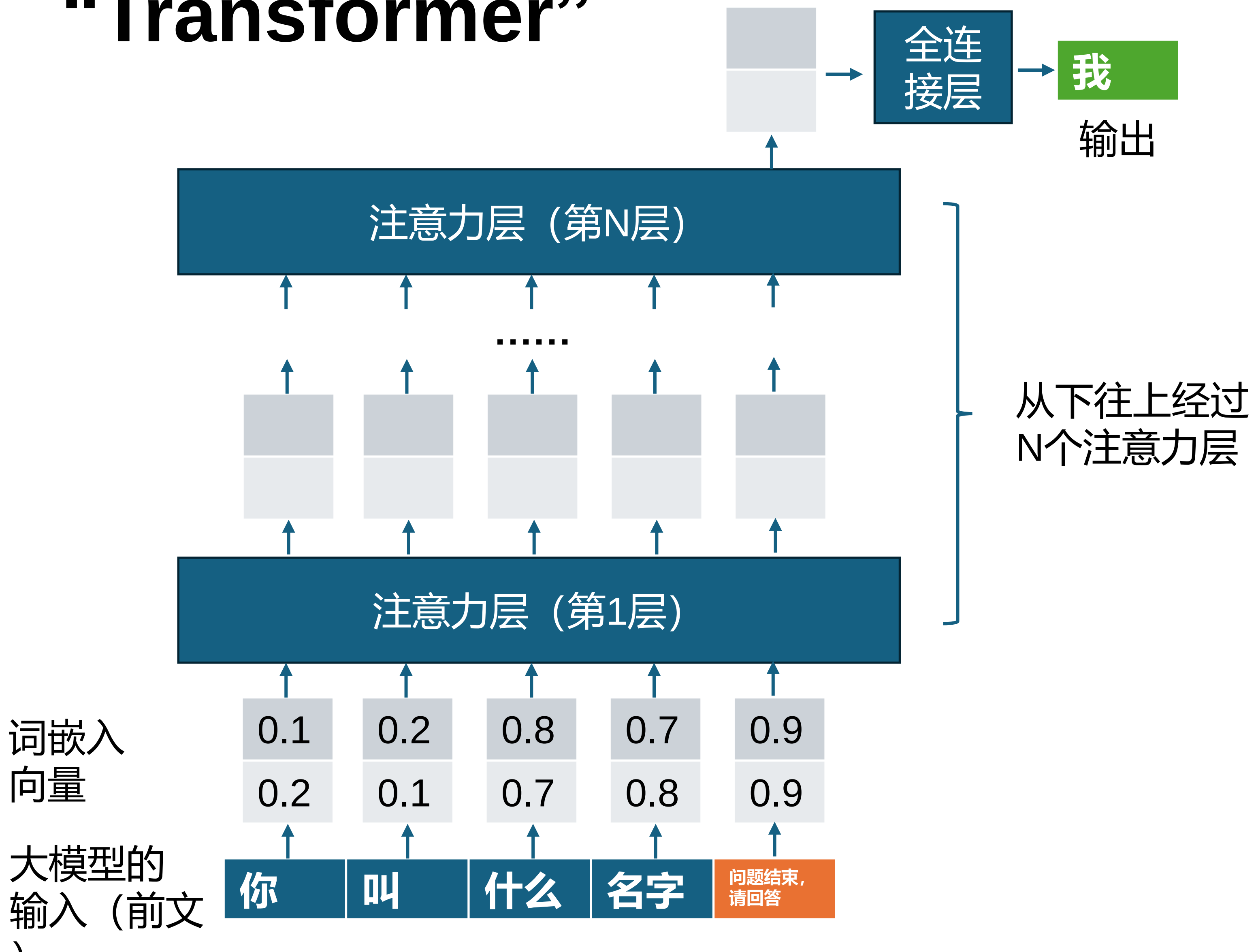
Reference: https://weibo.com/1727858283/NcKwECOwi

大语言模型（表面上）在做的事情：根据前文不断地生成下一个词




```
tokens = ['你', '叫', '什么', '名字', '问题结束，请回答']
while True:                                    # 不断循环生成下一个词
    token = model.generate(tokens)             # 根据前文生成下一个词
    if token == ‘回答完了’ :                    # 如果回答完毕
        break                                  # 则退出生成过程
    tokens.append(token)                       # 将当前生成的词加入前文
```

大语言模型的结构
“Transformer”



“Attention Is All You Need”
注意力就是你所需的一切

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com
Noam Shazeer[*] Google Brain noam@google.com
Niki Parmar[*] Google Research nikip@google.com
Jakob Uszkoreit[*] Google Research usz@google.com
Llion Jones[*] Google Research llion@google.com
Aidan N. Gomez[* †] University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com
Illia Polosukhin[* ‡] illia.polosukhin@gmail.com

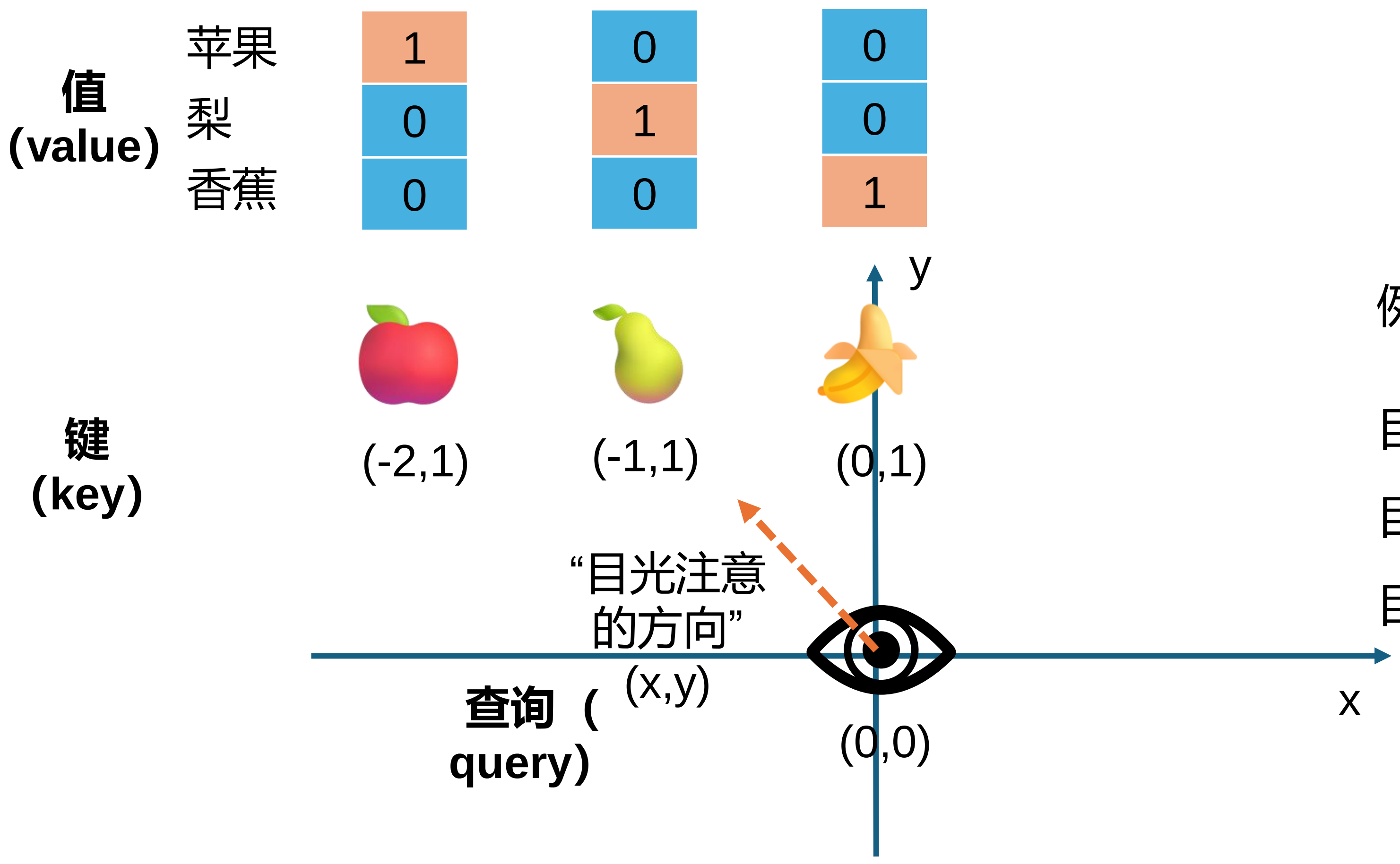
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

https://arxiv.org/abs/1706.0376

大语言模型的结构：注意力机制（Attention）

详细原理可见《简明的大语言模型原理介绍》：
https://snowkylin.github.io/talks



1. 计算“目光注意的方向”与物品方向的**匹配程度**（可以通过向量乘来实现）
2. 根据匹配程度，对每个物品对应的“嵌入向量”进行**加权求和**

例：当目光注意的方向为“梨”时

$$(x,y)=(-1,1)$$

目光方向与苹果的匹配程度=$\frac{(-2,1)\times(-1,1)}{|(-2,1)||(-1,1)|}\approx 0.949$

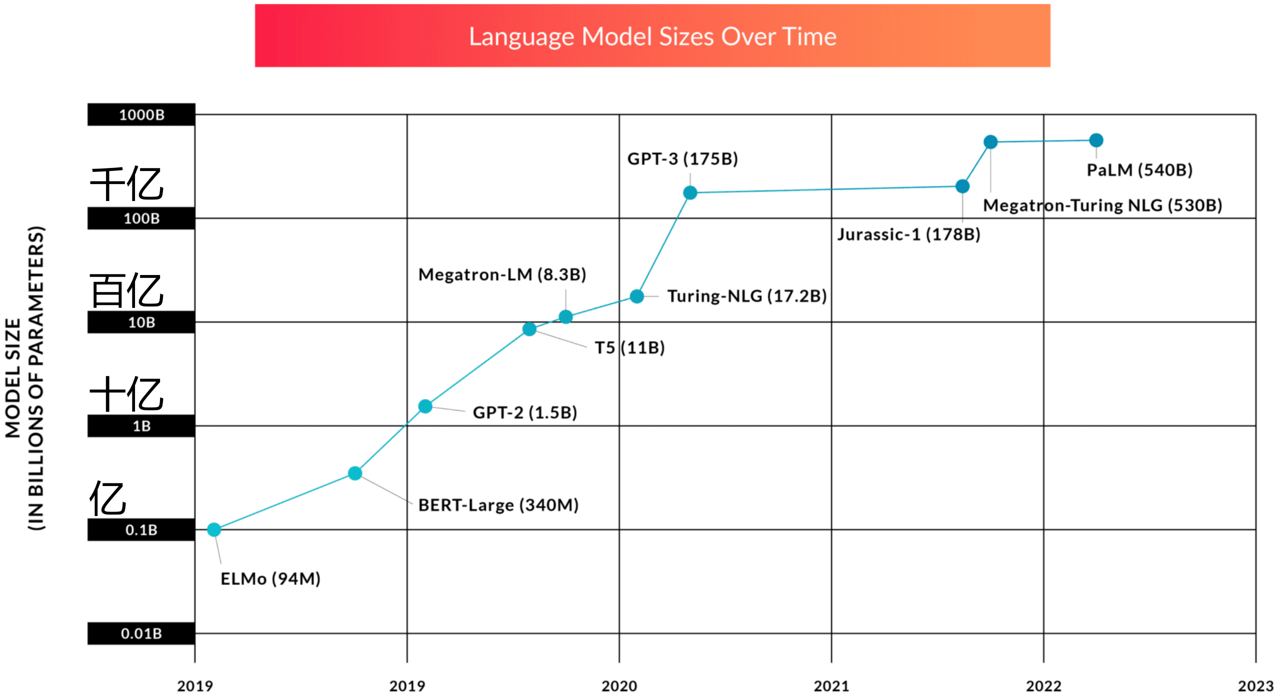
目光方向与梨的匹配程度=$\frac{(-1,1)\times(-1,1)}{|(-1,1)||(-1,1)|}=1$

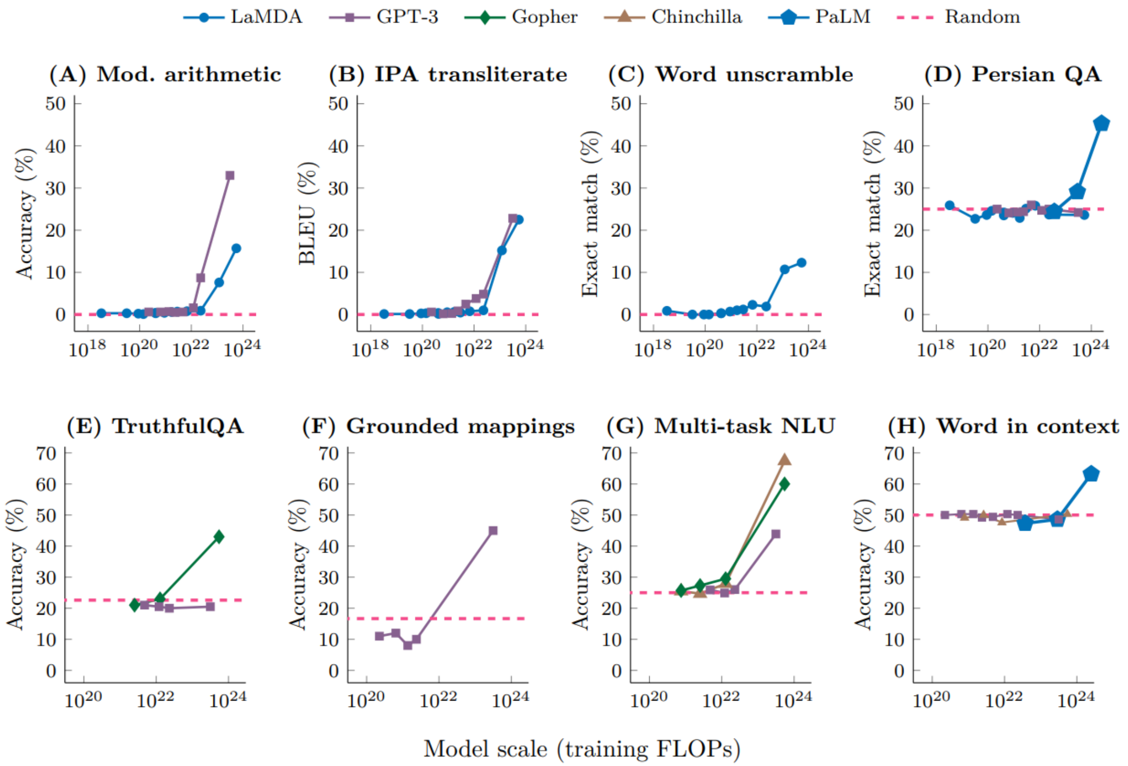
目光方向与香蕉的匹配程度=$\frac{(0,1)\times(-1,1)}{|(0,1)||(-1,1)|}\approx 0.707$

$$\begin{bmatrix}1\\0\\0\end{bmatrix}\times 0.949+\begin{bmatrix}0\\1\\0\end{bmatrix}\times 1+\begin{bmatrix}0\\0\\1\end{bmatrix}\times 0.707=\begin{bmatrix}0.949\\1\\0.707\end{bmatrix}$$

这几年的语言模型，发生了什么？

1. 随着硬件算力、数据、模型、训练方法的发展，模型越来越大（从“语言模型”到“大语言模型”）——人们发现，随着模型规模的增大，越过了某些“临界点”时，能够 “从量变到质变”，逐步**涌现**出新的能力（emergent abilities）





Reference: https://cmte.ieee.org/futuredirections/2023/04/24/how-much-bigger-can-should-llms-become/
Emergent Abilities of Large Language Models https://arxiv.org/abs/2206.07682

这几年的语言模型，发生了什么?

模型	规模	涌现的能力
BERT/GPT （2018）	12层Transformer 7000本书（4.6GB） 1.17亿参数	**预训练（Pre-training)** 为了完成某个特定任务（比如说机器翻译），我们可以不用专门从零开始训练一个语言模型，而是可以先用海量数据无监督地训练一个“预训练模型”，然后再使用较少的有监督数据对模型进行“微调”（fine-tuning） 降低了对训练数据的要求
GPT-2（2019）	模型架构相同（只是更大了） 训练数据扩展到40GB（Reddit高赞文章） 15亿参数	**多任务处理（Multi-task)** 即使完全不针对特定任务进行微调和参数更新，（在精心、人为的推理方式设计下）也能在很多自然语言任务中取得好的结果。
GPT-3（2020） Codex（2021） GPT-3.5（2022）	训练数据扩展到600GB 参数扩展到1750亿 Codex加入代码训练 GPT-3.5使用指令微调和基于人类反馈的强化学习	**语境“学习”（In-Context Learning)** 即使完全不对模型参数进行微调或更新，在给模型输入的上下文中直接用自然语言提供几个“示例”，模型也能完成任务。 “请输出鸡腿的个数：一只鸡=2条腿、两只鸡=4条腿，三只鸡=” **思维链（Chain of Thought, CoT)** “Let’s think step by step”，让模型输出自己的思考步骤

这几年的语言模型，发生了什么?

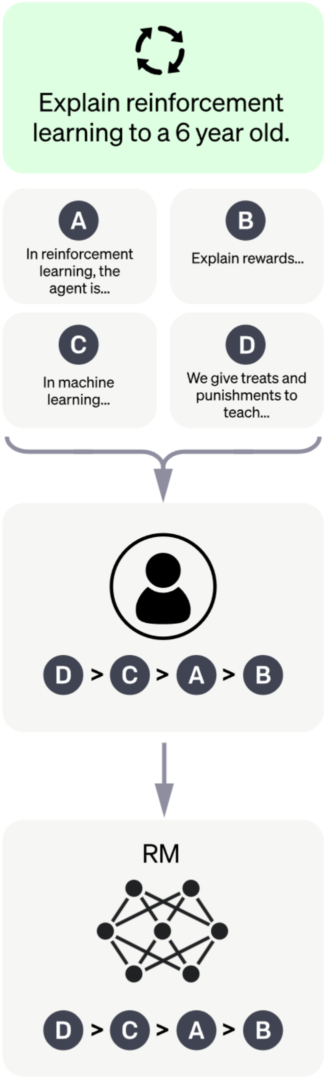
2. 一系列技术促进语言模型的输出符合人类期望

- 指令微调技术（instruction finetuning）
  - 先无监督预训练一个语言模型，然后使用有监督的“指令-回答”语料，在多种任务上进行训练
  - 数据获得太昂贵，对于开放性问题的效果不好（write a story about…）
- 基于人类反馈的强化学习（Reinforcement Learning from Human Feedback, RLHF）
  - 训练一个“奖励模型”，为模型生成的结果打分（右图）
  - 让人类对模型生成出的结果打分，或比较好坏
  - 使用强化学习，对训练好的语言模型进行微调，鼓励模型生成奖励高的回复，抑制模型生成奖励低的回复

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Reference: https://openai.com/blog/chatgpt

推荐阅读

概念入门

- 《这就是ChatGPT》，斯蒂芬·沃尔弗拉姆，人民邮电出版社，2023
- 关于 AI 的深度研究：ChatGPT 正在产生心智吗？ https://www.bilibili.com/video/BV1uu4y1m7ak

前沿进展

- Natural Language Processing with Deep Learning CS224N/Ling284 https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture11-prompting-rlhf.pdf

编程实现

- TensorFlow官方教程：Neural machine translation with a Transformer and Keras，https://www.tensorflow.org/text/tutorials/transformer
- The Illustrated GPT-2，http://jalammar.github.io/illustrated-gpt2/
- 中译 https://blog.csdn.net/g534441921/article/details/104312983

Thank you!
谢谢!

Xihan Li 李锡涵
Department of Computer Science, University College London
Google Developers Expert in Machine Learning
xihan.li@cs.ucl.ac.uk
https://snowkylin.github.io

Mar 2024