深度学习：零基础体验

PyTorch/Tensorflow的本质：大规模快速偏微分求导推理机器

TensorFlow

提纲

1. 求导方法
   - 什么是导数,微积分
   - 导数的传递(chain rule)
   - 手算例子
   - PyTorch/Tensorflow 计算图

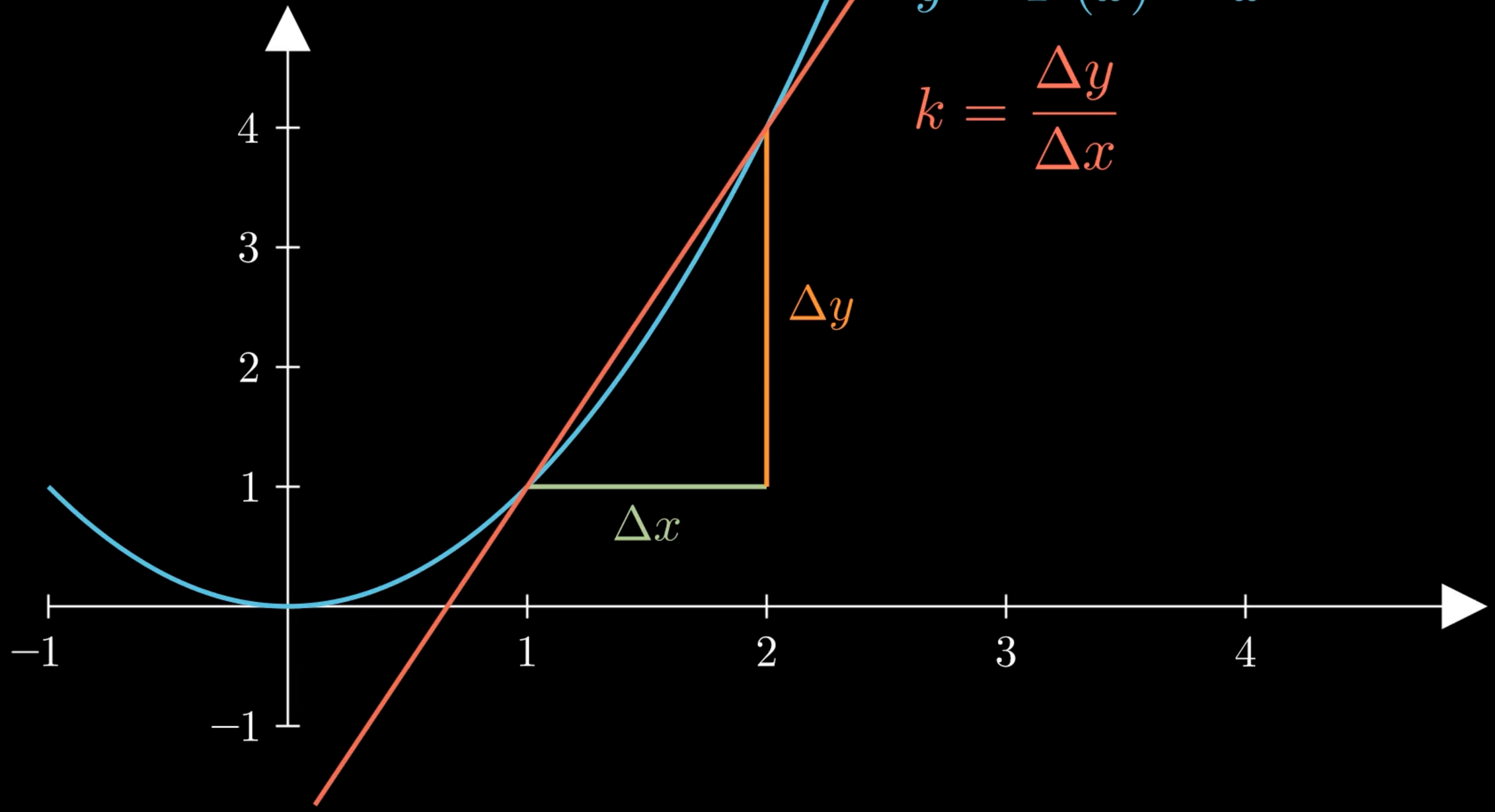
2. 深度学习的结构
   - 全联接网络（FCNN/ANN）
   - 非线性变换
     - 单位变换/概率映射/损失函数
   - 随机梯度下降（SGD）
   - 卷积神经网络CNN

3. 深度学习的目的：
   - 概率论视角
   - 循环神经网络RNN
   - 注意力机制(Attention mechanism)

导数的诞生

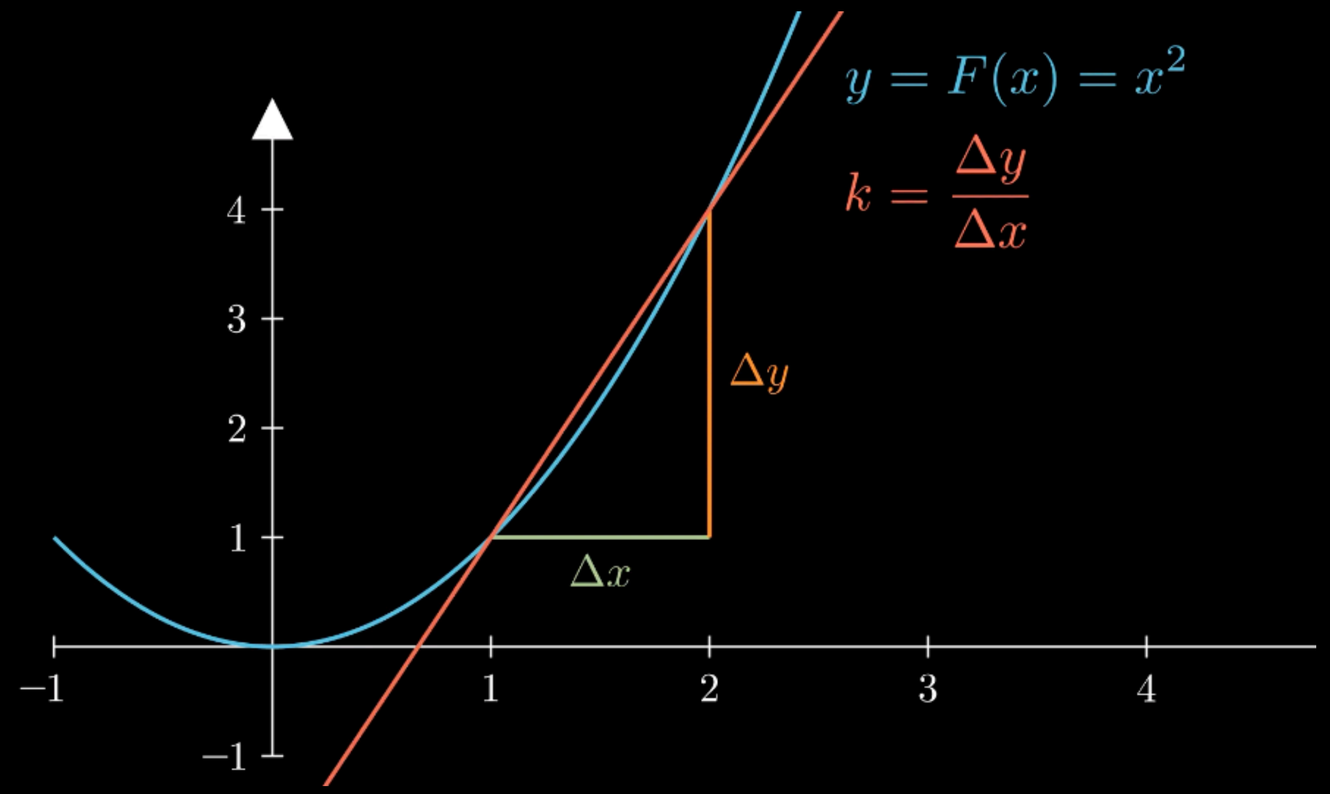
$$y = F(x) = x^2$$

$$k = \frac{\Delta y}{\Delta x}$$



导数的诞生

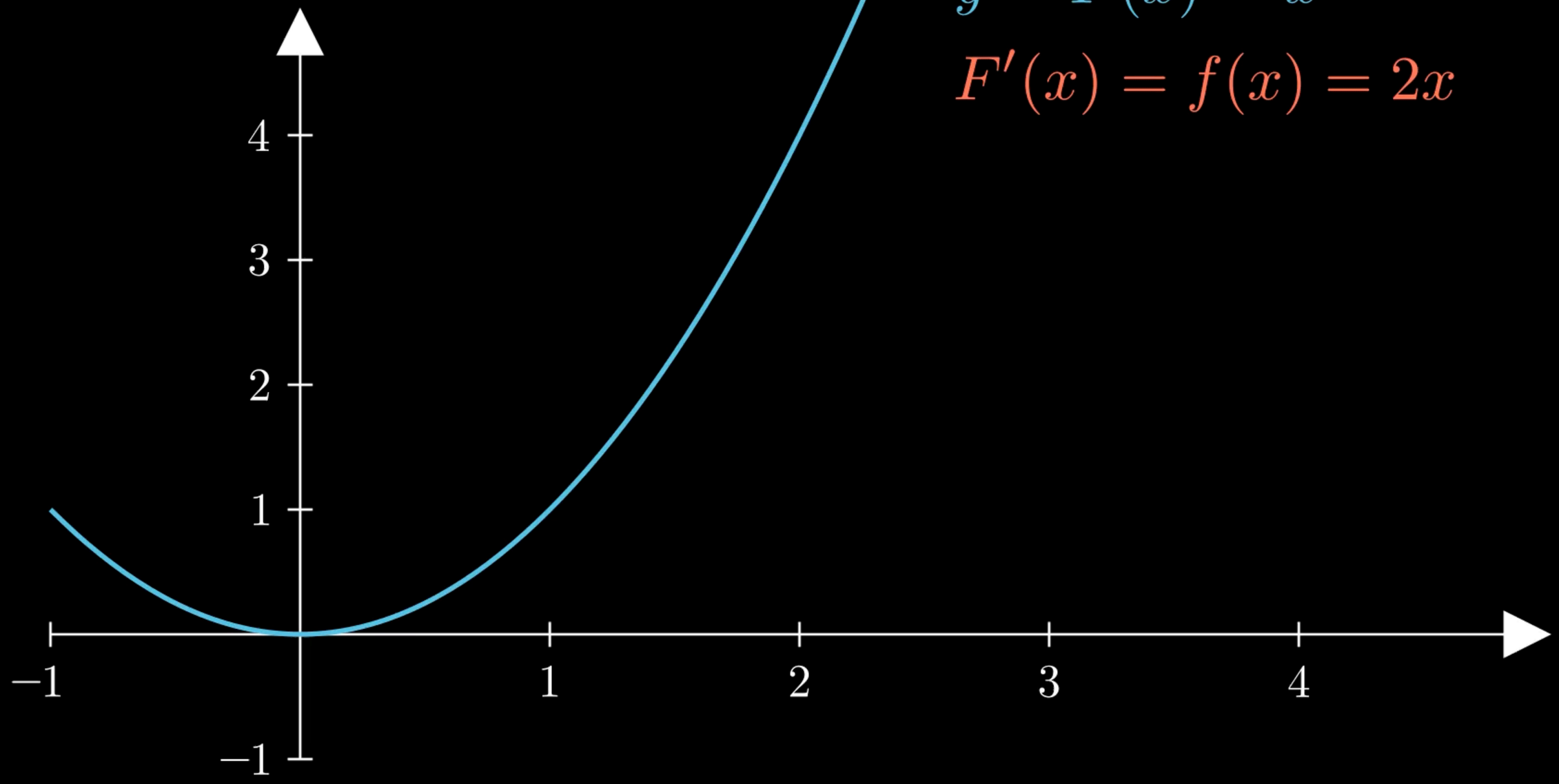
$$
\begin{aligned}
f(x) = F'(x) &= y' \\
&= \lim_{\Delta x \to 0} \frac{F(x+\Delta x) - F(x)}{\Delta x} \\
&= \lim_{\Delta x \to 0} \frac{(x+\Delta x) \times (x+\Delta x) - F(x)}{\Delta x} \\
&= \lim_{\Delta x \to 0} \frac{x^2 + 2x\Delta x + \Delta x^2 - x^2}{\Delta x} \\
&= \lim_{\Delta x \to 0} (2x + \Delta x) \\
&= 2x
\end{aligned}
$$



仅已知导函数f(x)，估计函数F(x)

$$y = F(x) = x^2$$

$$F'(x) = f(x) = 2x$$



$\Delta x = 1$

x	$f(x) = 2x$	$\Delta y = 2x \cdot \Delta x$	$\hat{F}(x)$	$F(x) = x^2$
0	0	0	0	0
1	2	2	0	1
2	4	4	2	4
3	6	6	6	9

$\Delta x = 0.5$

0	0	0	0	0
0.5	1	0.5	0	0.25
1	2	1	0.5	1
1.5	3	1.5	1.5	2.25
2	4	2	3	4
2.5	5	2.5	5	6.25
3	6	3	7.5	9

$\Delta x = 0.01$

	A	B	C	D
1	x	dy	F(x)_hat	F(x)
297	2.95	0.059	8.673	8.7025
298	2.96	0.0592	8.732	8.7616
299	2.97	0.0594	8.7912	8.8209
300	2.98	0.0596	8.8506	8.8804
301	2.99	0.0598	8.9102	8.9401
302	3	0.06	8.97	9

$$F(N) \approx \sum_{i=0}^{m} f(x_i) \cdot \Delta x, \quad \Delta x = N/m$$

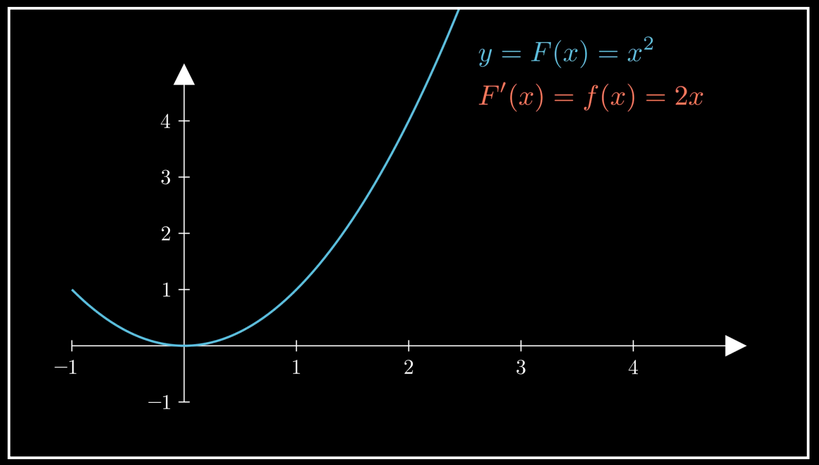
$$F(N) = \sum_{i=0}^{m \to \infty} f(x_i) \cdot \Delta x, c = \Delta x = N/m \to 0$$

$$\int_{x=0}^{N} f(x)\, dx = F(N) - F(0)$$

更常见理解：曲线下面积(Area under curve)

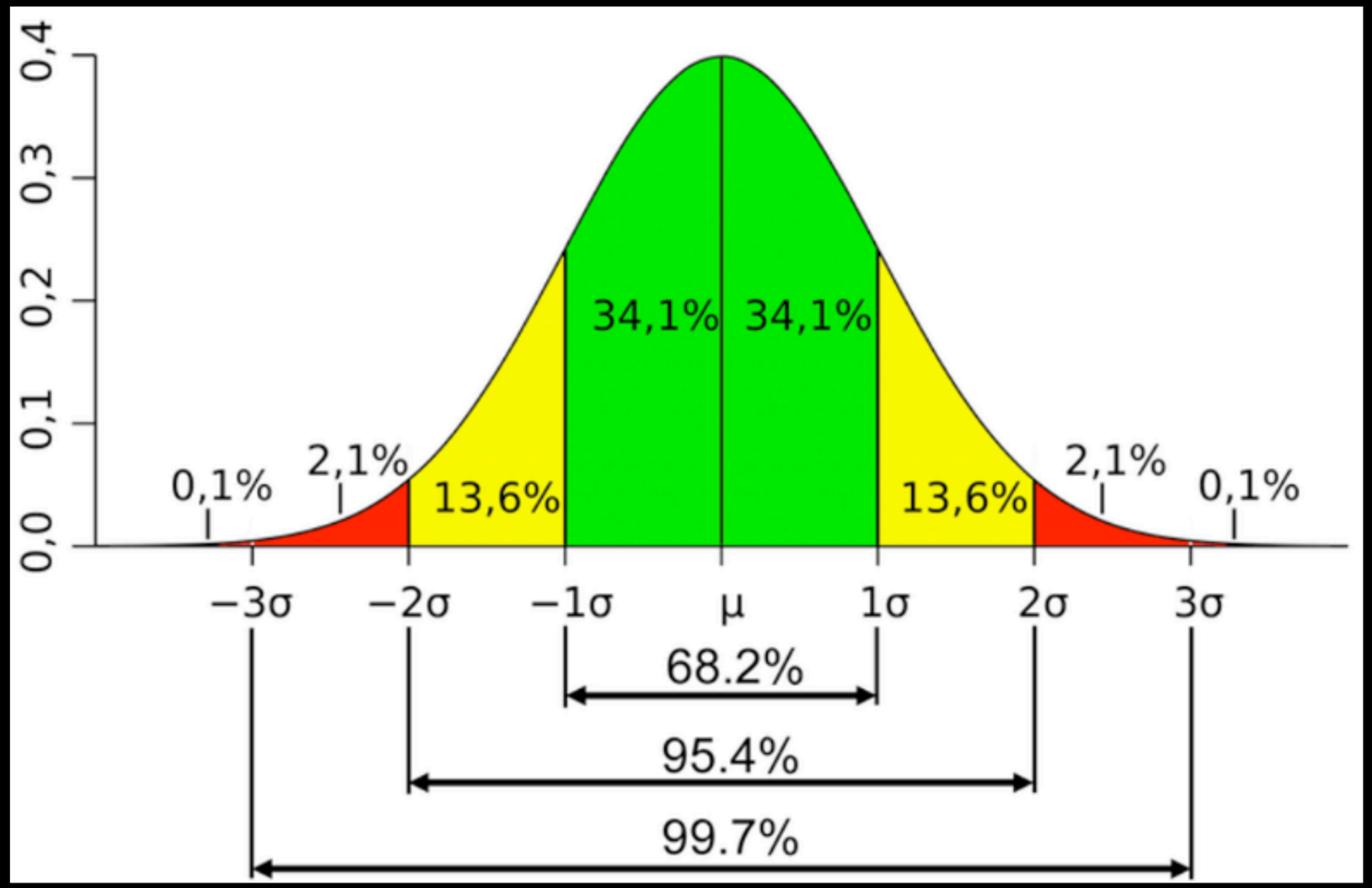
$$f(x) = 2 \times x$$

$$F(x) = S = x^2$$



4
3
2
1
−1
1
2
3
4

更常见理解：曲线下面积(Area under curve)



$$\int_{x=x_l}^{x_u} f(x)\,dx = F(x_u) - F(x_l)$$

标准正态分布：概率密度函数(pdf)
积分得到概率

一些常见的导数公式

Basic Properties and Formulas

If $f(x)$ and $g(x)$ are differentiable functions (the derivative exists), c and n are any real numbers,

1. $(cf)' = cf'(x)$
2. $(f \pm g)' = f'(x) \pm g'(x)$
3. $(fg)' = f'g + fg'$ – **Product Rule**
4. $\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$ – **Quotient Rule**
5. $\frac{d}{dx}(c) = 0$
6. $\frac{d}{dx}(x^n) = nx^{n-1}$ – **Power Rule**
7. $\frac{d}{dx}(f(g(x))) = f'(g(x))g'(x)$
   This is the **Chain Rule**

Common Derivatives

$$\frac{d}{dx}(x) = 1$$
$$\frac{d}{dx}(\sin x) = \cos x$$
$$\frac{d}{dx}(\cos x) = -\sin x$$
$$\frac{d}{dx}(\tan x) = \sec^2 x$$
$$\frac{d}{dx}(\sec x) = \sec x \tan x$$
$$\frac{d}{dx}(\csc x) = -\csc x \cot x$$
$$\frac{d}{dx}(\cot x) = -\csc^2 x$$
$$\frac{d}{dx}(\sin^{-1} x) = \frac{1}{\sqrt{1-x^2}}$$
$$\frac{d}{dx}(\cos^{-1} x) = -\frac{1}{\sqrt{1-x^2}}$$
$$\frac{d}{dx}(\tan^{-1} x) = \frac{1}{1+x^2}$$
$$\frac{d}{dx}(a^x) = a^x \ln(a)$$
$$\frac{d}{dx}(\mathbf{e}^x) = \mathbf{e}^x$$
$$\frac{d}{dx}(\ln(x)) = \frac{1}{x}, \quad x > 0$$
$$\frac{d}{dx}(\ln|x|) = \frac{1}{x}, \quad x \neq 0$$
$$\frac{d}{dx}(\log_a(x)) = \frac{1}{x \ln a}, \quad x > 0$$

Source: https://www.yumpu.com/en/document/read/40809568/calculus-cheat-sheet-derivatives-pauls-online-math-notes

Derivative Rules

Ace Tutors

Product Rule

乘法求导规则

$$y = f(x) \cdot g(x)$$

$$y' = f'(x) \cdot g(x) + f(x) \cdot g'(x)$$

Ex

$$y = x^4 \cdot sin(x)$$

$$f(x) = x^4 \qquad g(x) = sin(x)$$

$$f'(x) = 4x^3 \qquad g'(x) = cos(x)$$

$$y' = 4x^3 \cdot sin(x) + x^4 \cdot cos(x)$$

Ex

$$y = 2x^2 \cdot ln(x)$$

$$f(x) = 2x^2 \qquad g(x) = ln(x)$$

$$f'(x) = 4x \qquad g'(x) = \frac{1}{x}$$

$$y' = 4x \cdot ln(x) + 2x^2 \cdot \frac{1}{x}$$

$$y' = 4x \cdot ln(x) + 2x$$

Quotient Rule

除法求导规则

$$y = \frac{f(x)}{g(x)} = \frac{hi}{lo}$$

$$y' = \frac{lo \cdot dhi - hi \cdot dlo}{(lo)^2}$$

Ex

$$y = \frac{e^x}{4x^3}$$

$$hi = e^x \qquad lo = 4x^3$$

$$dhi = e^x \qquad dlo = 12x^2$$

$$y' = \frac{4x^3 \cdot e^x - e^x \cdot 12x^2}{(4x^3)^2}$$

Ex

$$y = \frac{5x}{sin(x)}$$

$$hi = 5x \qquad lo = sin(x)$$

$$dhi = 5 \qquad dlo = cos(x)$$

$$y' = \frac{sin(x) \cdot 5 - 5x \cdot cos(x)}{(sin(x))^2}$$

Chain Rule

链式求导规则

$$y = f(g(x))$$

$$y' = f'(g(x)) \cdot g'(x)$$

Ex

$$y = cos(3x^2)$$

$$f(g(x)) = cos(3x^2) \qquad g(x) = 3x^2$$

$$f'(g(x)) = -sin(3x^2) \qquad g'(x) = 6x$$

$$y' = -sin(3x^2) \cdot 6x$$

Ex

$$y = e^{2x^3+4}$$

$$f(g(x)) = e^{2x^3+4} \qquad g(x) = 2x^3 + 4$$

$$f'(g(x)) = e^{2x^3+4} \qquad g'(x) = 6x^2$$

$$y' = e^{2x^3+4} \cdot 6x^2$$

求导：链式法则 (chain rule)

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

$$z = \log(x_1 \cdot x_2) \cdot \sin x_2$$

$$\frac{dz}{dx_1} = \frac{x_2}{x_1 x_2} \cdot \sin x_2 + \log(x_1 x_2) \cdot 0 = \frac{\sin x_2}{x_1}$$

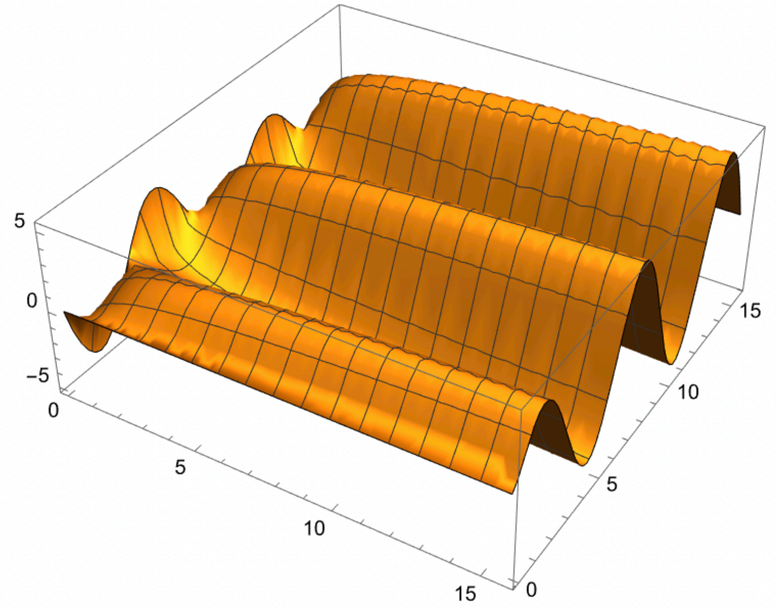
$$\frac{dz}{dx_2} = \frac{x_1}{x_1 x_2} \cdot \sin x_2 + \log(x_1 x_2) \cdot \cos x_2 = \frac{\sin x_2}{x_2} + (\cos x_2) \cdot \log(x_1 x_2)$$

手算结果（基本上是高中数学内容）

Take derivatives (求导)

```
In[2]:= f[z] := Log[x1 * x2] * Sin[x2]

In[3]:= D[f[z], x1]

Out[3]= Sin[x2]/x1

In[4]:= D[f[z], x2]

Out[4]= Cos[x2] Log[x1 x2] + Sin[x2]/x2

In[5]:= Plot3D[Log[x1 * x2] * Sin[x2], {x1, 0.01, 5*Pi}, {x2, 0.01, 5*Pi}]
```

Out[5]=



WOLFRAM

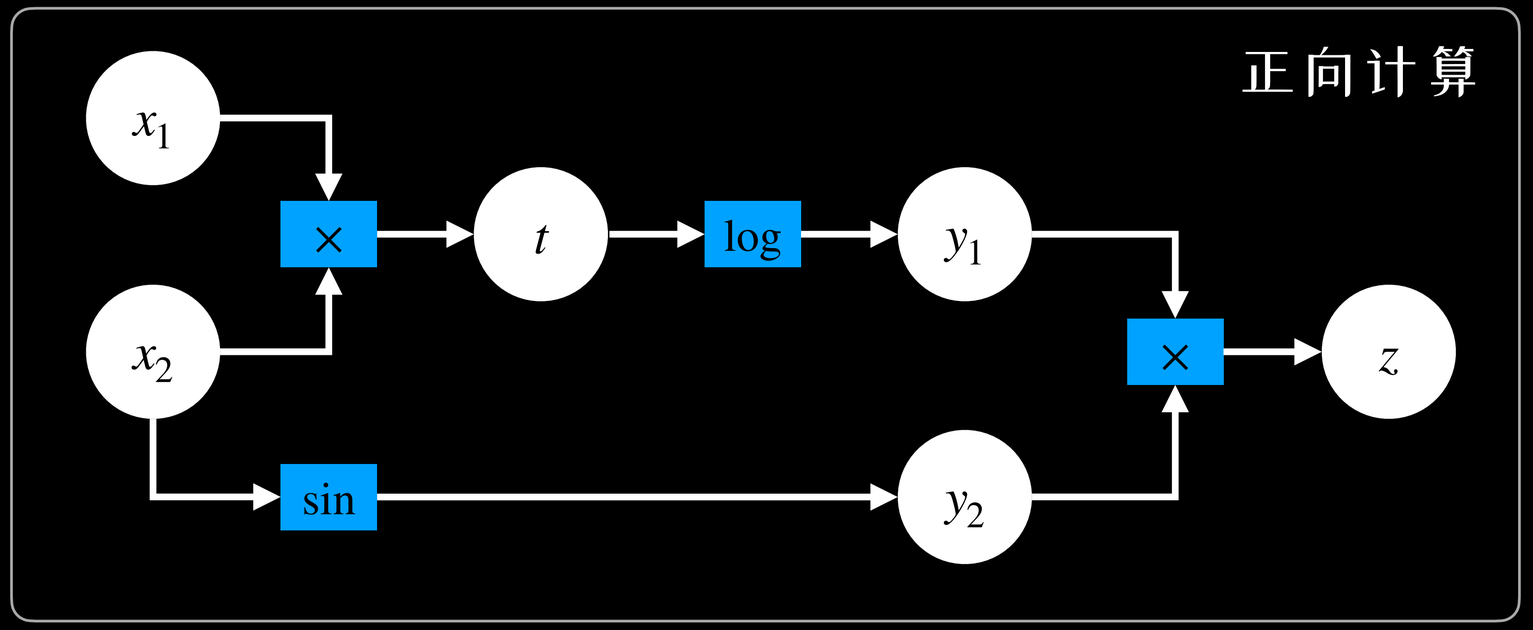
Mathematica求导结果

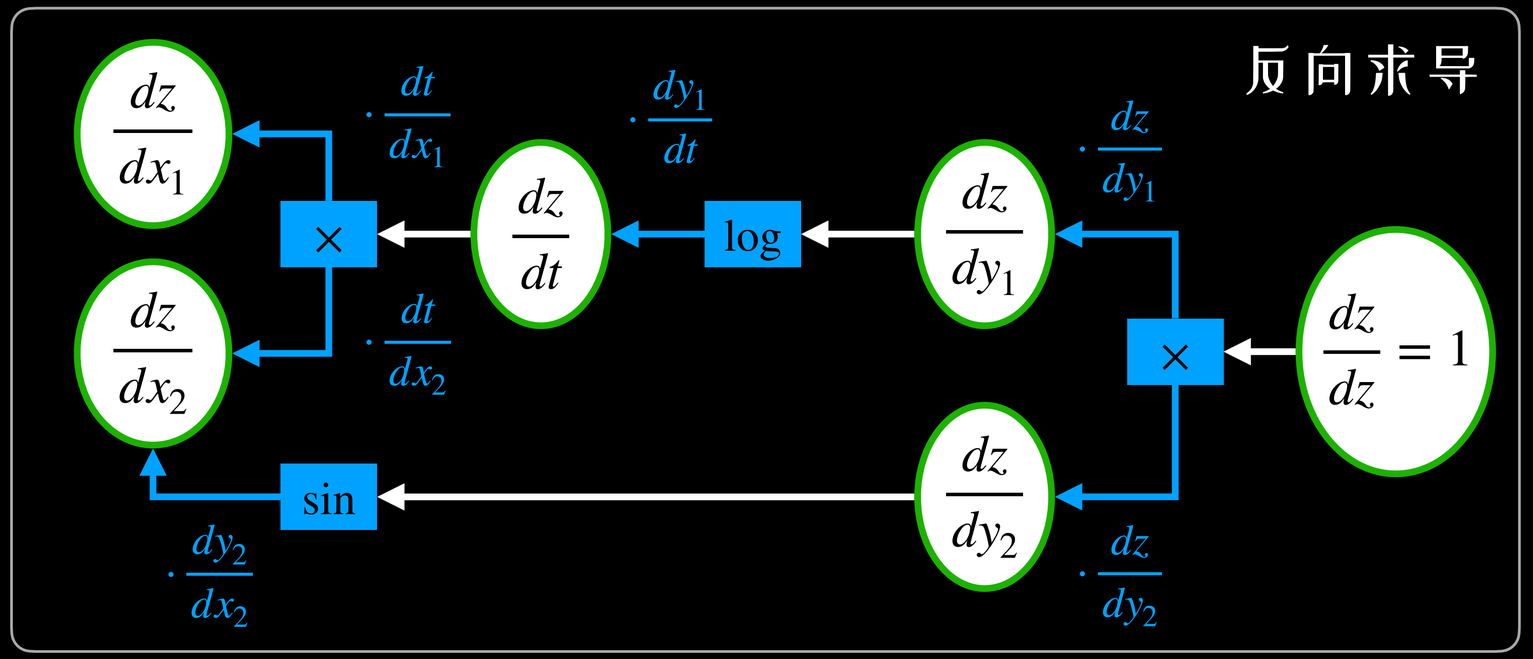
PyTorch/Tensorflow:
通过计算图推导/求导

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

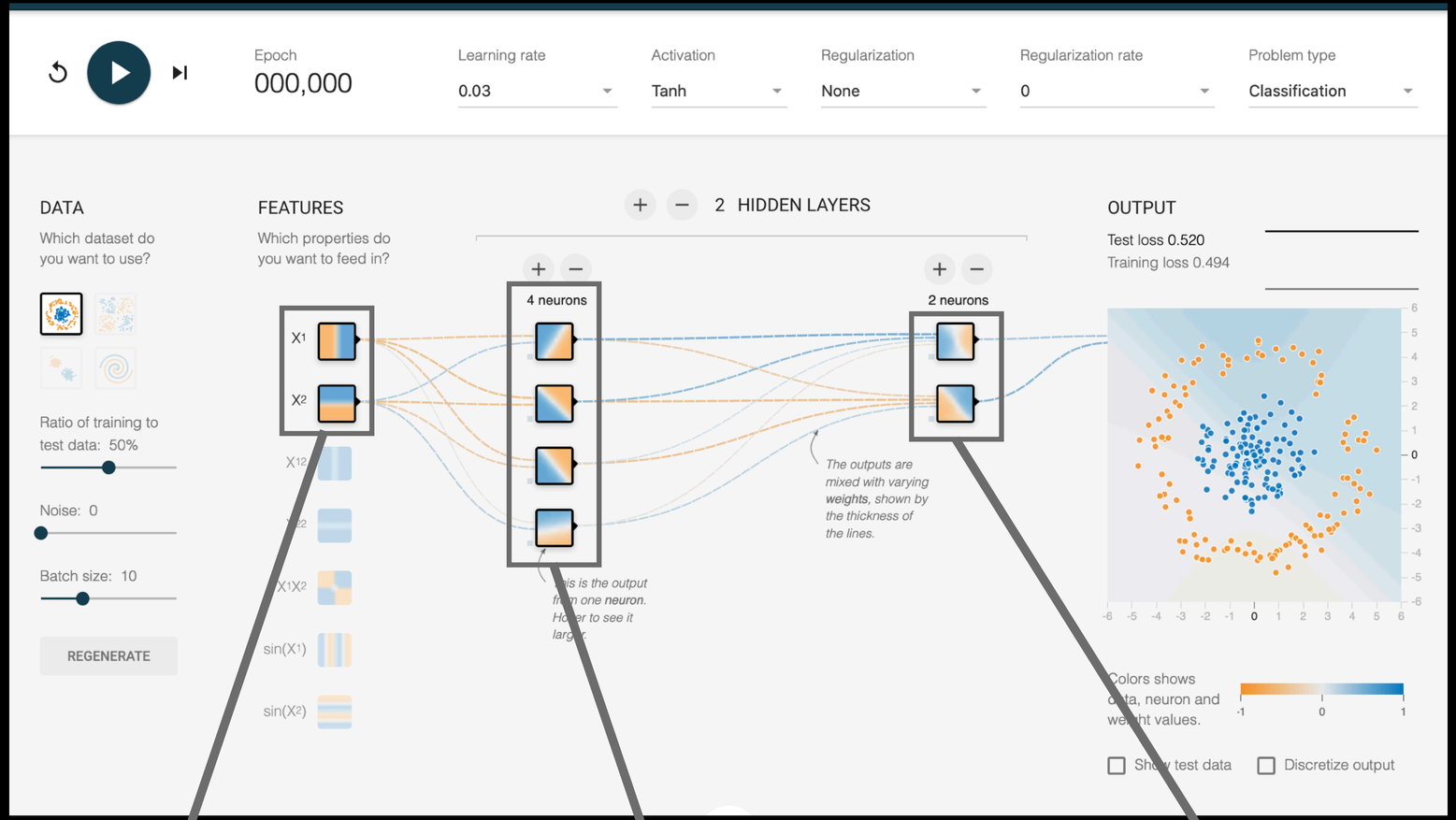
$$z = \log(x_1 \cdot x_2) \cdot \sin x_2$$

1. 每种运算的导数是确定的
2. 通过计算图，实现连续求导
3. 一遍运算，求出所有偏导
4. 需要带入具体数值（不做符号推导，只是传递导数数值）





深度学习：线性变换(全联接网络/FCNN)



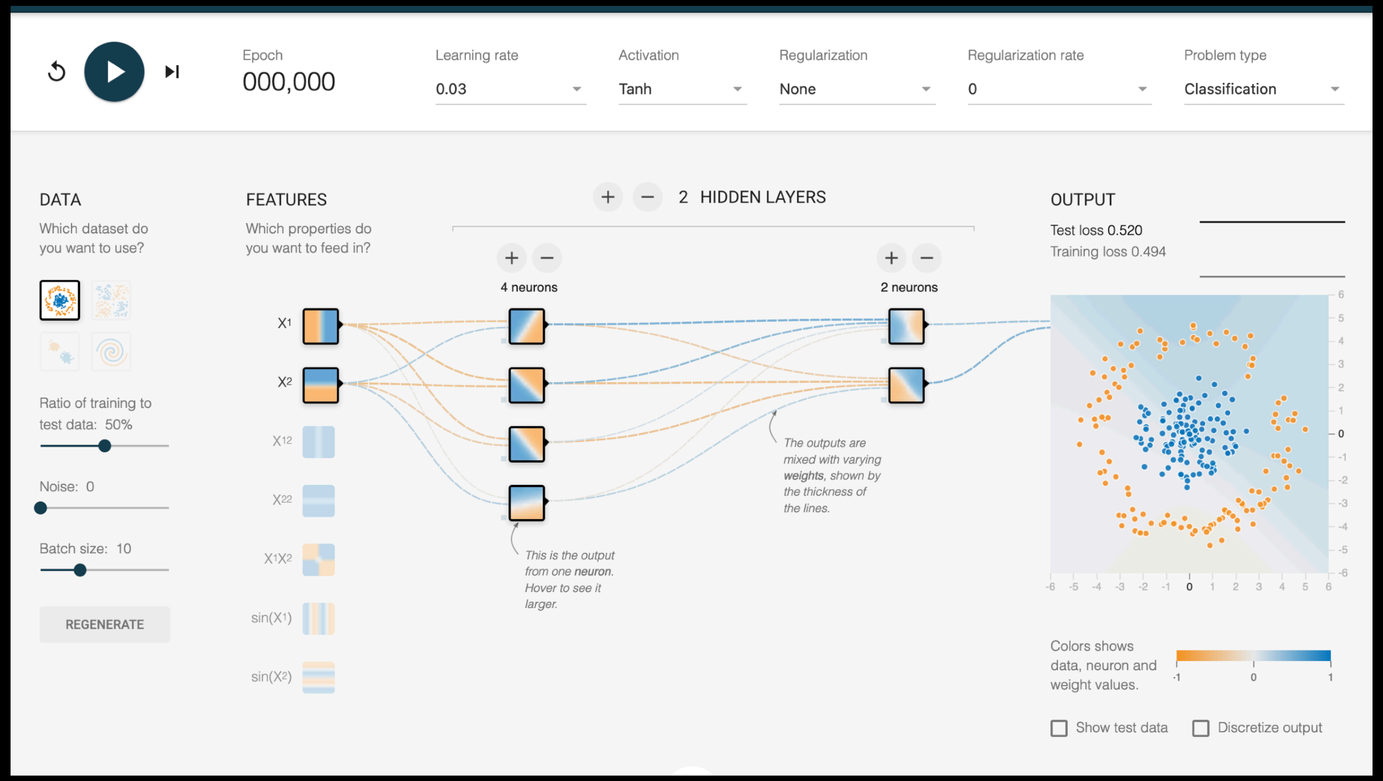
Source: Tensorflow Playground: https://playground.tensorflow.org/

$$[x_1, x_2] \rightarrow \cdot M_{2\times4} \rightarrow [y_{11}, y_{12}, y_{13}, y_{14}] \rightarrow \cdot M_{4\times2} \rightarrow [y_{21}, y_{22}]$$

分类器

1. 输入：(x1, x2)
2. 输出：y=-1 or 1

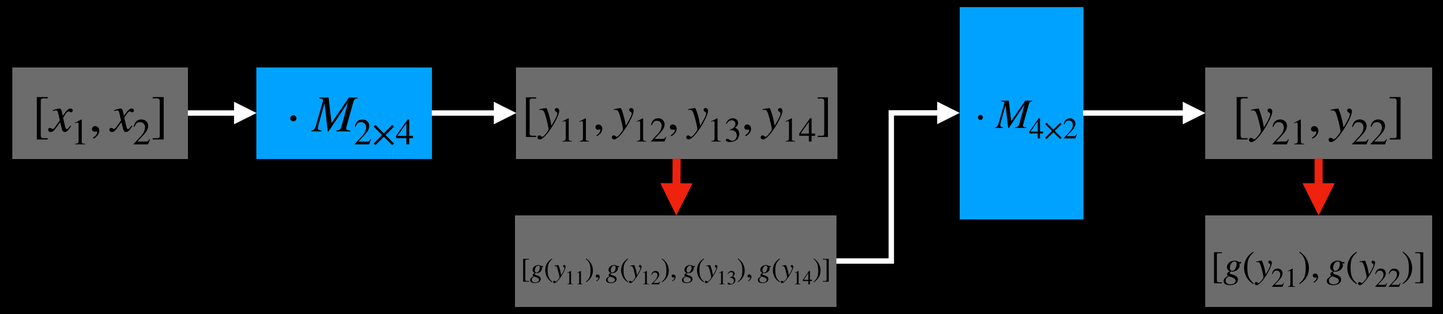
深度学习：线性变换(全联接网络/FCNN)



Source: Tensorflow Playground: https://playground.tensorflow.org/

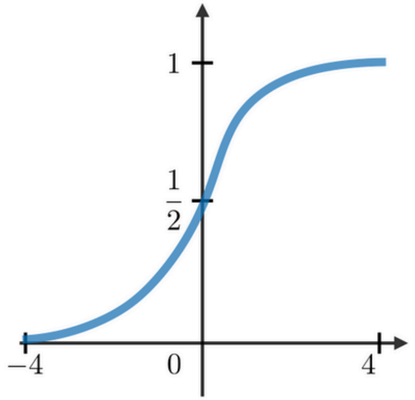
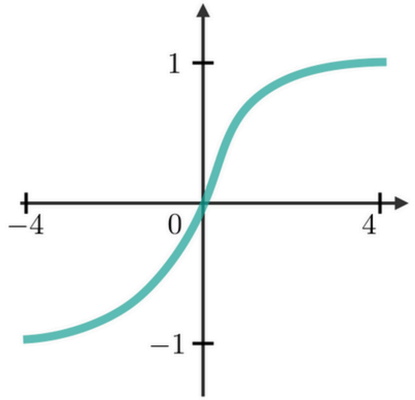
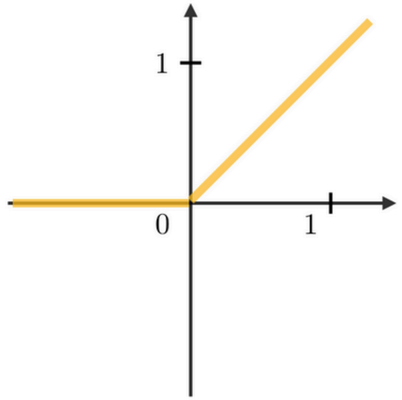
$$M_{2\times4} \bullet M_{4\times2} = M_{2\times2}$$

多个矩阵相乘 等价于 一个矩阵
所以，需要在那个矩阵运算后添加非线性变换



深度学习：非线性变换（单个神经元的变换）

❒ **Commonly used activation functions** — The most common activation functions used in RNN modules are described below:

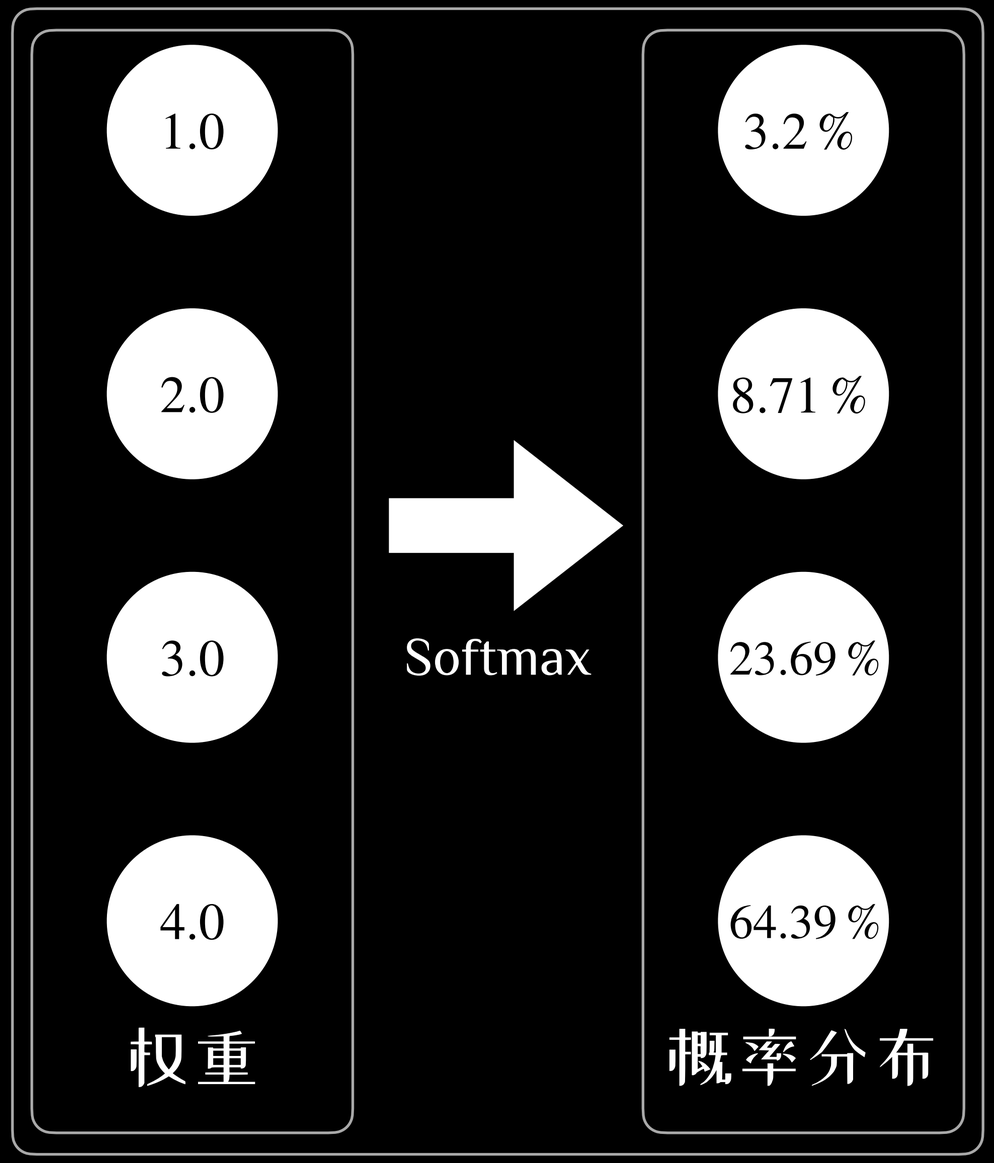
Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

$$\frac{dg(z)}{dz} = g(z)(1 - g(z))$$

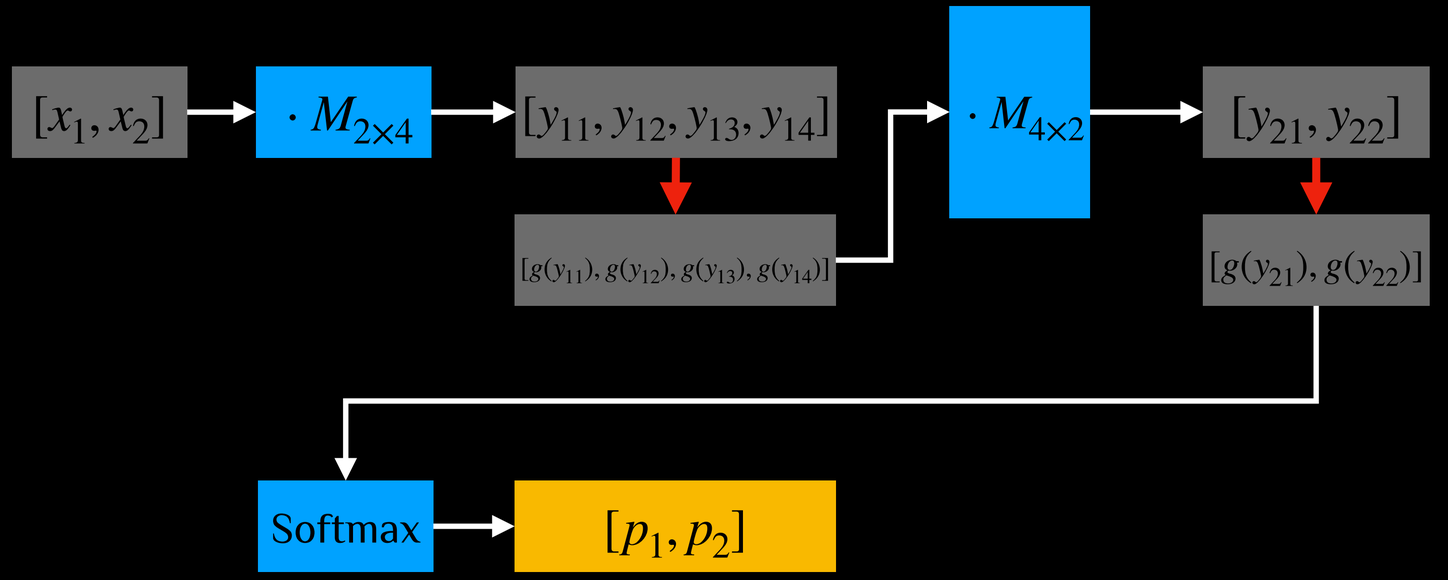
$$\frac{dg(z)}{dz} = 1 - (g(z))^2$$

$$\frac{dg(z)}{dz} = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

深度学习：非线性变换（Softmax单层神经元变换为概率）



$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K.$$



深度学习: 非线性变换（损失函数）

Cross Entropy Loss

$$l = -\sum_{i=1}^{M} t_i \log p_i, \ M \text{ 个标注类型}$$

$$L = \sum_{j=1}^{N} l, \ N \text{ 条训练数据}$$

样本1

[0,1] 真实概率

[0.1,0.9] 模型预测概率

$$l = -(0 \cdot log[0.1] + 1 \cdot log[0.9]) = 0.1053$$

样本2

[0,1] 真实概率

[0.9,0.1] 模型预测概率

$$l = -(0 \cdot log[0.9] + 1 \cdot log[0.1]) = 2.3026$$

深度学习: 非线性变换（损失函数）

$$l = -\sum_{i=1}^{M} t_i \log p_i, M\text{个标注类型}$$

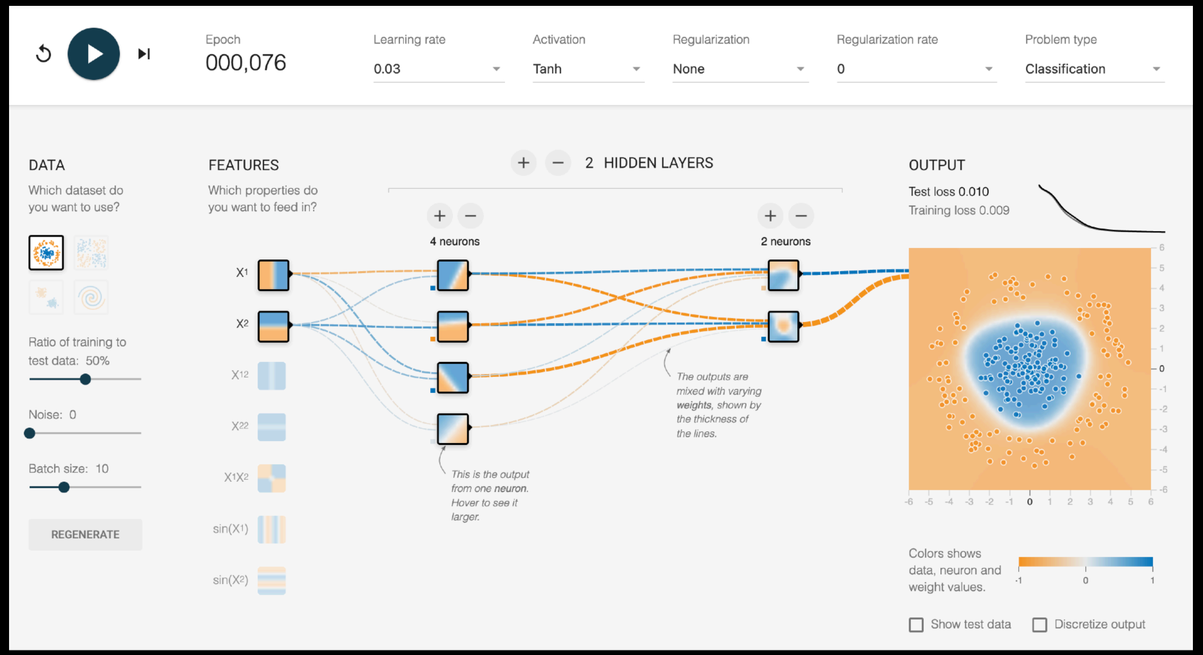
$$L = \sum_{j=1}^{N} l, N\text{条训练数据}$$

$$\frac{\partial L}{\partial w_k}, \forall k \in \{1, \ldots, K\}$$

梯度下降更新参数(Gradient Descent)

$$w_k := w_k - \gamma \frac{\partial L}{\partial w_k} \rightarrow \begin{cases} n = N, \text{G.D.} \\ n = \text{B.S.} = 4, \text{S.G.D.} \end{cases}$$

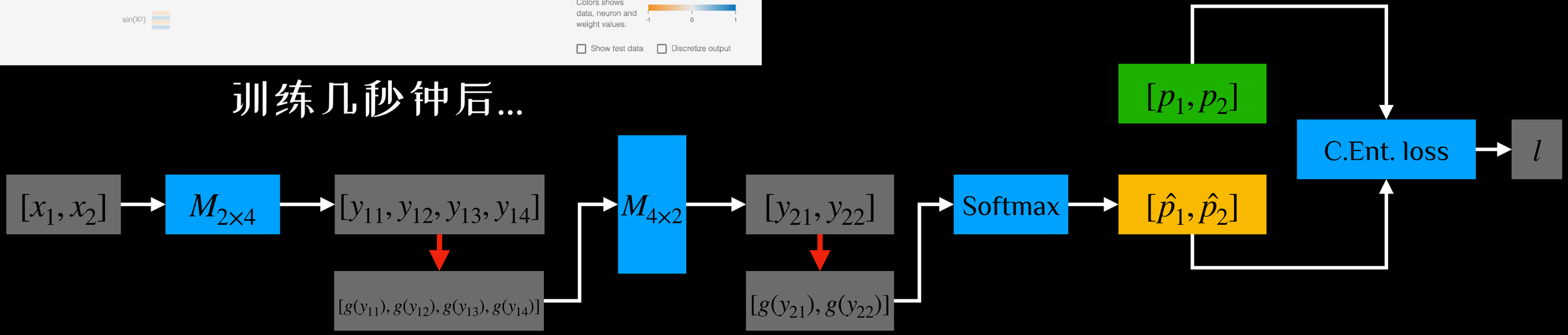
深度学习：线性变换(全联接网络/FCNN)



需要训练的模型参数(矩阵权重)：

$2 \times 4 + 4 \times 2 = 16$

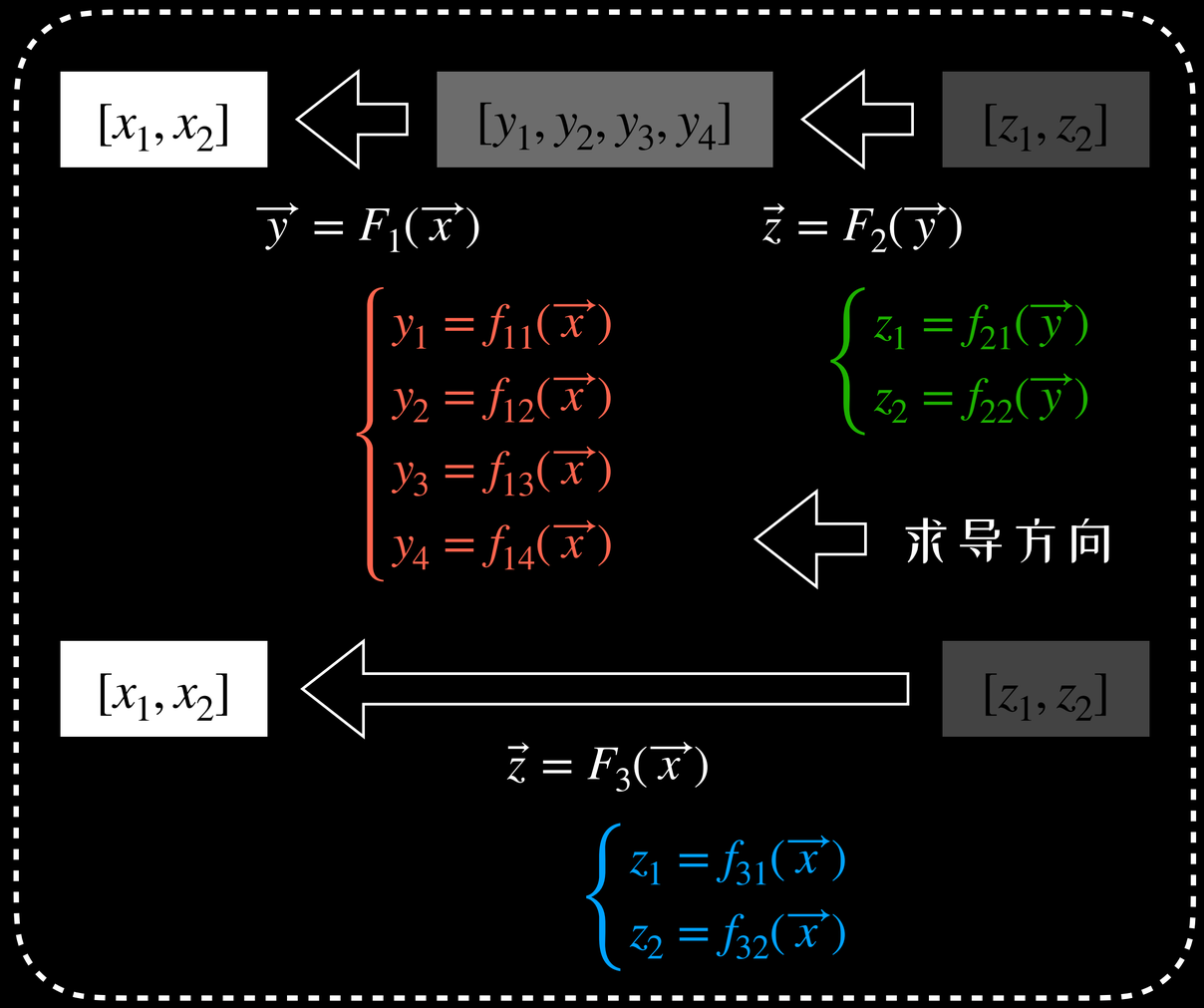
训练几秒钟后...



对应的大致计算图

向量化/矩阵化/张量化(vectorization)

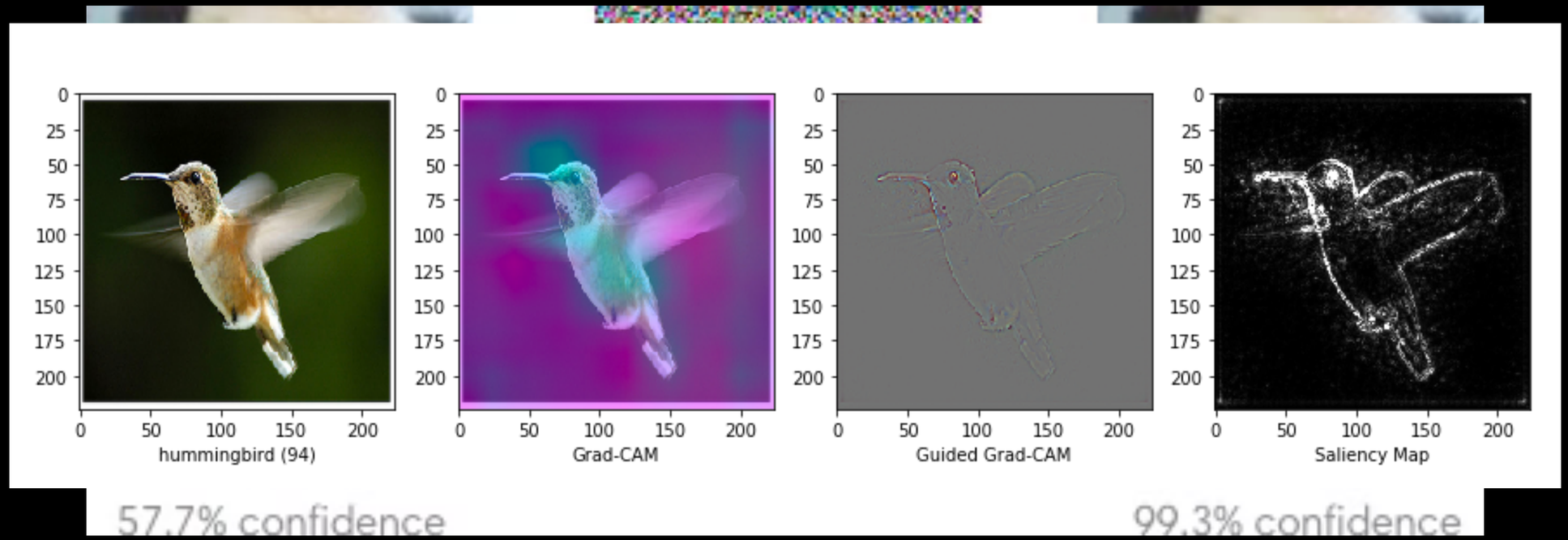
- Jacobian Matrix (每层之间导数传递遵循矩阵“乘法”运算法则)



$$\begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} & \frac{\partial z_1}{\partial y_3} & \frac{\partial z_1}{\partial y_4} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} & \frac{\partial z_2}{\partial y_3} & \frac{\partial z_2}{\partial y_4} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} \\ \frac{\partial y_4}{\partial x_1} & \frac{\partial y_4}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} \end{bmatrix}$$

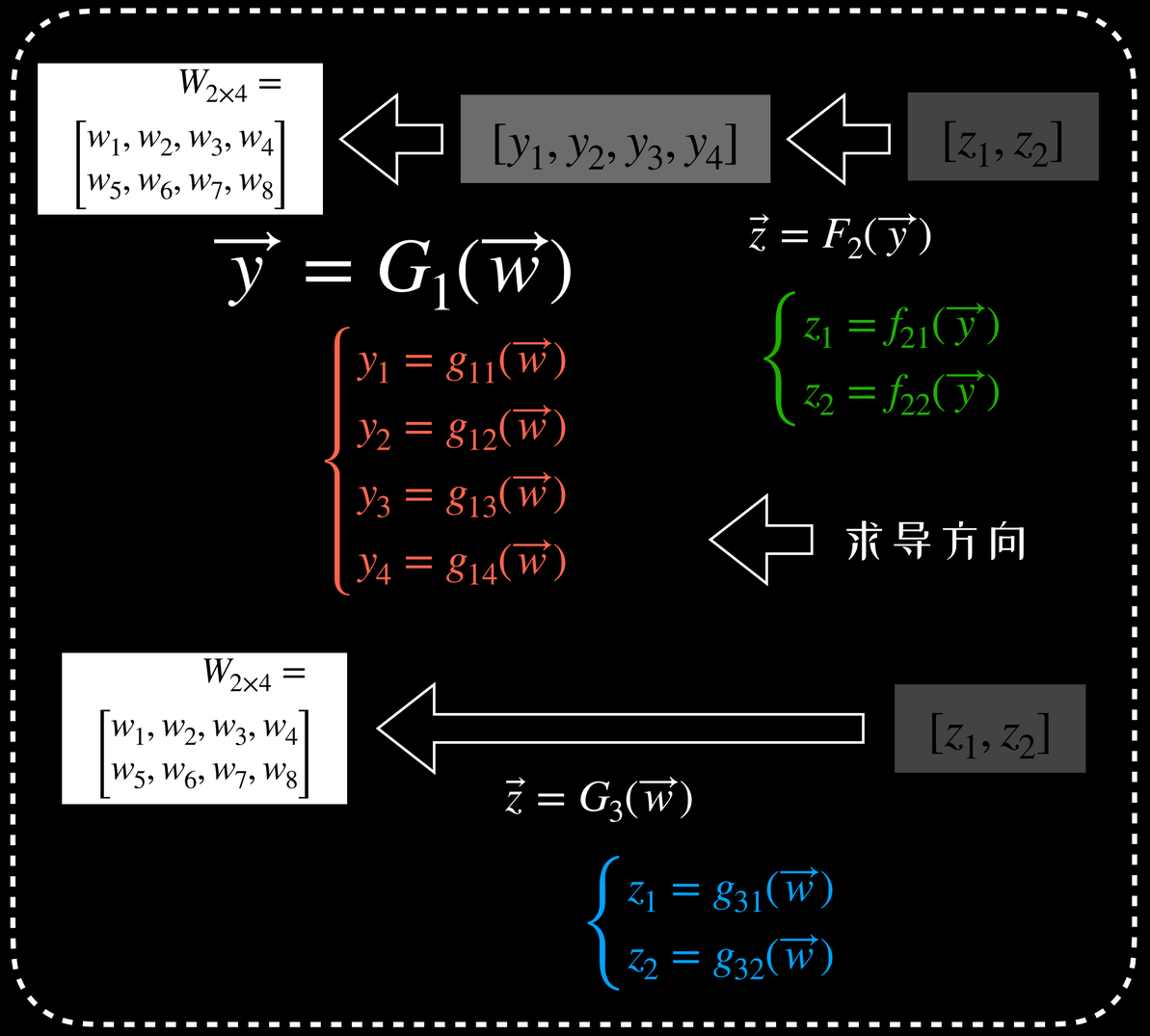
向量化/矩阵化/张量化(vectorization)

- 这里是对 “数据” / “神经元” 进行求导
  - 应用1 攻击模型 Adversarial Attack
  - 应用2 解释模型 saliency map (突出 “导数更大” 的图片像素)



向量化/矩阵化/张量化(vectorization)

- Jacobian Matrix (每层之间导数传递遵循矩阵“乘法”运算法则)



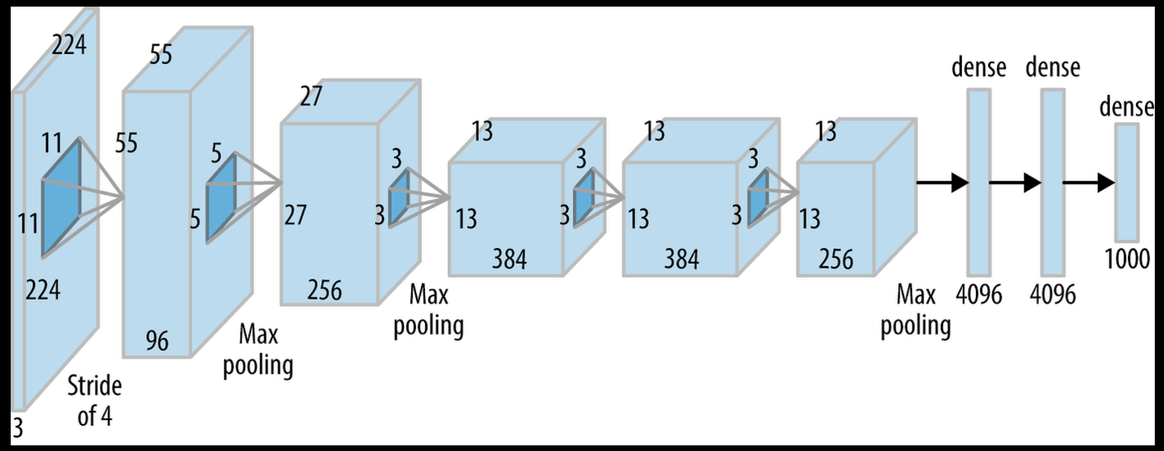
$$\begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} & \frac{\partial z_1}{\partial y_3} & \frac{\partial z_1}{\partial y_4} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} & \frac{\partial z_2}{\partial y_3} & \frac{\partial z_2}{\partial y_4} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_8} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_8} \\ \frac{\partial y_3}{\partial w_1} & \frac{\partial y_3}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_8} \\ \frac{\partial y_4}{\partial w_1} & \frac{\partial y_4}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_8} \end{bmatrix} = \begin{bmatrix} \frac{\partial z_1}{\partial w_1} & \frac{\partial z_1}{\partial w_2} & \cdots & \frac{\partial z_1}{\partial w_8} \\ \frac{\partial z_2}{\partial w_1} & \frac{\partial z_2}{\partial w_2} & \cdots & \frac{\partial z_1}{\partial w_8} \end{bmatrix}$$

- 训练中，对模型参数求导也是非常类似的
- 我的观点：“自动求导机”
  - 通过计算图，可以求二阶导/多阶导（也很多基于二阶导数的方法）

卷积神经网络（CNN）

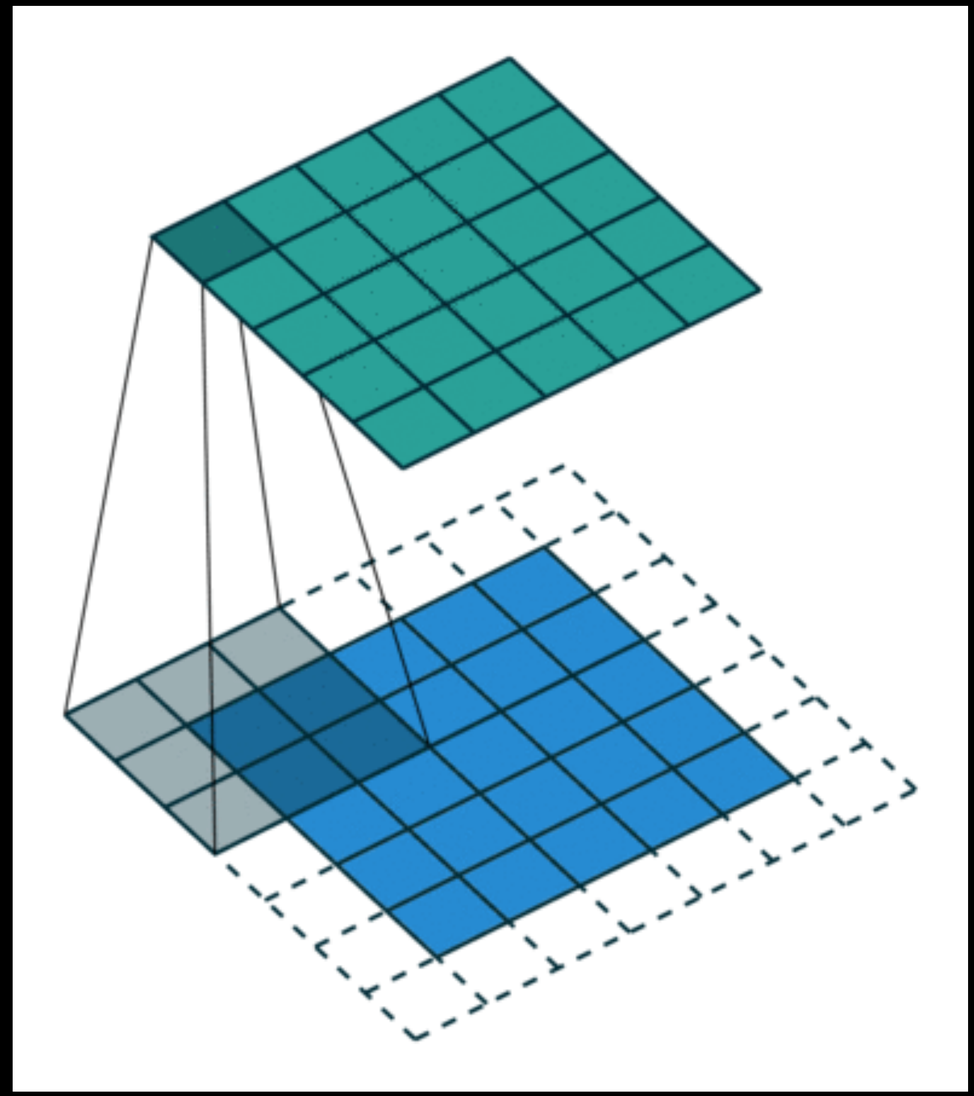
Convolutional Neural Network

- FCNN参数量很大
- 神经元互相影响，一定阶段后难以训练
- CNN使用卷积运算
  - 降低参数量，高效提取图片信息



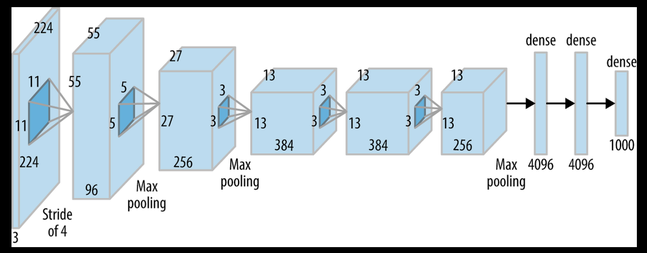
AlexNet 结构

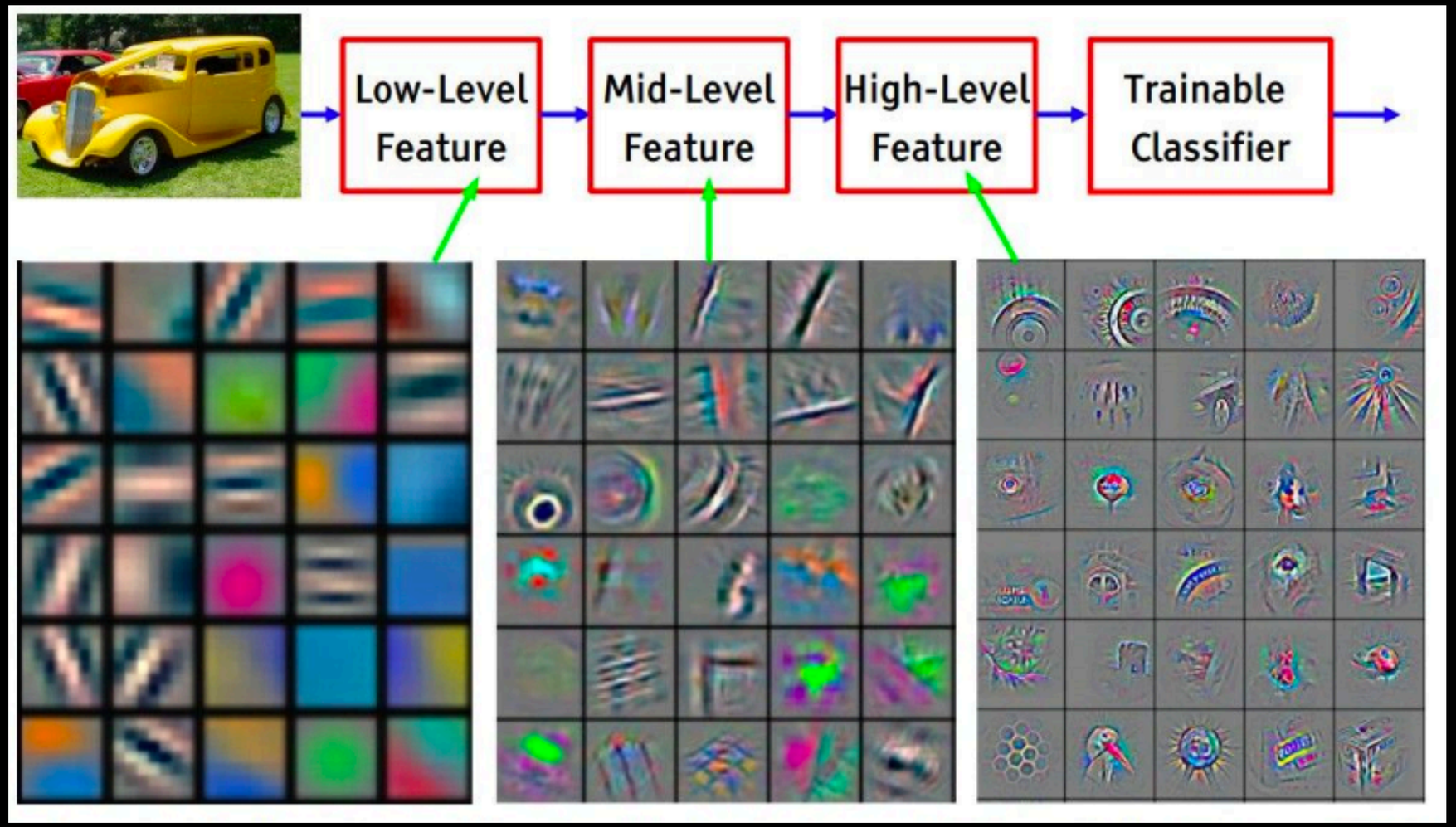
Source：https://www.researchgate.net/figure/Simplified-illustration-of-the-AlexNet-architecture_fig2_329790469



卷积神经网络（CNN）

Convolutional Neural Network





*统计学视角：训练超高维空间中的概率分布

- 每个输入数据属于一个 m 维空间
  - $\vec{x_i} \in X \subset R^m$
- 每个数据对应的结果也在这个k维空间
  - $\vec{y_i} \in Y \subset R^k$
- 模型输出结果也属于一个 k 维空间
  - $\vec{o_i} \in O \subset R^k$
- 模型参数属于一个 n 维空间 $\vec{w} \in R^n$
- 问题：使用概率论语言（概率分布）描述机器学习（监督学习）模型?

$$p_t(Y|X)$$ 给出数据，对应结果分布

$$\arg\min_w D_{KL}(q_w||p_t)$$ KL Divergent：衡量两个分布不不同

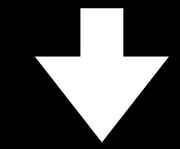
$$\approx \arg\min_w \sum_{i=1}^{N} L_w(f_w(\vec{x_i}), \vec{y_i})$$

$$q_w(O|X, \vec{w})$$ 使用模型预测结果的分布

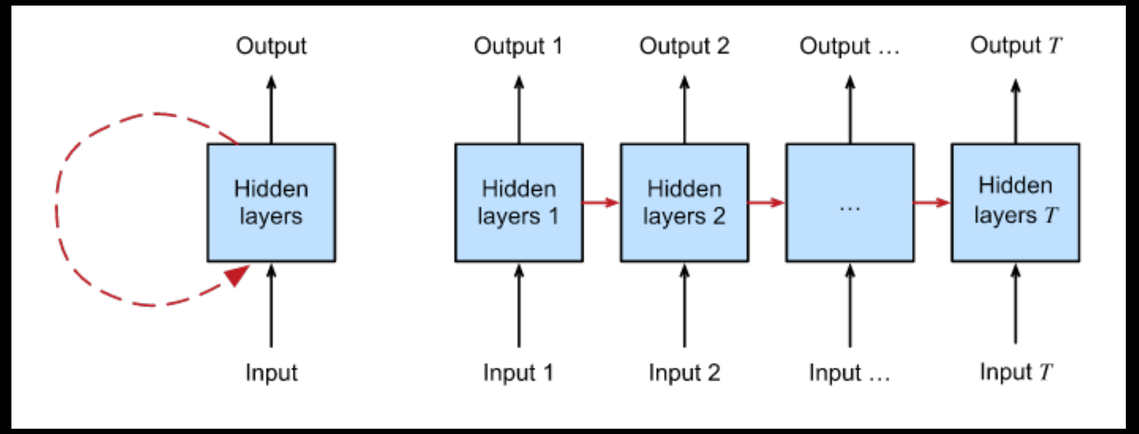
- 核心思想：只要维度够高，通过概率分布组合可以得到任意复杂的分类器
- 观点：记忆的过程是在高纬空间中的记录/索取“高维向量”的能力

Recurrent Neural Network
循环神经网络

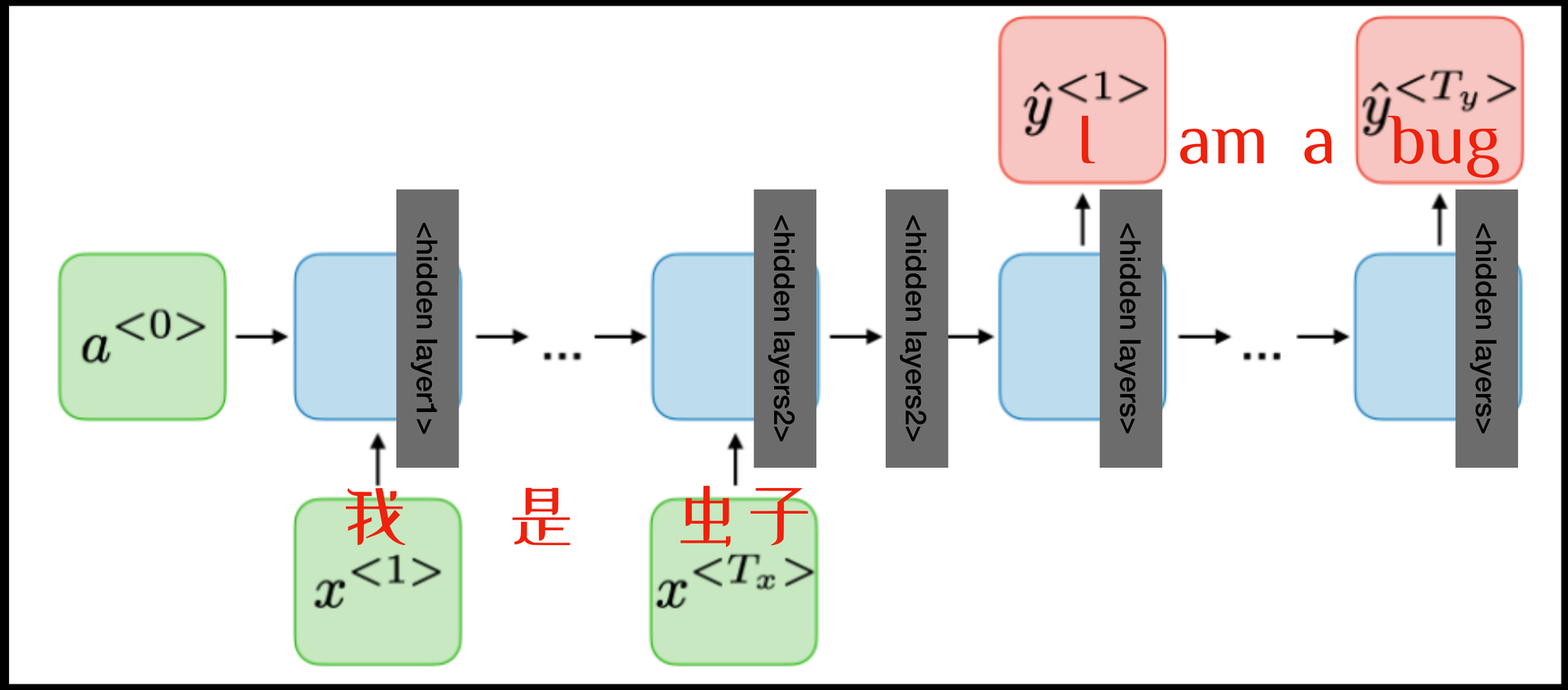
我 是 虫子.



I am a bug.

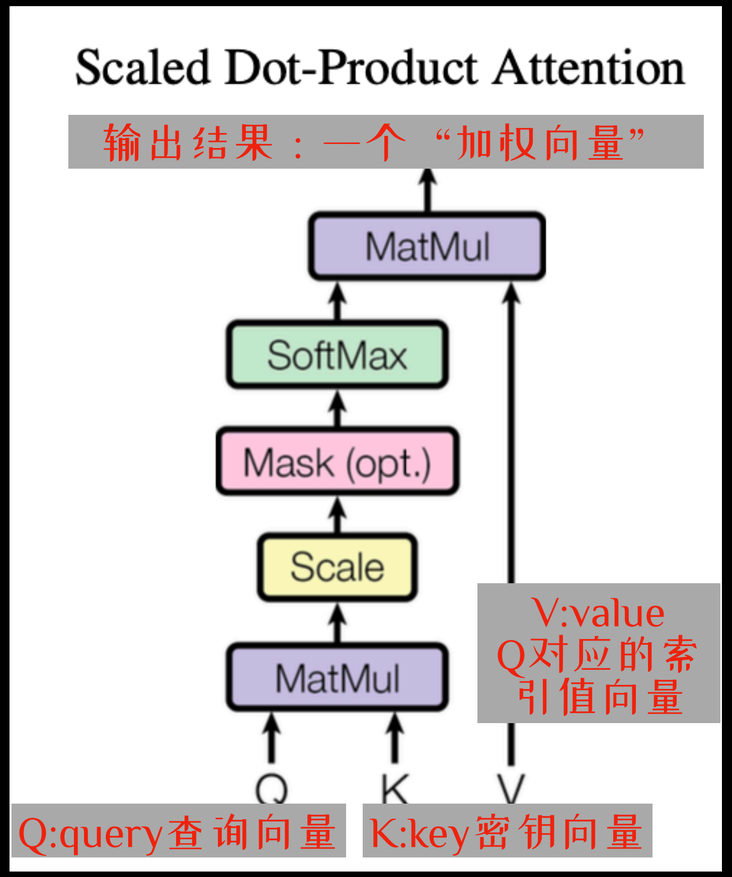


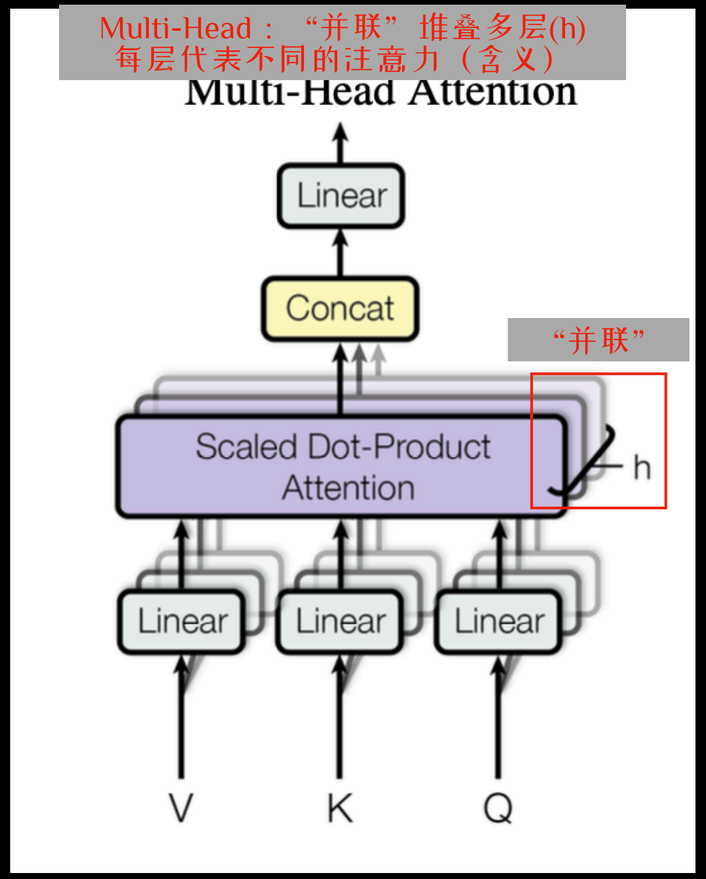
循环神经网络



Many-to-Many结构（用作语言翻译/对话）

注意力机制：《Attention is All You Need》





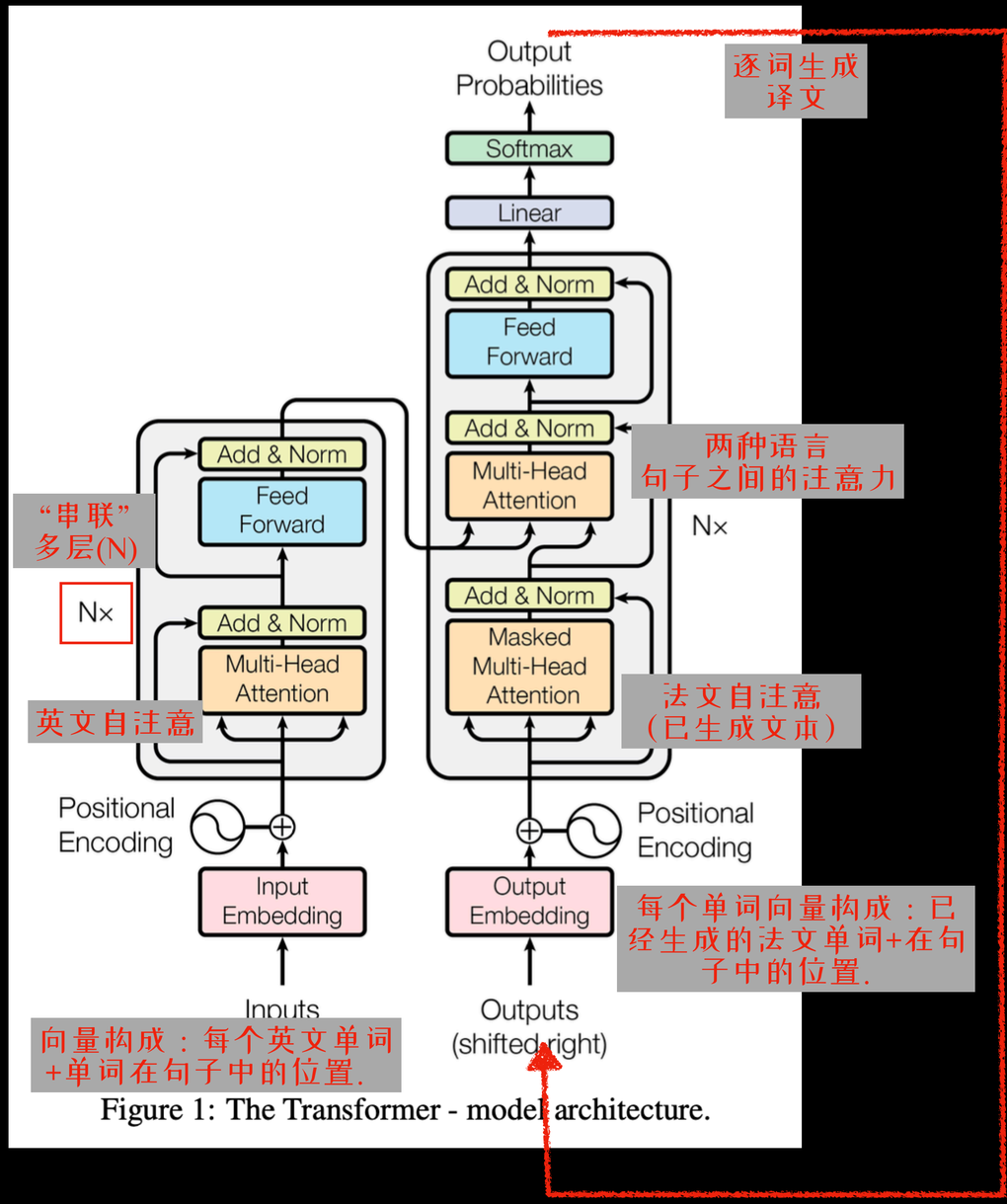
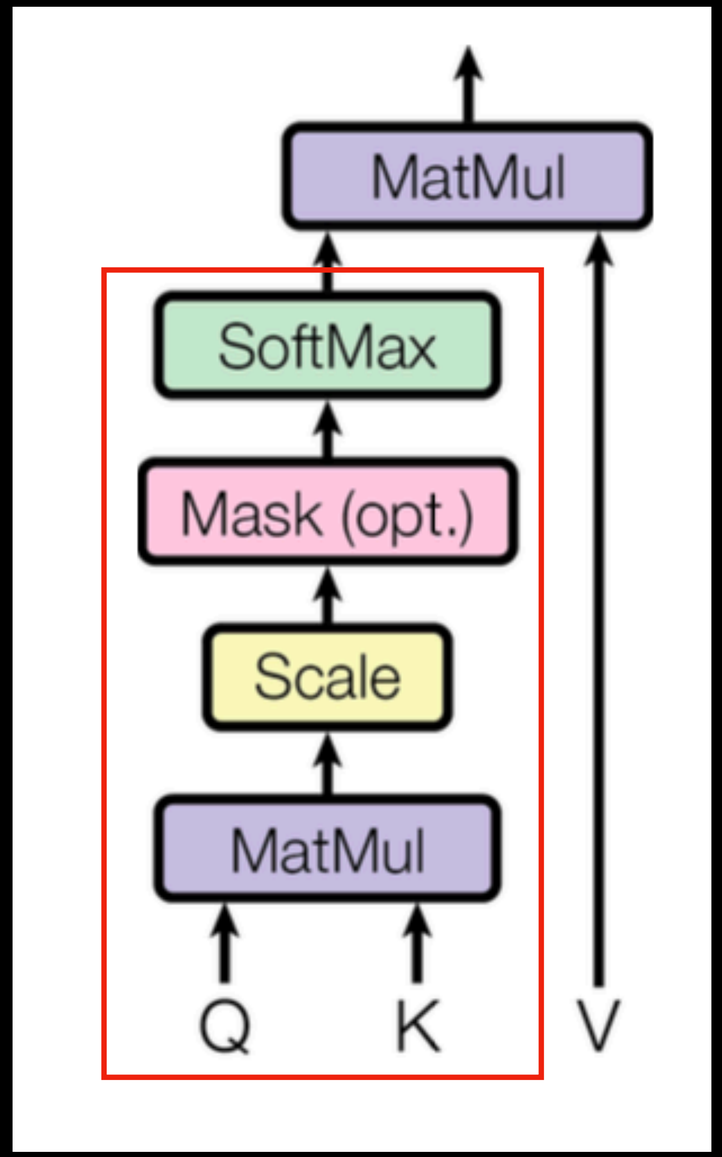


Figure 1: The Transformer - model architecture.

注意力机制（Attention Mechanism）

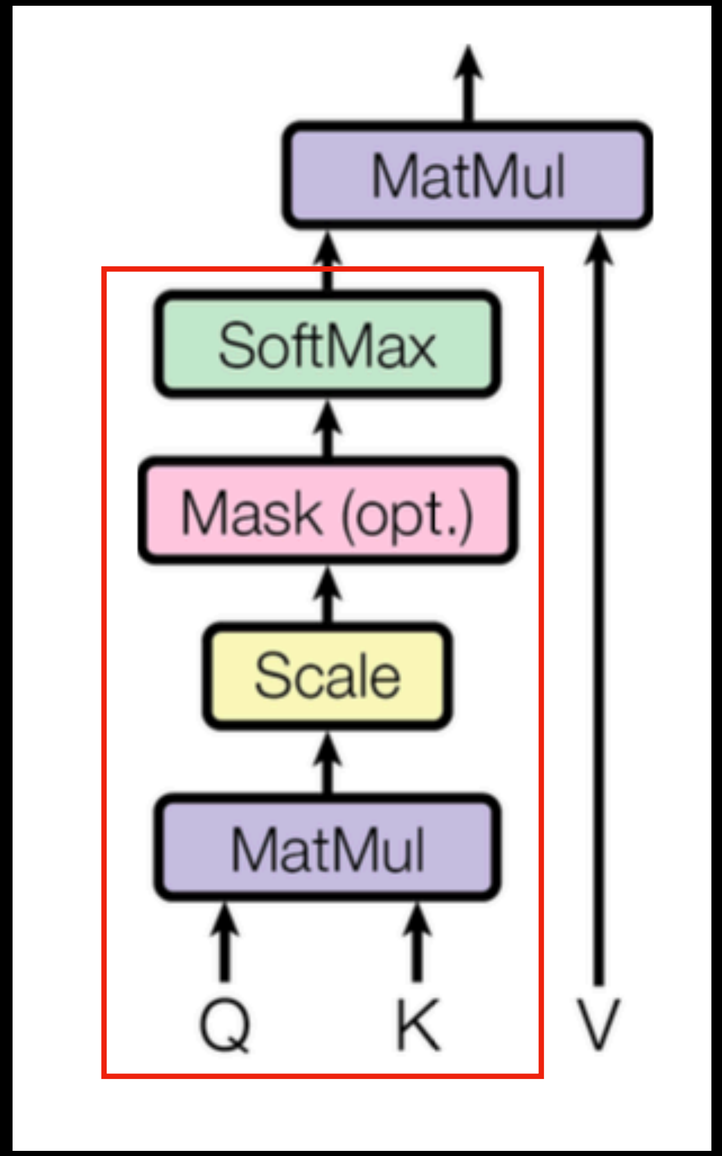
自注意力（可能的主谓关系）

$\overrightarrow{Q_3}$ 虫子	0.01	0.00	0.00
$\overrightarrow{Q_2}$ 是	0.99	0.00	0.00
$\overrightarrow{Q_1}$ 我	0.00	0.99	0.01
	我 $\overrightarrow{K_1}$	是 $\overrightarrow{K_2}$	虫子 $\overrightarrow{K_3}$



可能的词义注意力(概率)

$\overrightarrow{Q_3}$ 虫子	0.01	0.05	0.94
$\overrightarrow{Q_2}$ 是	0.00	0.95	0.05
$\overrightarrow{Q_1}$ 我	0.99	0.00	0.01
	I $\overrightarrow{K_1}$	am $\overrightarrow{K_2}$	bug $\overrightarrow{K_3}$



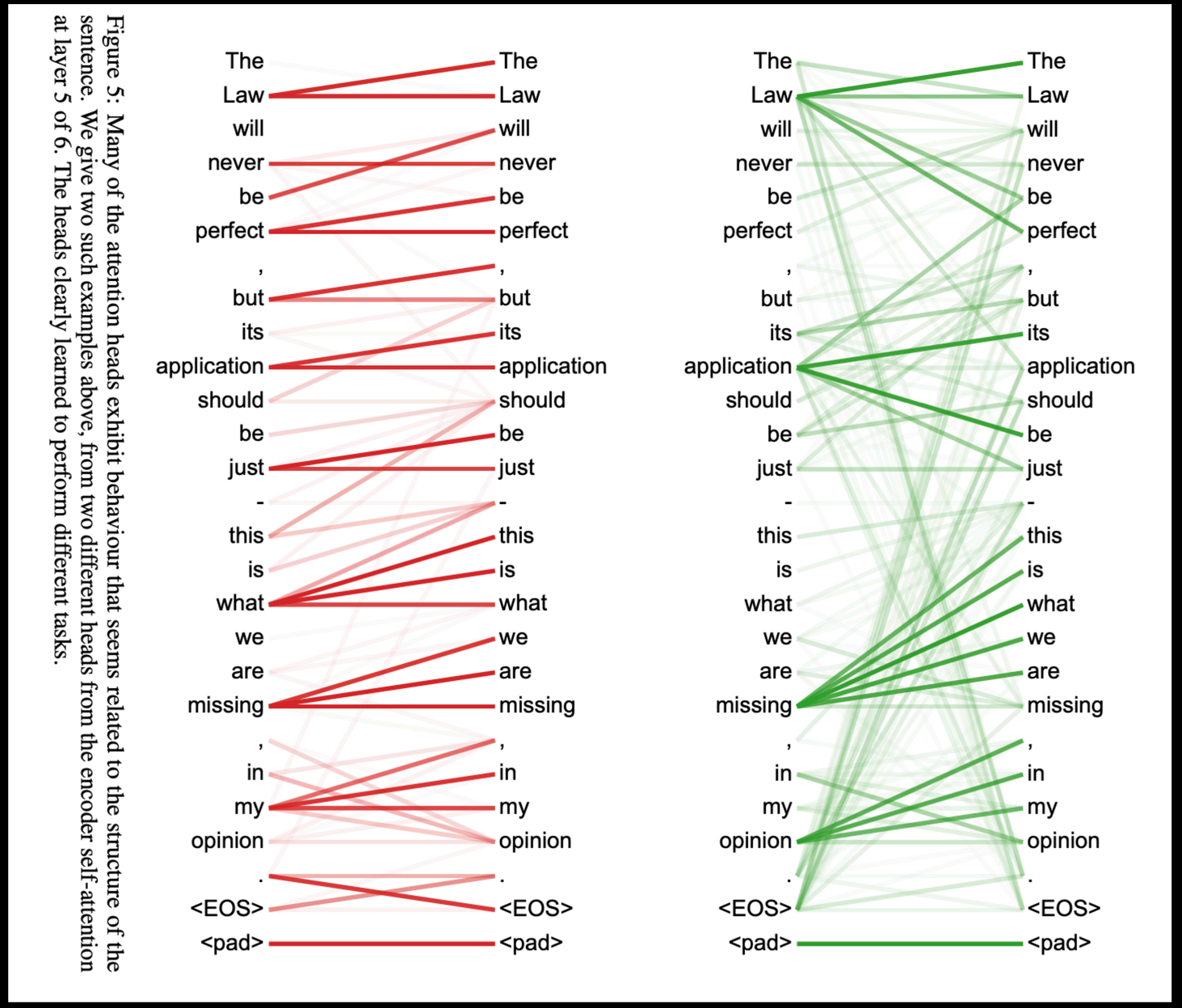


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

Transformer论文中模型展现的一些注意力模型