

The background is a dark blue gradient with a central illustration of a human brain. The brain is rendered in a realistic style with orange and yellow tones, but it is overlaid with a complex network of glowing blue and green circuit lines. These lines radiate from the brain and connect to various abstract digital icons scattered across the frame, including a gear-like hexagon on the left, a circular brain icon with yellow dots on the right, and several horizontal bars and squares. In the center, a white square frame contains the letters 'AI' in a bold, white, sans-serif font.

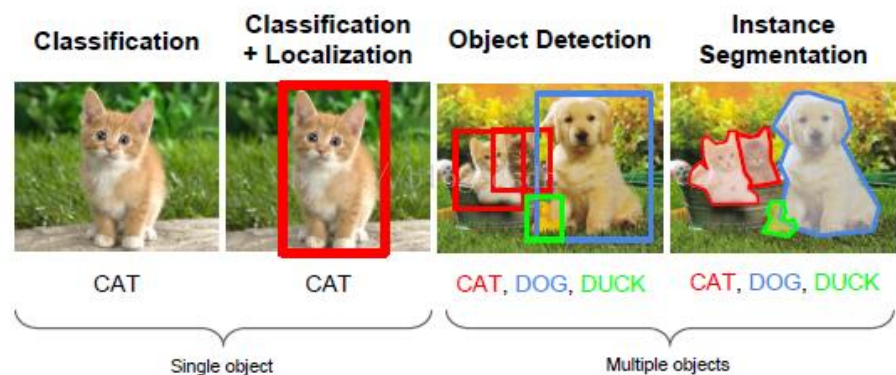
AI

人工神经网络与深度学习

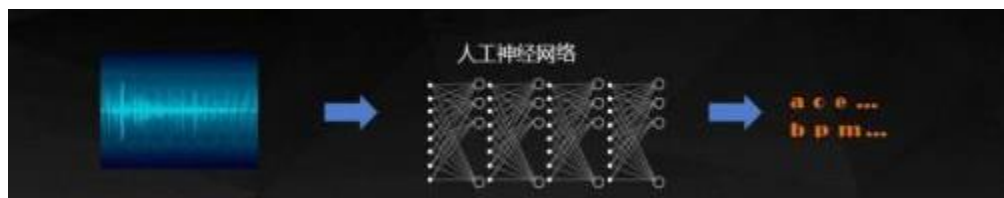
北京大学地空数算SESSDSA2019

深度学习应用场景

Computer Vision Tasks



计算机视觉



语音识别



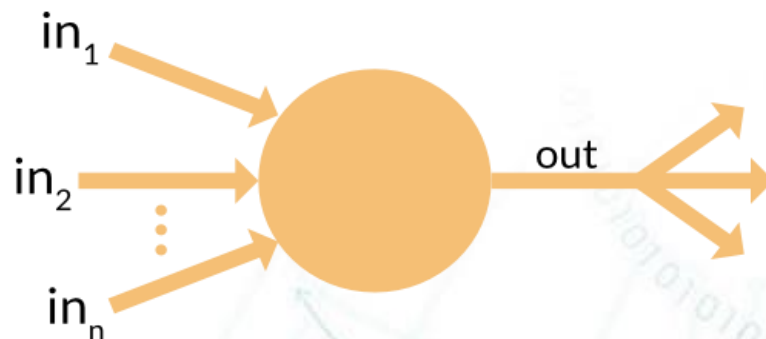
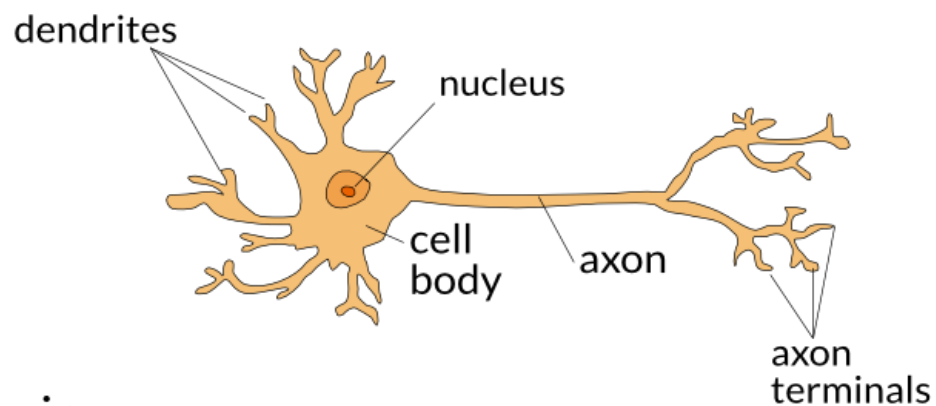
自然语言处理



人机博弈

神经元

- 是生物神经细胞的简单抽象。
- 神经细胞结构大致可分为：树突、突触、细胞体及轴突。
- 单个神经细胞可被视为一种只有两种状态的机器——激动时为‘是’，而未激动时为‘否’。

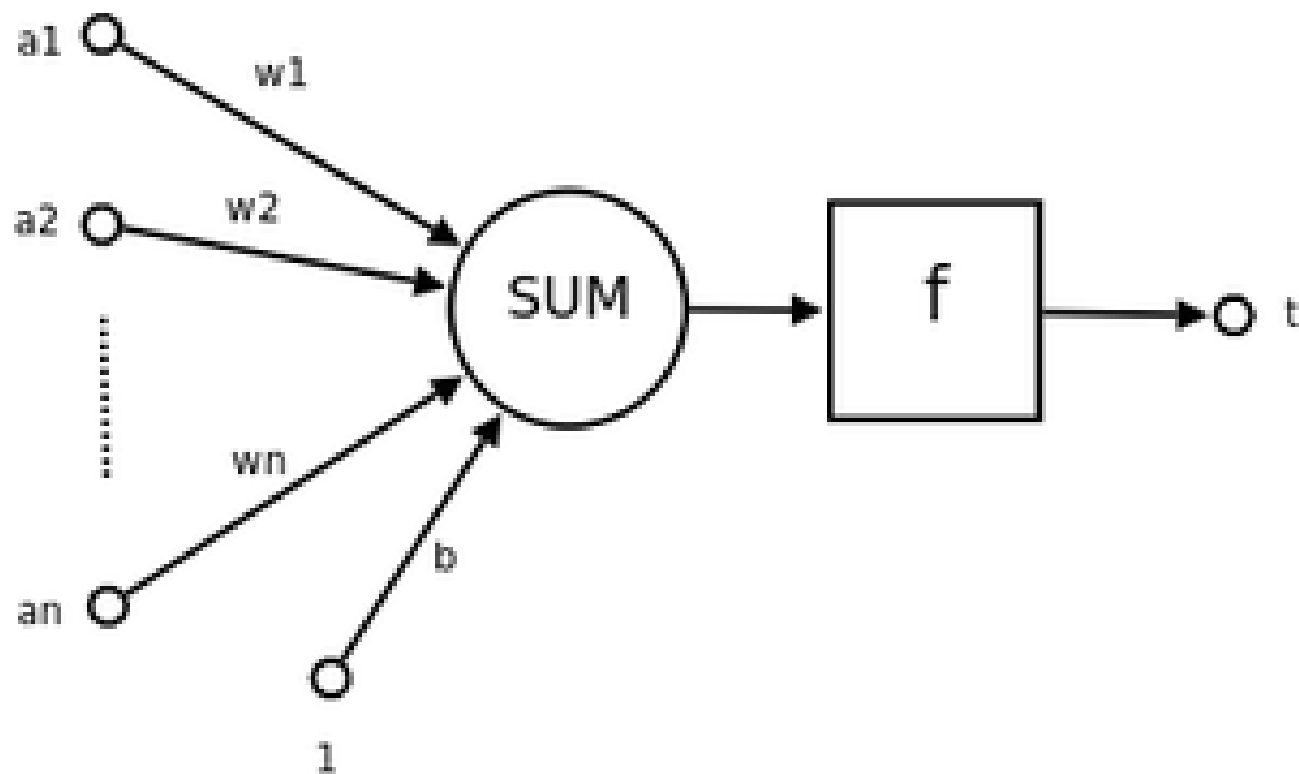


神经元

- 设 $\vec{a} = (a_1, a_2, \dots, a_n)$ 为这个输入的n维分量， $\vec{w} = (w_1, w_2, \dots, w_n)$ 为权重，
b为偏置量， $f(x)$ 为激活函数，即

- $$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

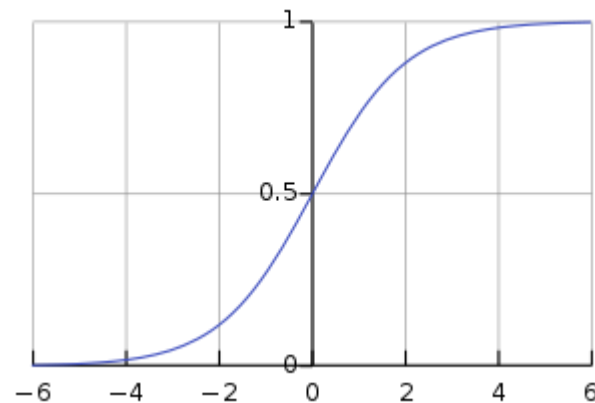
- 那么预测值就是 $\hat{y} = f(\vec{a} \cdot \vec{w} + b)$



激活函数

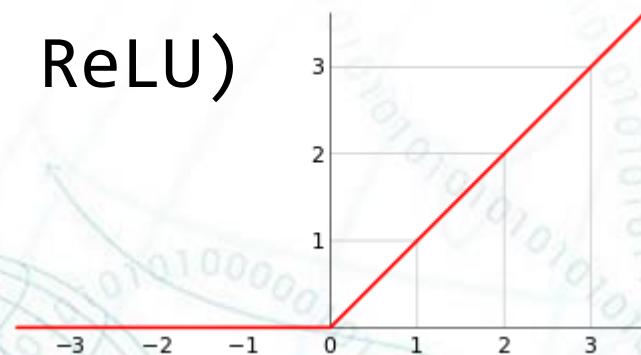
- Sigmoid function

- $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$

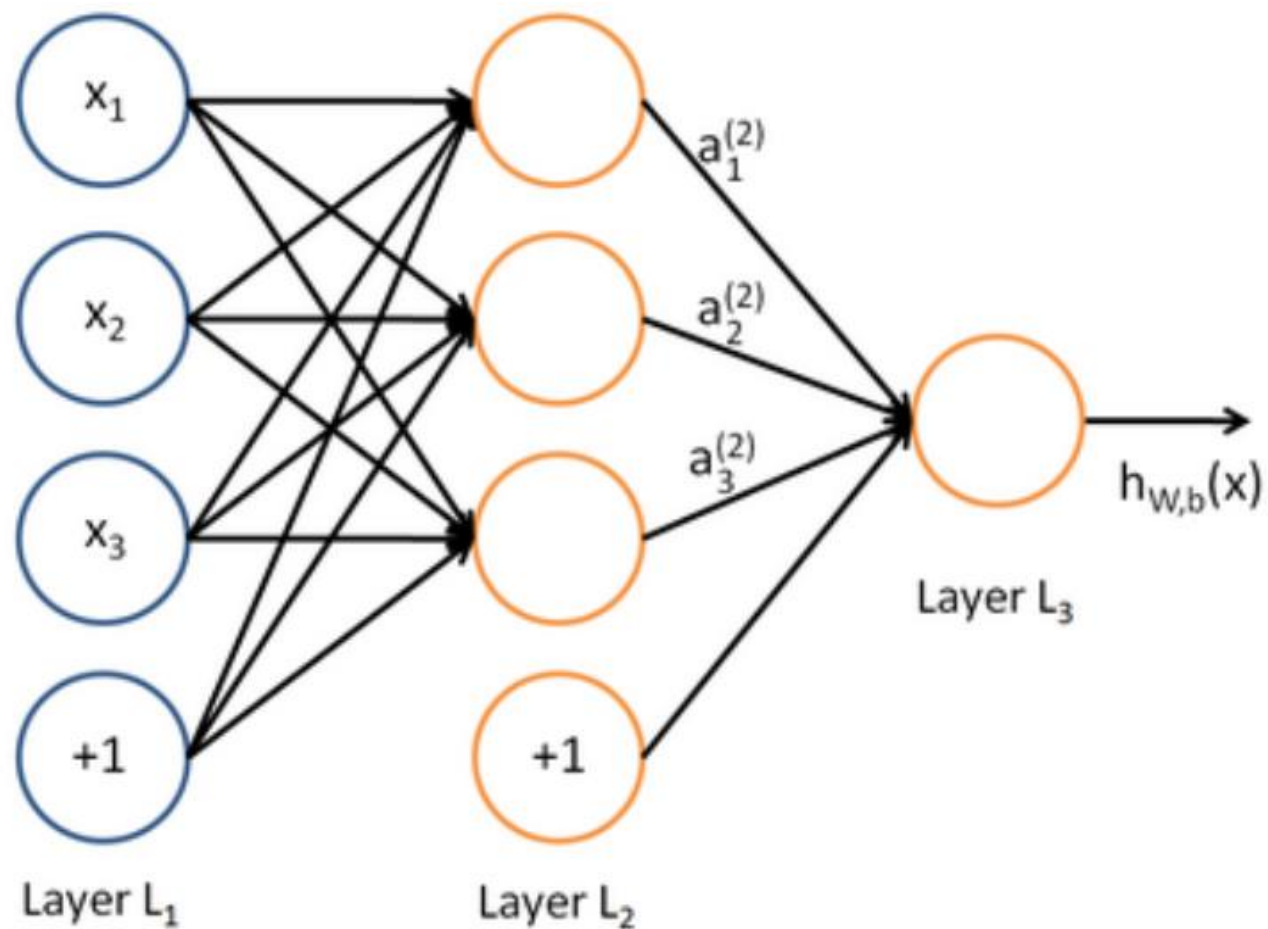


- 线性整流函数 (Rectified Linear Unit, ReLU)

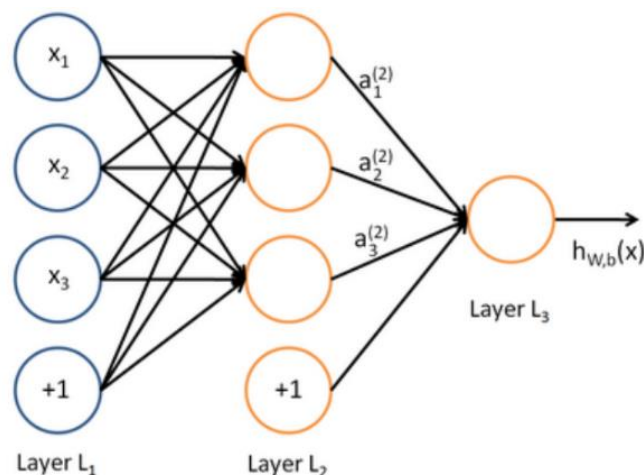
- $f(x) = \max(0, x)$



神经网络



前向传播



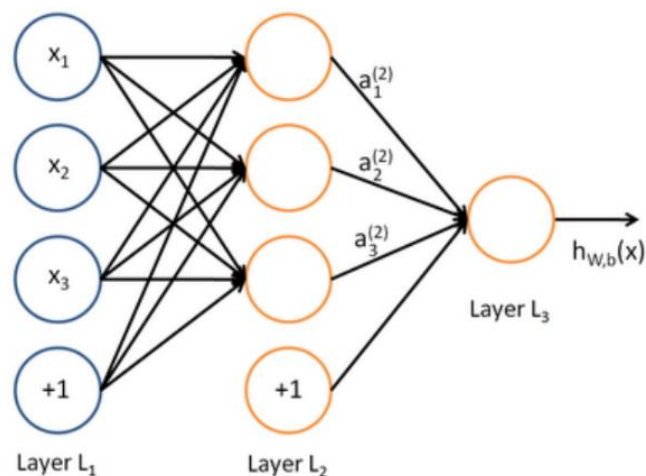
$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$

$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

前向传播



$$z^{(2)} = W^{(1)}x + b^{(1)}$$

$$a^{(2)} = f(z^{(2)})$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

损失函数

- 交叉熵 (cross entropy)
 - $\text{CE}(Y, f(x)) = -\sum_{x \in X} Y \log f(x)$
- 平方损失函数
 - $L(Y, f(x)) = \sum_{x \in X} (Y - f(x))^2$



反向传播

假设我们有一个固定样本集 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ ，它包含 m 个样例。我们可以用批量梯度下降法来求解神经网络。具体来讲，对于单个样例 (x, y) ，其代价函数为：

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2.$$

反向传播

给定一个包含 m 个样例的数据集，我们可以定义整体代价函数为：

$$\begin{aligned} J(W, b) &= \left[\frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \\ &= \left[\frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \end{aligned}$$

反向传播

梯度下降法中每一次迭代都按照如下公式对参数 W 和 b 进行更新：

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)$$

反向传播

首先看如何使用反向传播算法来计算 $\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y)$ 和 $\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y)$ ，这两项是单个样例 (x, y) 的代价函数

$J(W, b; x, y)$ 的偏导数。一旦我们求出该偏导数，就可以推导出整体代价函数 $J(W, b)$ 的偏导数：

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) = \left[\frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x^{(i)}, y^{(i)}) \right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b) = \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b_i^{(l)}} J(W, b; x^{(i)}, y^{(i)})$$

反向传播

1. 进行前馈传导计算，利用前向传导公式，得到 $L_2, L_3, \dots$ 直到输出层 L_{n_l} 的激活值。
2. 对于第 n_l 层（输出层）的每个输出单元 i ，我们根据以下公式计算残差：

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

$$\begin{aligned} \delta_i^{(n_l)} &= \frac{\partial}{\partial z_i^{n_l}} J(W, b; x, y) = \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 \\ &= \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - a_j^{(n_l)})^2 = \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - f(z_j^{(n_l)}))^2 \\ &= -(y_i - f(z_i^{(n_l)})) \cdot f'(z_i^{(n_l)}) = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \end{aligned}$$

反向传播

3. 对 $l = n_l - 1, n_l - 2, n_l - 3, \dots, 2$ 的各个层, 第 l 层的第 i 个节点的残差计算方法如下:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$



反向传播

$$\begin{aligned}\delta_i^{(n_l-1)} &= \frac{\partial}{\partial z_i^{n_l-1}} J(W, b; x, y) = \frac{\partial}{\partial z_i^{n_l-1}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = \frac{\partial}{\partial z_i^{n_l-1}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - a_j^{(n_l)})^2 \\&= \frac{1}{2} \sum_{j=1}^{S_{n_l}} \frac{\partial}{\partial z_i^{n_l-1}} (y_j - a_j^{(n_l)})^2 = \frac{1}{2} \sum_{j=1}^{S_{n_l}} \frac{\partial}{\partial z_i^{n_l-1}} (y_j - f(z_j^{(n_l)}))^2 \\&= \sum_{j=1}^{S_{n_l}} -(y_j - f(z_j^{(n_l)})) \cdot \frac{\partial}{\partial z_i^{(n_l-1)}} f(z_j^{(n_l)}) = \sum_{j=1}^{S_{n_l}} -(y_j - f(z_j^{(n_l)})) \cdot f'(z_j^{(n_l)}) \cdot \frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}} \\&= \sum_{j=1}^{S_{n_l}} \delta_j^{(n_l)} \cdot \frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}} = \sum_{j=1}^{S_{n_l}} \left(\delta_j^{(n_l)} \cdot \frac{\partial}{\partial z_i^{n_l-1}} \sum_{k=1}^{S_{n_l-1}} f(z_k^{n_l-1}) \cdot W_{jk}^{n_l-1} \right) \\&= \sum_{j=1}^{S_{n_l}} \delta_j^{(n_l)} \cdot W_{ji}^{n_l-1} \cdot f'(z_i^{n_l-1}) = \left(\sum_{j=1}^{S_{n_l}} W_{ji}^{n_l-1} \delta_j^{(n_l)} \right) f'(z_i^{n_l-1})\end{aligned}$$

将上式中的 $n_l - 1$ 与 n_l 的关系替换为 l 与 $l + 1$ 的关系，就可以得到：

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

以上逐次从后向前求导的过程即为“反向传导”的本意所在。

反向传播

4. 计算我们需要的偏导数，计算方法如下：

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}.$$

反向传播

求 $\frac{\partial}{\partial W_{ij}^{(l)}} J(W)$?

问题拆解: $\frac{\partial}{\partial W_{ij}^{(l)}} J(W) = \frac{\partial J(W)}{\partial Z_i^{(l+1)}} * \frac{\partial Z_i^{(l+1)}}{\partial W_{ij}^{(l)}}$

神经元求和: $Z_i^{(l+1)} = \sum_j^n W_{ij}^{(l)} * a_j^{(l)}$

输出对权值的偏导数: blog.csdn.net/u014403897

$$\frac{\partial Z_i^{(l+1)}}{\partial W_{ij}^{(l)}} = \frac{\partial \sum_j^n W_{ij}^{(l)} * a_j^{(l)}}{\partial W_{ij}^{(l)}} = a_j^{(l)}$$

设神经元的错误变化率为:

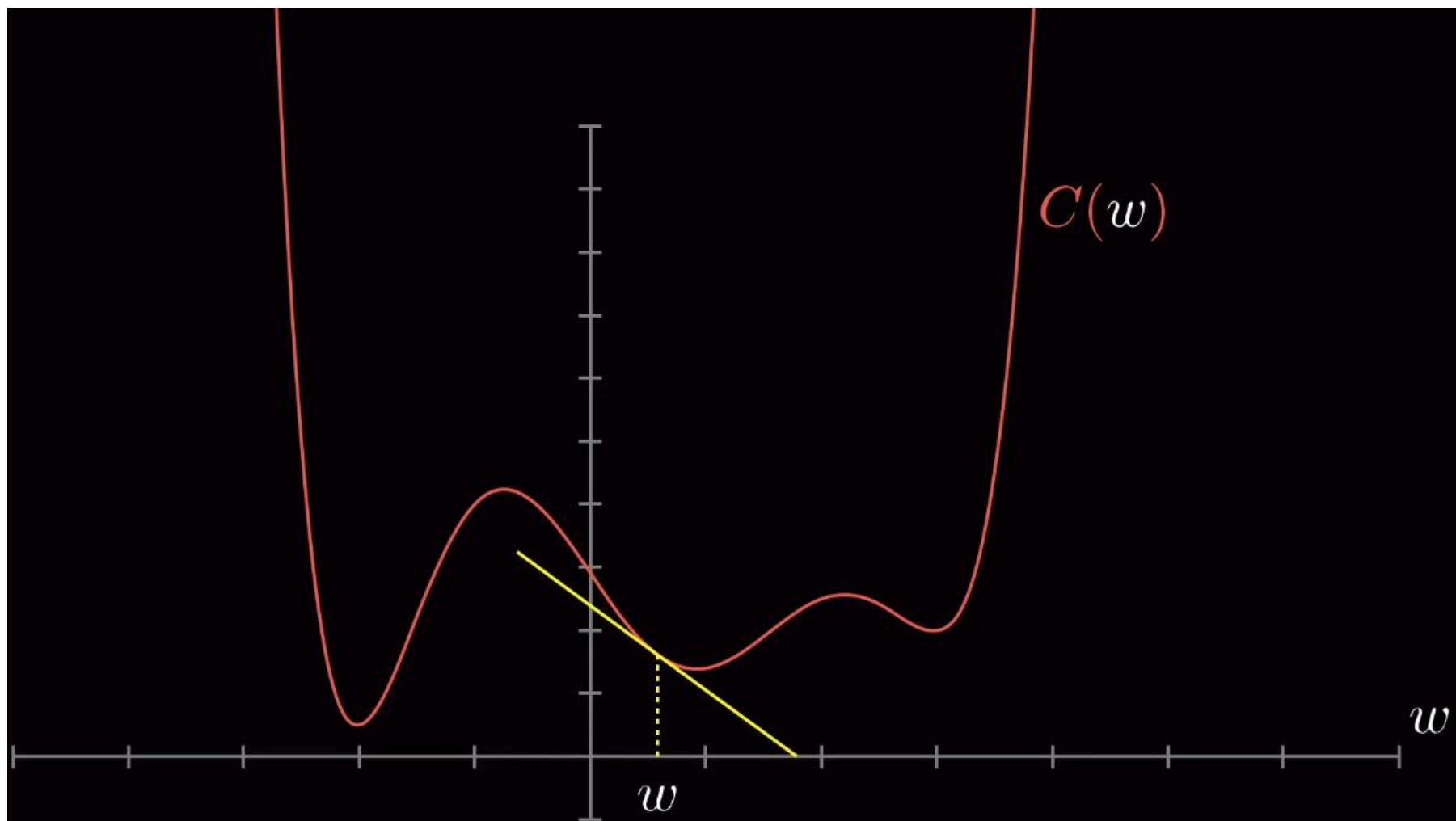
最终:

$$\delta_i^{l+1} = \frac{\partial J(W)}{\partial Z_i^{(l+1)}}$$

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W) = \delta_i^{l+1} * a_j^{(l)}$$

优化算法

- 梯度下降算法
 - 计算梯度
 - $f'(x)$
 - 变量 x 向梯度反方向移动
 - $x = x - r * f'(x)$
 - 循环直至满足条件
- Adam, RMSProp 算法



深度学习流行框架：tensorflow

基于数据流图的数值计算开源软件库。数据流图中的点表示数学计算操作，边表示操作与操作之间的高维度数组数据，称为tensor。

深度学习的hello world：MNIST手写数字识别



28

28



784

输入层

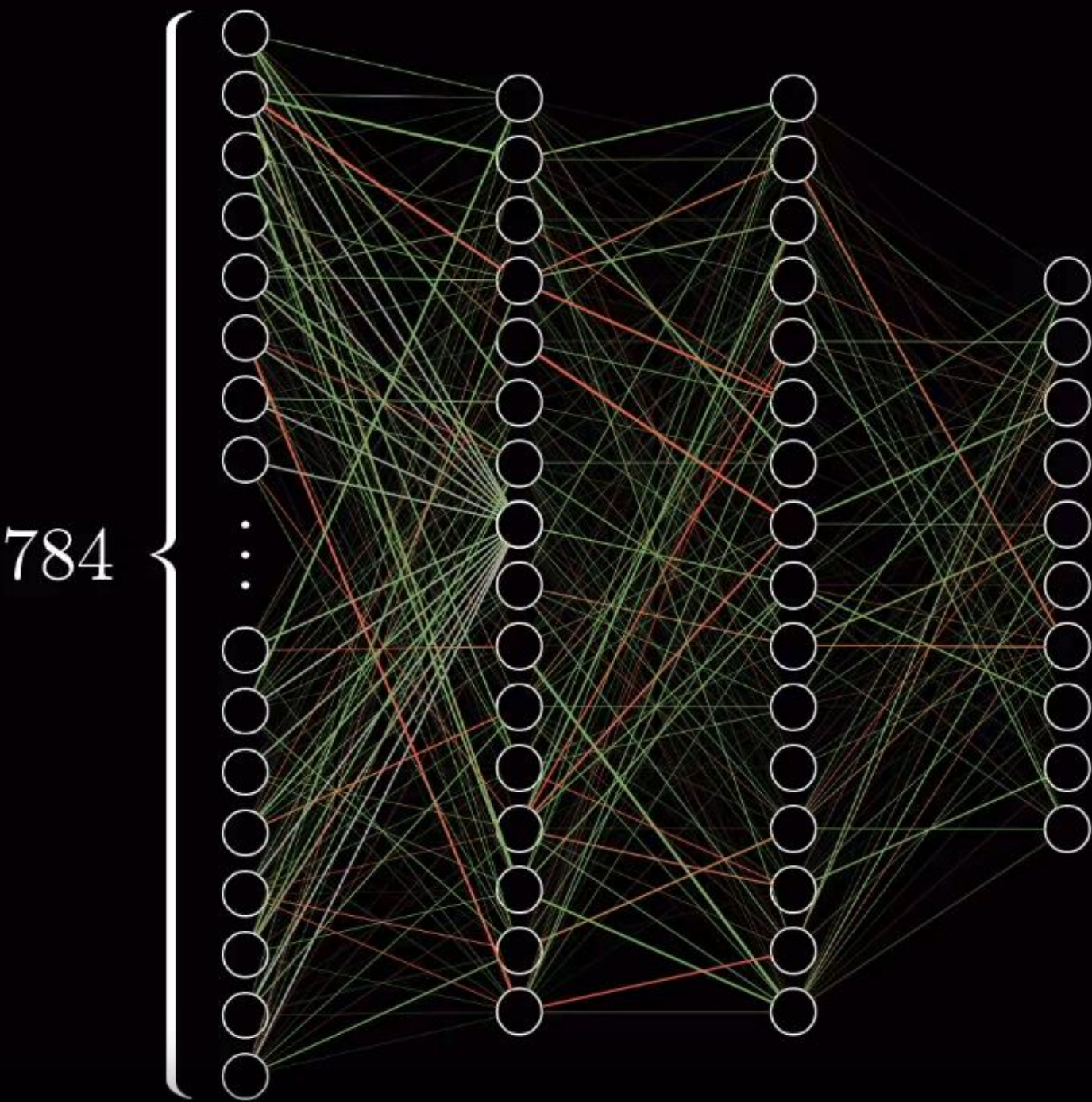
隐藏层1

隐藏层2

输出层

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9





$$784 \times 16 + 16 \times 16 + 16 \times 10$$

weights

$$16 + 16 + 10$$

biases

13,002

Learning $\rightarrow$ Finding the right weights and biases

Tensorflow:MNIST识别手写数字

```
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True) # 读取MNIST手写数字识别数据集
import tensorflow as tf

X = tf.placeholder(tf.float32, [None, 784], name='x-input') # tensorflow变量占位符, 在执行运算时才传入数据
Y = tf.placeholder(tf.float32, [None, 10], name='y-input')

def init_weights(shape): # 使用服从正态分布的随机值来初始化参数矩阵
    return tf.Variable(tf.random_normal(shape, stddev=0.01))

w_h = init_weights([784, 16]) # 从输入层到hidden layer1, 28x28=784维转化为16维
w_h2 = init_weights([16, 16]) # 从hidden layer1到hidden layer2, 16维转化为16维
w_o = init_weights([16, 10]) # 从hidden layer2到输出层, 16维转化为10维
```

Tensorflow:MNIST识别手写数字

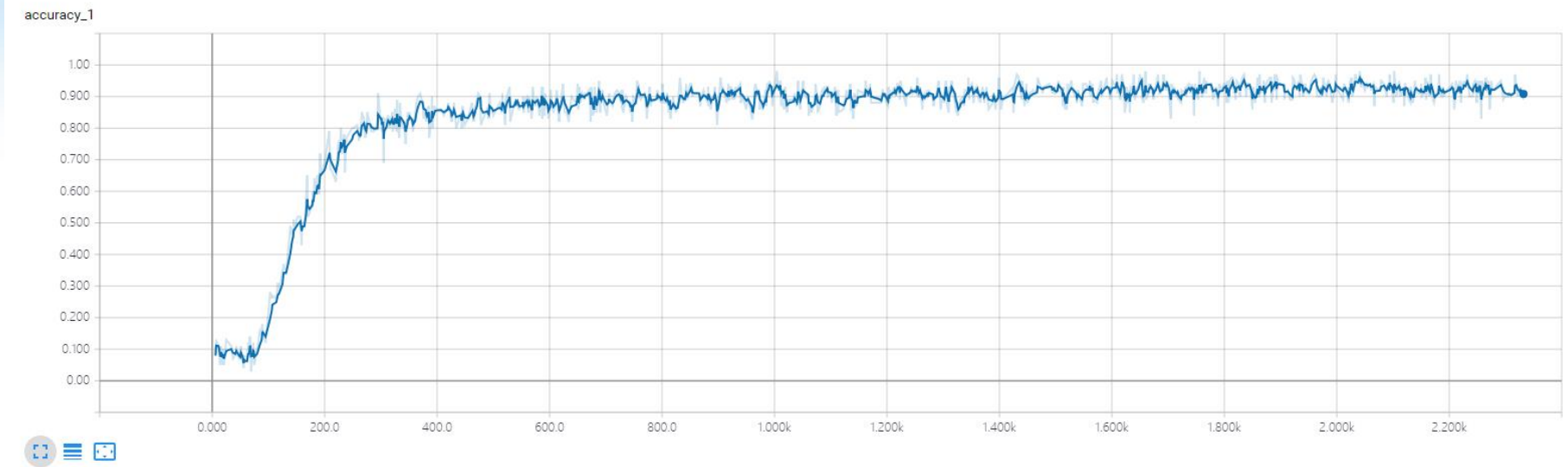
```
def model(X, w_h, w_h2, w_o): # 构建神经网络训练模型
    h = tf.nn.relu(tf.matmul(X, w_h)) # hidden layer1是由输入层和权重矩阵1进行矩阵乘法后经过激活函数得到
    h2 = tf.nn.relu(tf.matmul(h, w_h2)) # hidden layer2是由hidden layer1和权重矩阵2进行矩阵乘法后经过激活函数得到
    return tf.matmul(h2, w_o) # 返回hidden layer2和权重矩阵3的矩阵乘法, 即输出层

py_x = model(X, w_h, w_h2, w_o) # 初始化一个神经网络训练模型
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=py_x, labels=Y)) # 设置模型的代价函数
train_op = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost) # 设置优化算法, 这里是改进后的梯度下降算法
predict_acc = tf.reduce_mean(tf.cast(tf.equal(tf.argmax(py_x, 1), tf.argmax(Y, 1)), tf.float32)) # 计算预测的准确率
epoch_count = 2333 # 需要计算的次数
batch_size = 64 # 每一批的图片数量
```

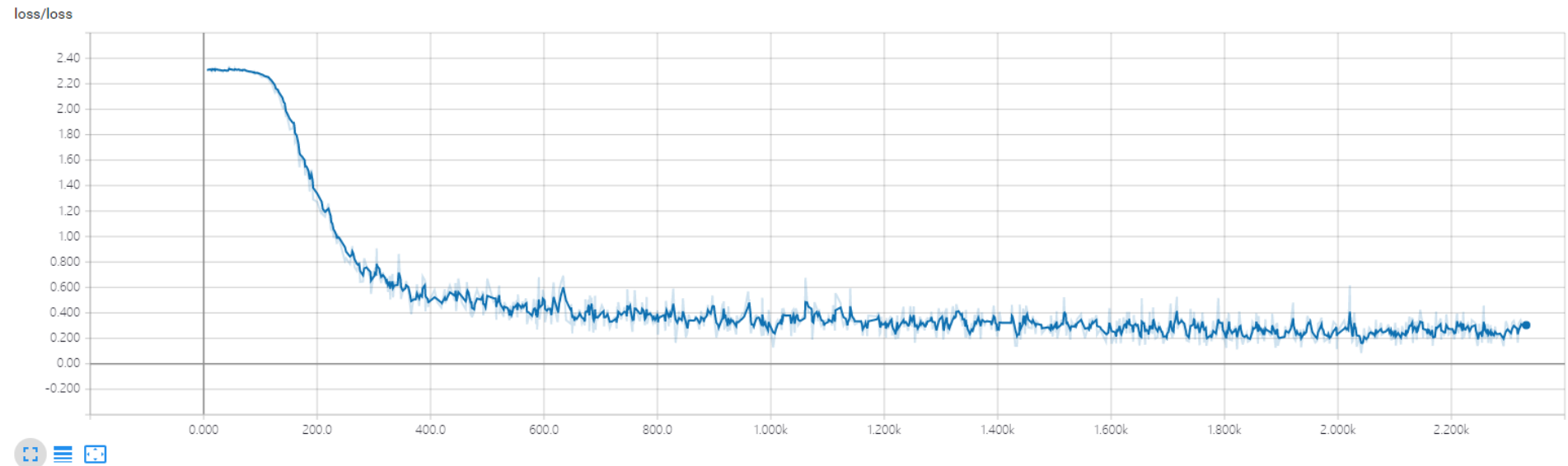

Tensorflow:MNIST识别手写数字

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer()) # 初始化各种变量
    step = 0
    for i in range(epoch_count):
        step += 1
        batch_x, batch_y = mnist.train.next_batch(batch_size) # 读取相应一批的图片和标签数量
        sess.run(train_op, feed_dict={X: batch_x, Y: batch_y}) # 通过神经网络训练
        if step % 128 == 0: # 输出训练记录
            loss, acc = sess.run([cost, predict_acc],
                                feed_dict={X: batch_x, Y: batch_y})
            print("Epoch: {}".format(step), "\tLoss: {:.6f}".format(loss), "\tTraining Accuracy: {:.5f}".format(acc))
    print("Testing Accuracy: {:.0.5f}".format(sess.run(predict_acc,
                                                         feed_dict={X: mnist.test.images, Y: mnist.test.labels})))
```


Accuracy at step 0: 0.0819
Accuracy at step 128: 0.3761
Accuracy at step 256: 0.7901
Accuracy at step 384: 0.8468
Accuracy at step 512: 0.8702
Accuracy at step 640: 0.886
Accuracy at step 768: 0.8964
Accuracy at step 896: 0.9032
Accuracy at step 1024: 0.9078
Accuracy at step 1152: 0.9036
Accuracy at step 1280: 0.9111
Accuracy at step 1408: 0.9116
Accuracy at step 1536: 0.9141
Accuracy at step 1664: 0.9194
Accuracy at step 1792: 0.9227
Accuracy at step 1920: 0.9265
Accuracy at step 2048: 0.9275
Accuracy at step 2176: 0.9284
Accuracy at step 2304: 0.9313



loss



网络结构

- 卷积神经网络 (CNN)
 - 全连接层、卷积层、池化层
- 循环神经网络 (RNN)
 - 专门处理序列化数据
 - 长短期记忆网络 (LSTM)
- CNN+RNN
 - 根据图像语义自动生成摘要

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



The background features a stylized human brain in profile, facing left. The brain is rendered in a realistic brownish-tan color but is overlaid with a complex network of glowing blue and green circuit lines. These lines radiate from the brain and extend across the frame, connecting to various abstract digital icons. On the right side, there is a circular icon containing a brain-like shape made of yellow dots. On the left, there are several small square icons and a gear-like symbol. The overall color palette is dominated by dark blues, greens, and yellows, creating a high-tech, futuristic atmosphere.

AI

谢谢

Thanks for watching