基于TensorFlow和迁移学习构建图像识别应用

何辉辉 才云科技

©版权归才云所有，违反必究

提纲

01 深度学习与TensorFlow

02 卷积神经网络

03 迁移学习

04 构建图像识别应用

caicloud 才云
©版权归才云所有，违反必究

提纲

01 深度学习与TensorFlow

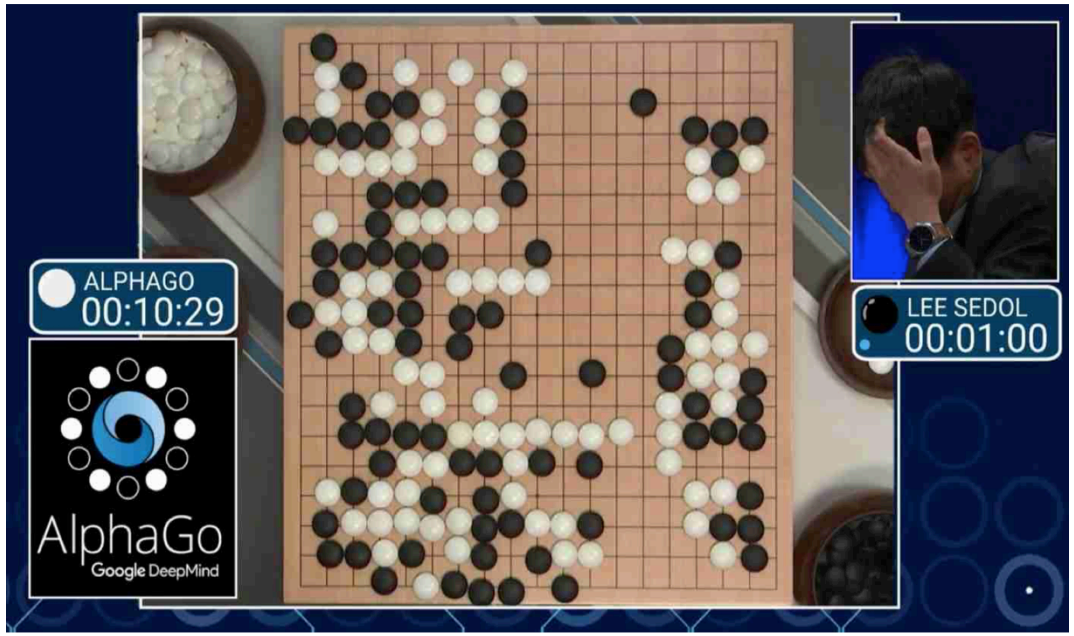
02 卷积神经网络

03 迁移学习

04 构建图像识别应用

caicloud 才云
©版权归才云所有，违反必究

深度学习的应用



语音

图像

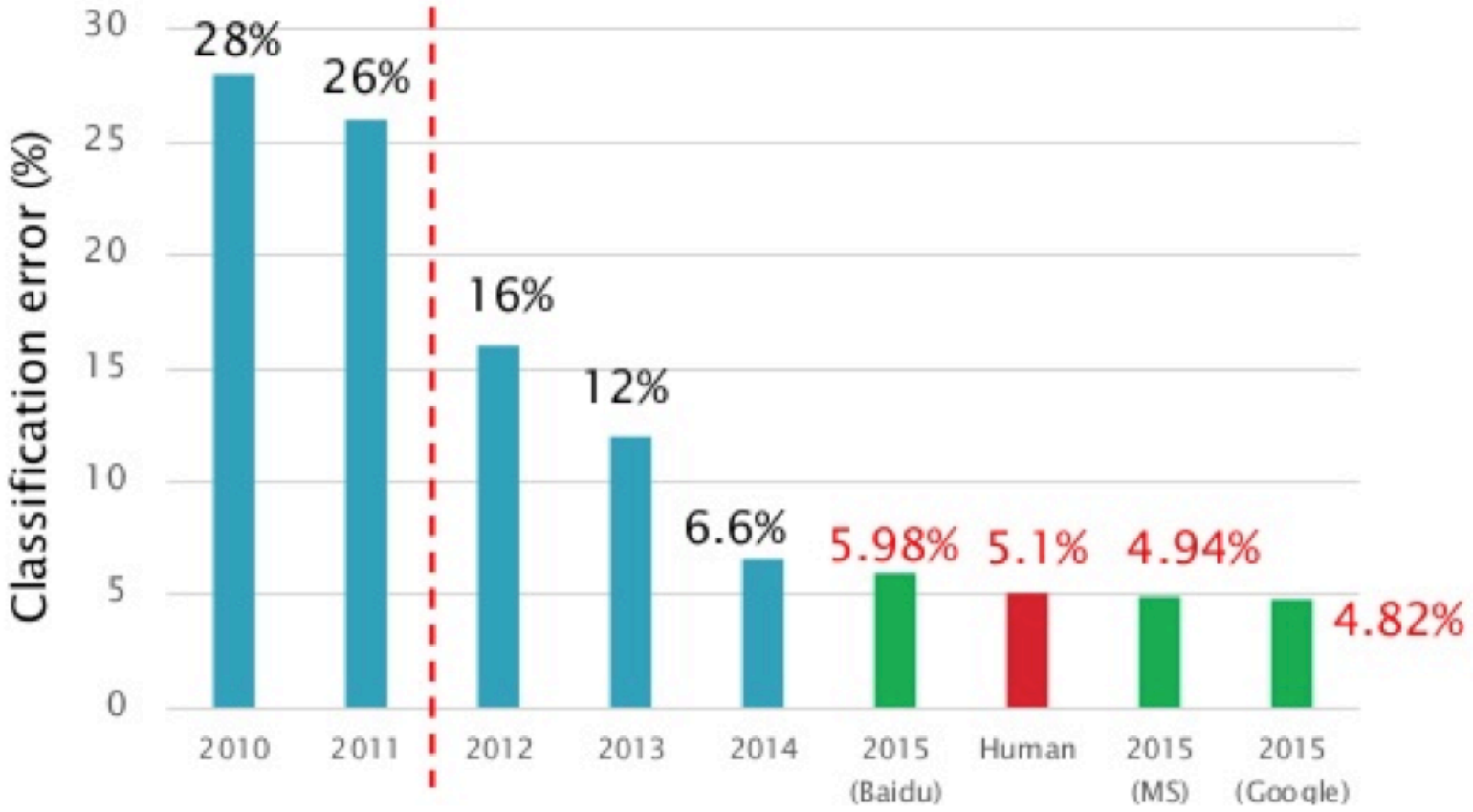
视频

caicloud才云
©版权归才云所有，违反必究

深度学习的效果

深度学习之前　　深度学习之后

IMAGENET



caicloud 才云
©版权归才云所有，违反必究

TensorFlow深度学习框架

ARM

quantiphi

AIRBUS DEFENCE & SPACE

CIST

CEVA

Google

Movidius

UBER

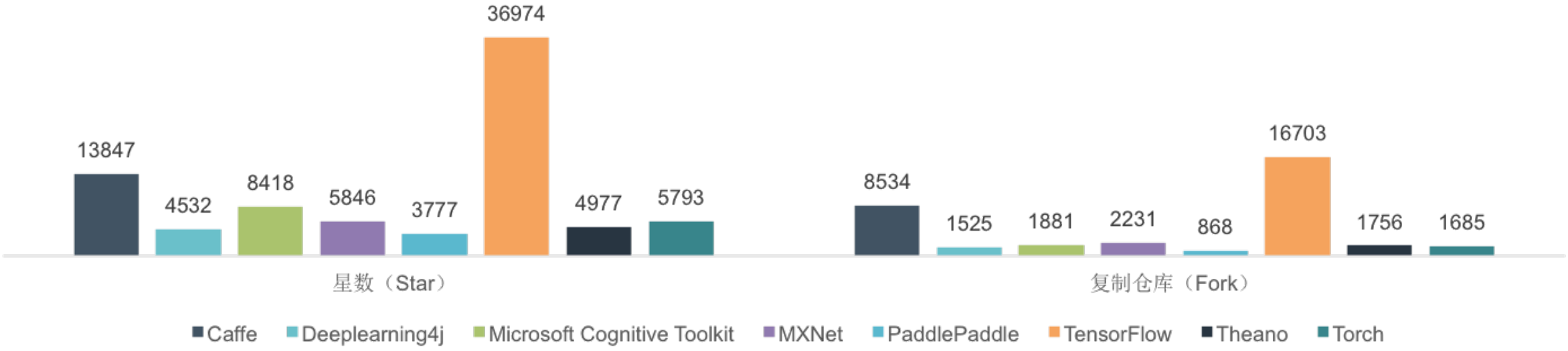
JD.COM 京东

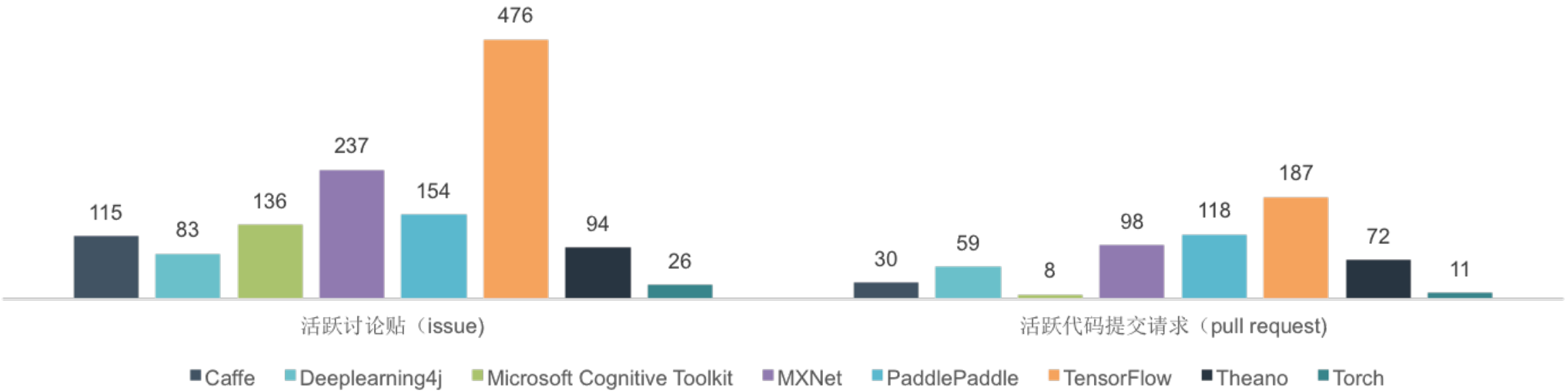
DeepMind

caicloud才云

©版权归才云所有，违反必究

深度学习框架对比





©版权归才云所有，违反必究

提纲

01 深度学习与TensorFlow

02 卷积神经网络

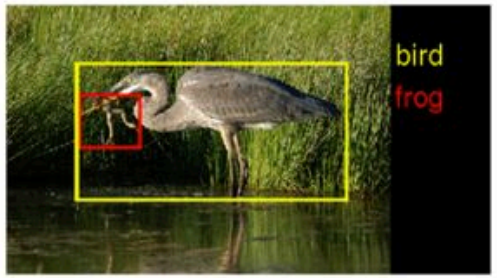
03 迁移学习

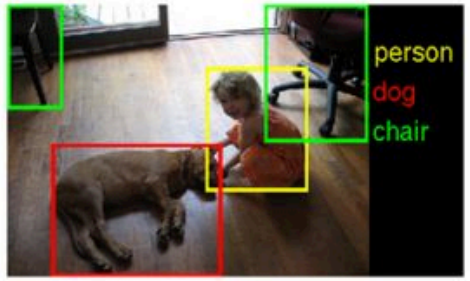
04 构建图像识别应用

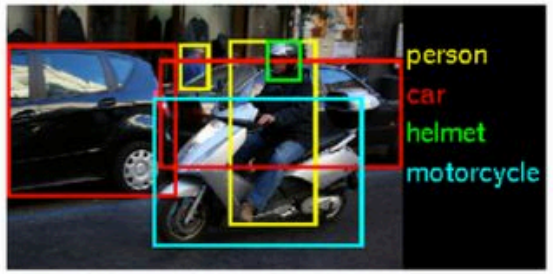
caicloud才云
©版权归才云所有，违反必究

图像识别的难点

ImageNet图像示例







计算机看到的图像

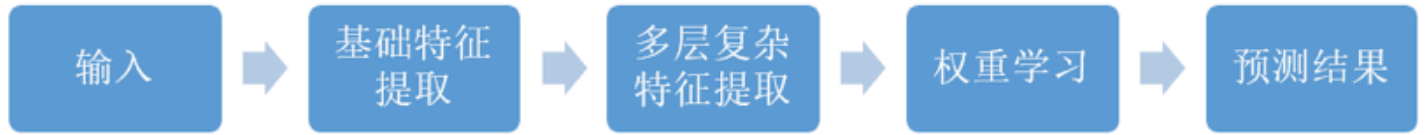
210	201	212	199	213	215	195	178	158	182	209
189	190	221	209	205	191	167	147	115	129	163
126	140	188	176	165	152	140	170	106	78	88
103	115	154	143	142	149	153	173	101	57	57
112	106	131	122	138	152	147	128	84	58	66
95	79	104	105	124	129	113	107	87	69	67
71	69	98	89	92	98	95	89	88	76	67
56	68	99	63	45	60	82	58	76	75	65
43	69	75	56	41	51	73	55	70	63	44
50	57	69	75	75	73	74	53	68	59	37
59	53	66	84	92	84	74	57	72	63	42
61	58	65	75	78	76	73	59	75	69	50

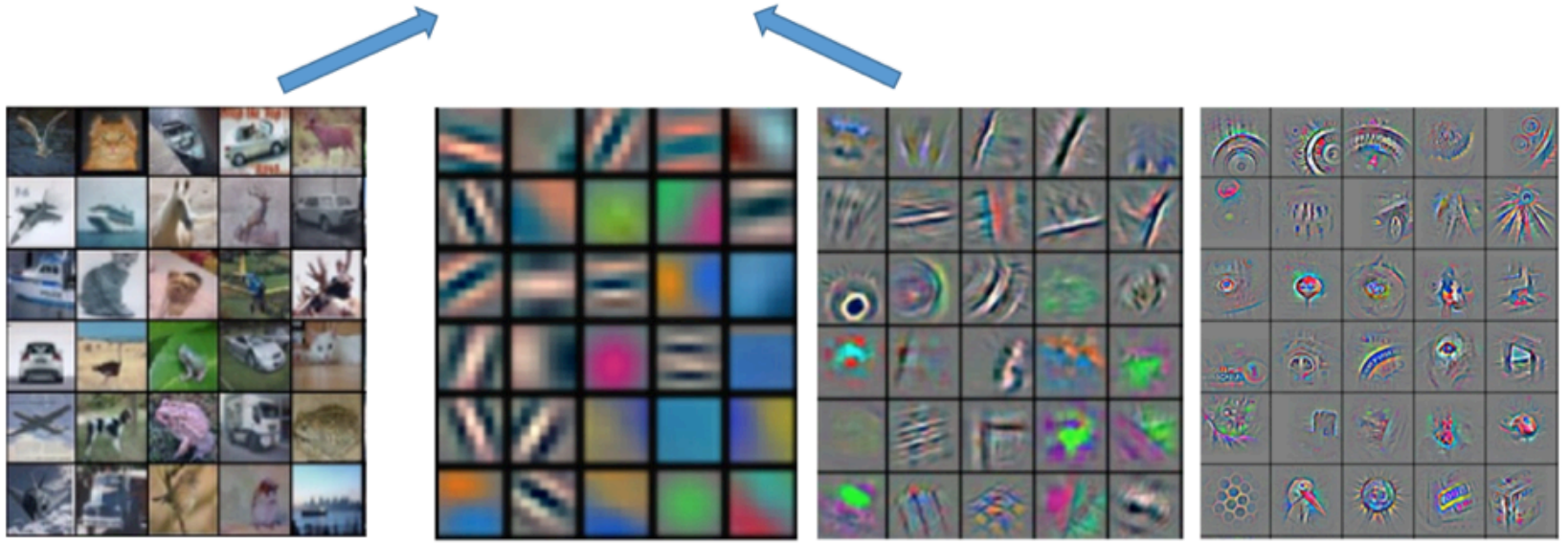
caicloud才云
©版权归才云所有，违反必究

图像识别方法

传统机器学习算法

深度学习算法

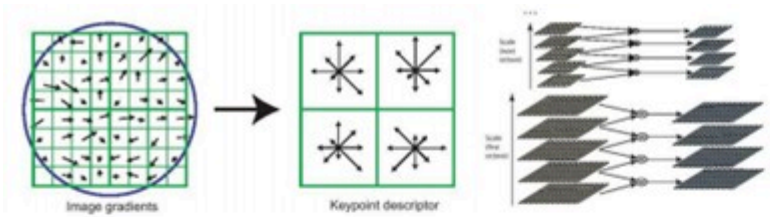




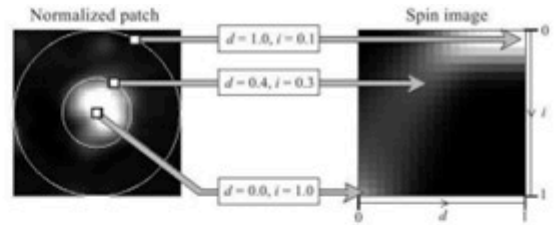
基础特征：图片像素　　第一层：线条　　第二层：简单形状　　第三层：复杂形状

caicloud 才云
©版权归才云所有，违反必究

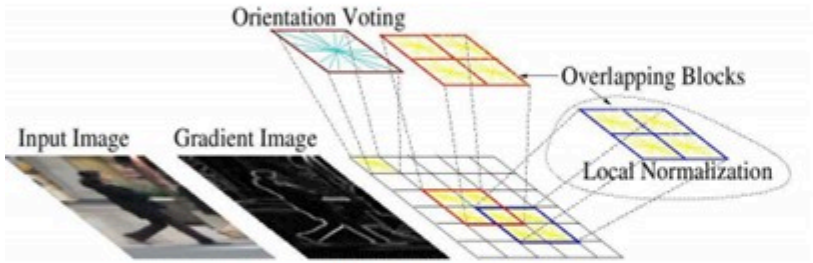
传统的特征提取方法



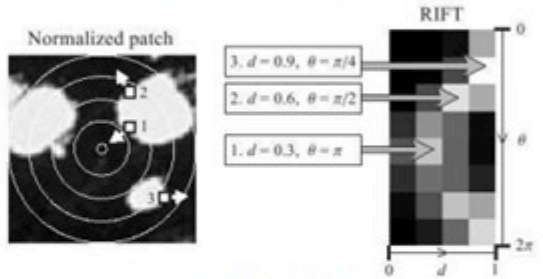
SIFT



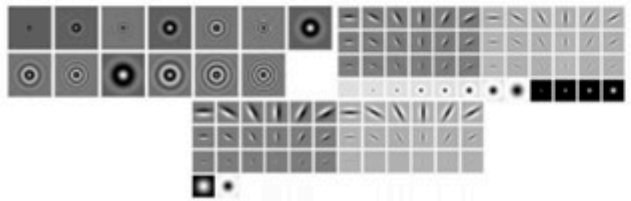
Spin image



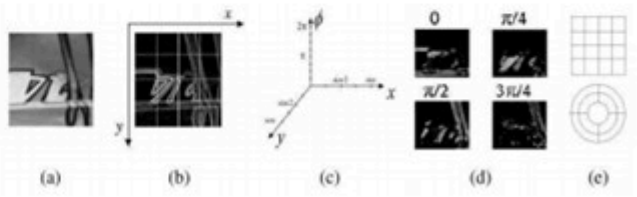
HoG



RIFT



Textons

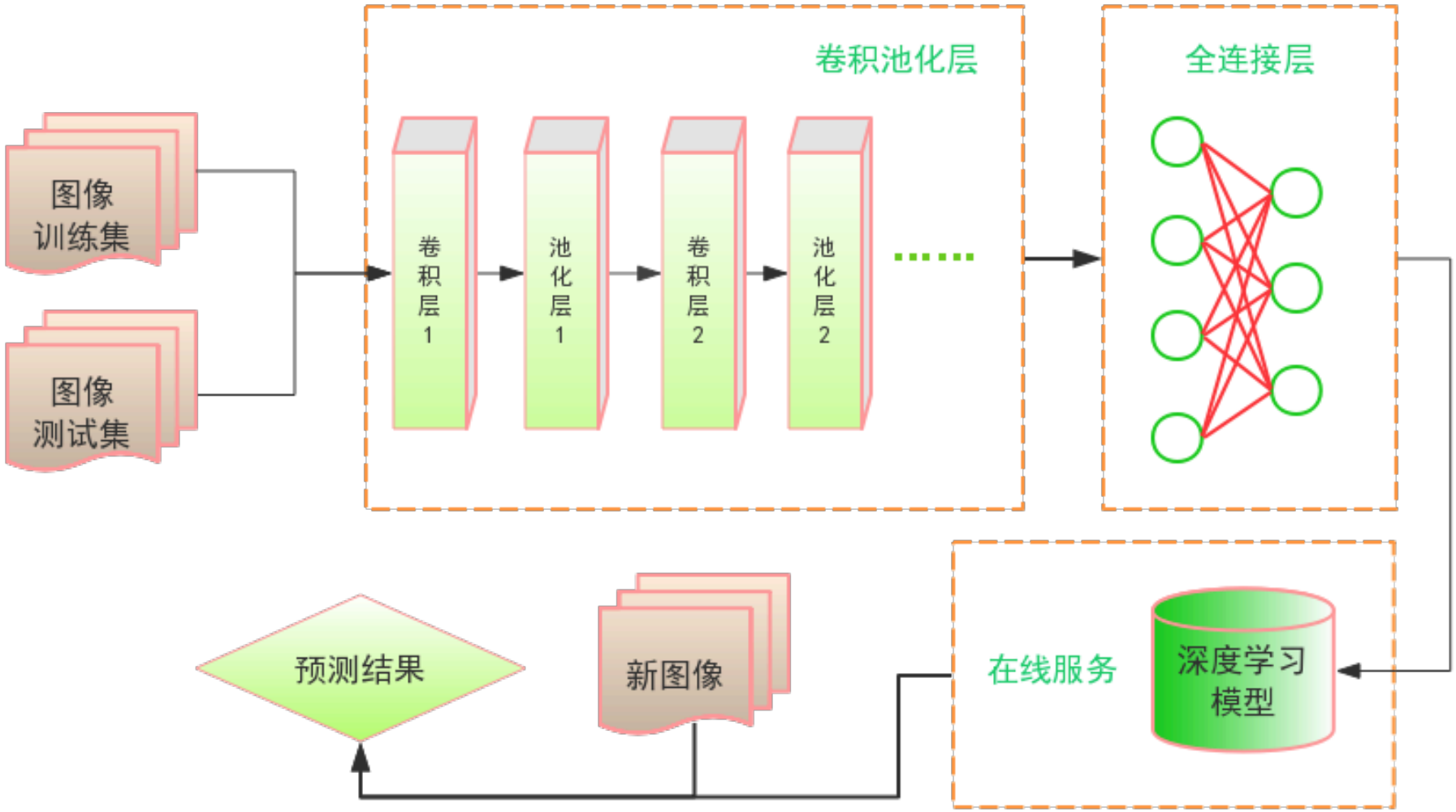


GLOH

- 费时费力
- 启发式
- 需要经验
- 效果不稳定

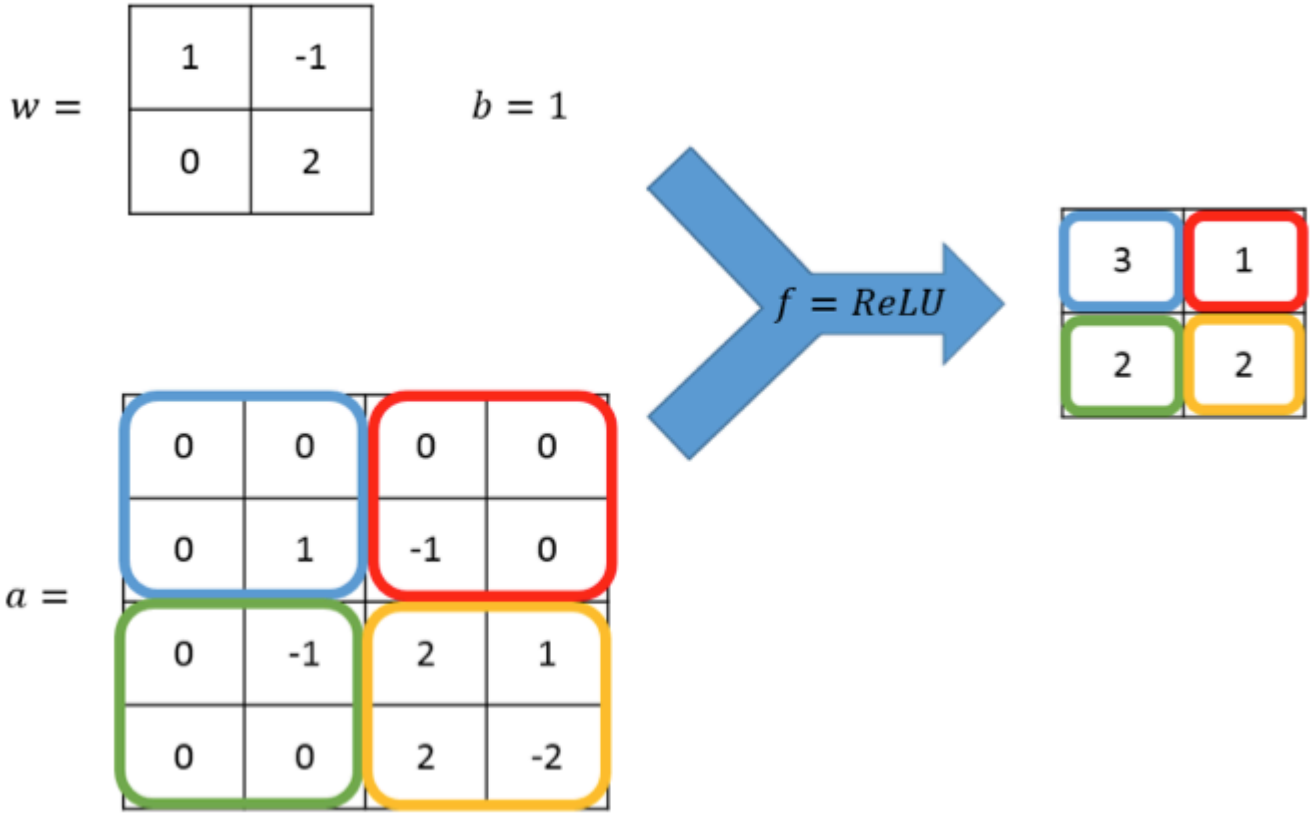
caicloud 才云
©版权归才云所有，违反必究

卷积神经网络



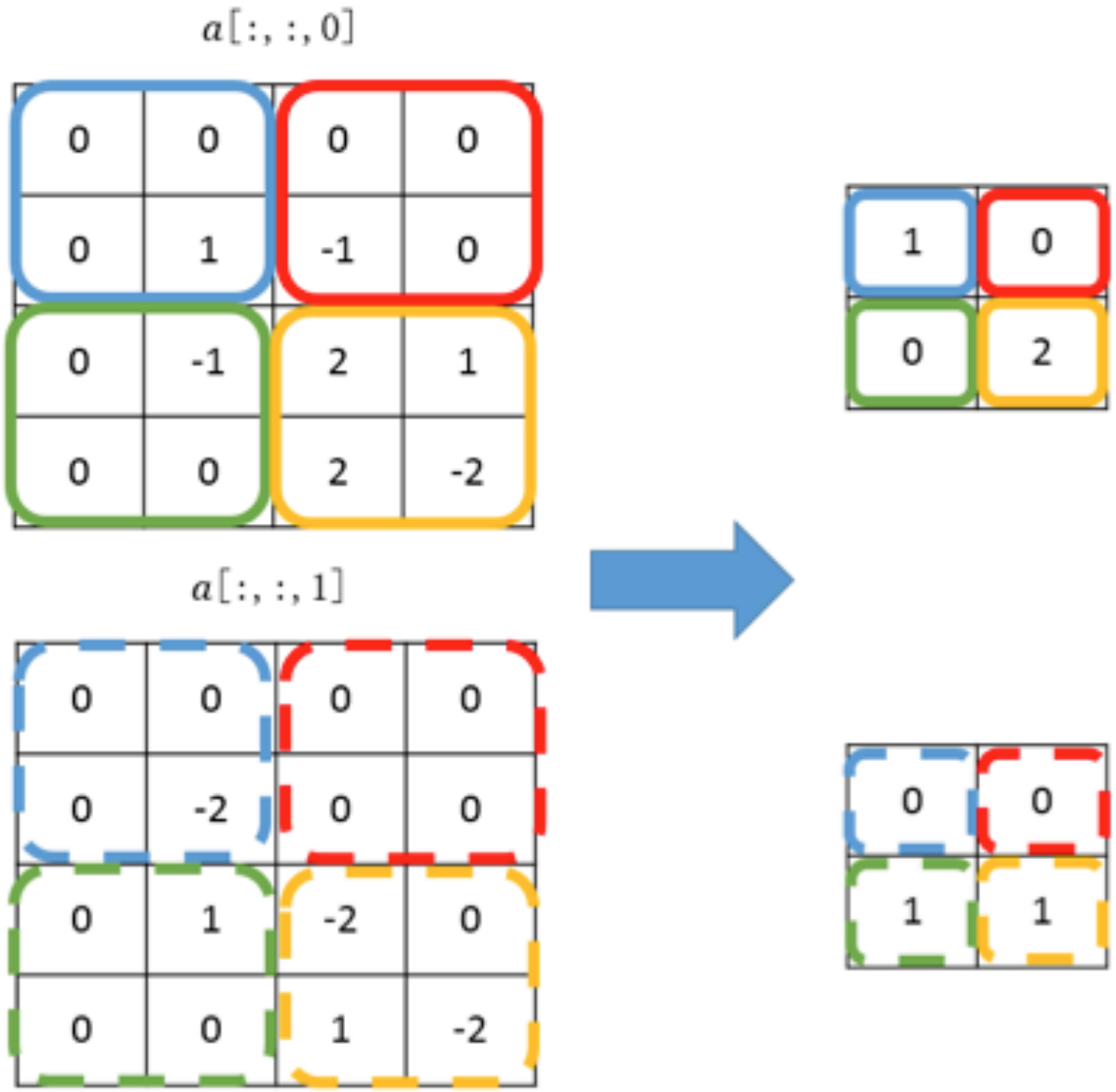
caicloud才云
©版权归才云所有，违反必究

卷积



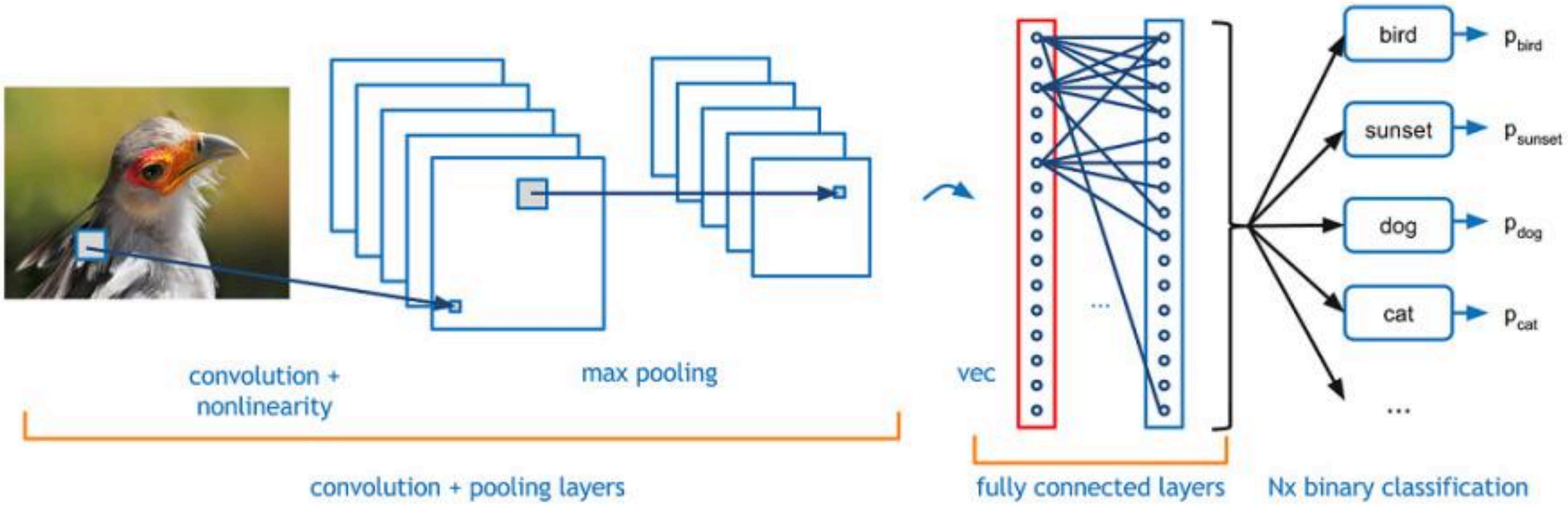
caicloud才云
©版权归才云所有，违反必究

池化



©版权归才云所有，违反必究

卷积神经网络的处理流程



caicloud才云
©版权归才云所有，违反必究

提纲

01 深度学习与TensorFlow

02 卷积神经网络

03 迁移学习

04 构建图像识别应用

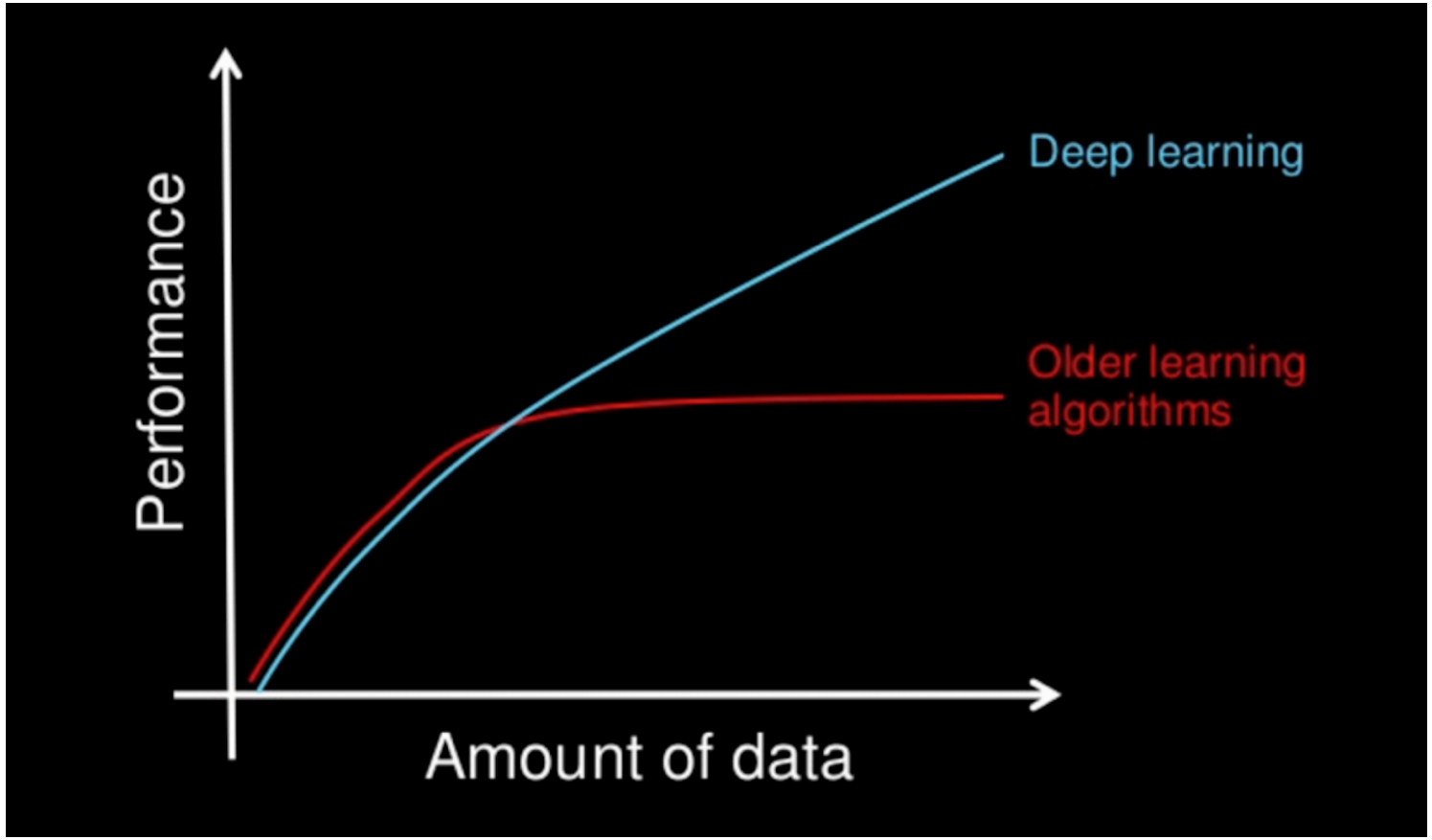
caicloud才云

©版权归才云所有，违反必究

深度学习的挑战——数据量

1. AlphaGo
   3000万局 高手棋谱
   3000万局+ 自我对弈

2. ImageNet
   1500万+ 图像
   2万+ 标签

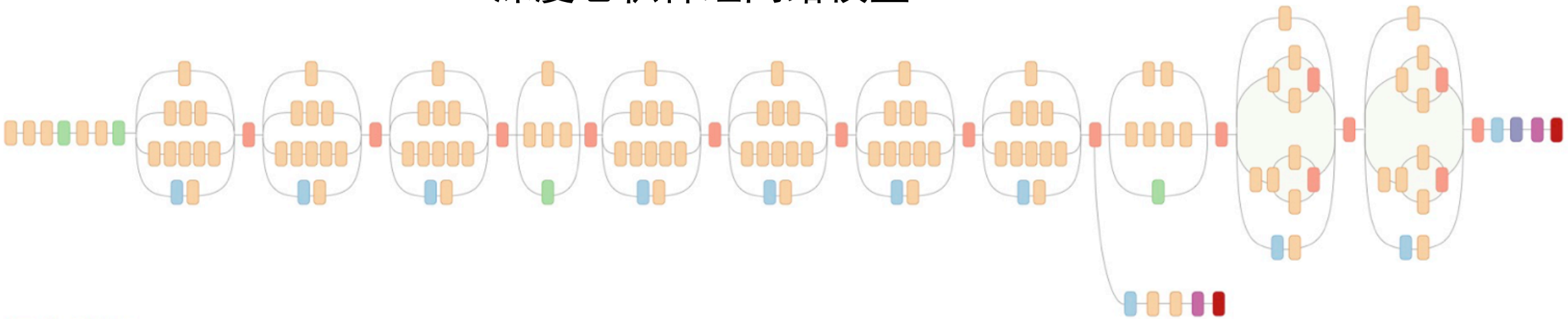


caicloud才云
©版权归才云所有，违反必究

深度学习的挑战——计算量

Inception-V3

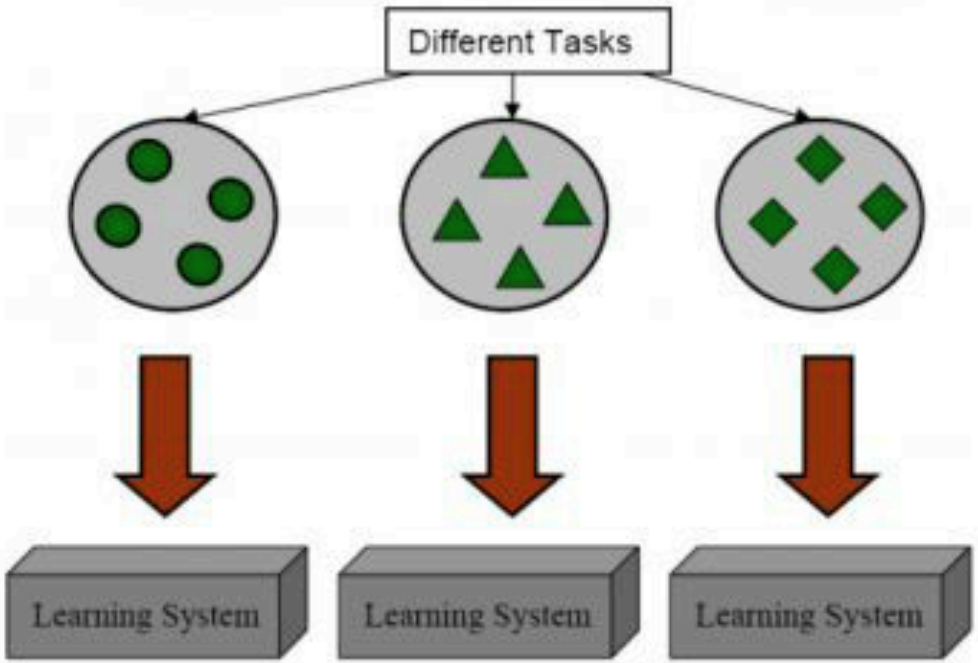
深度卷积神经网络模型



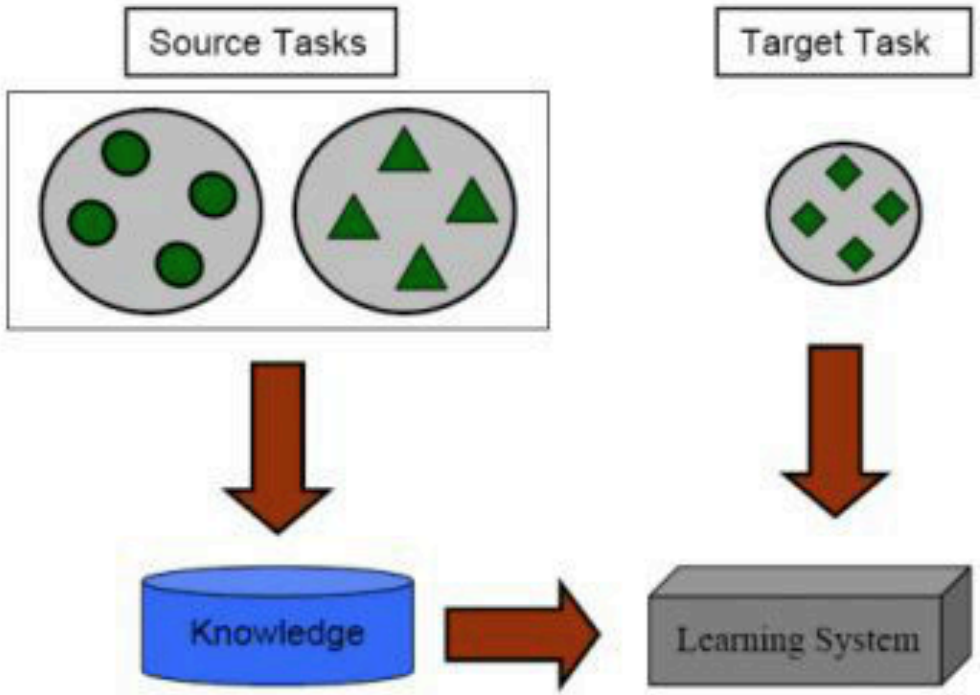
- 50余层深度网络
- 千万级参数
- 单次十亿级计算量
- 单机训练时间：半年

caicloud才云
©版权归才云所有，违反必究

迁移学习的思想



● 传统机器学习



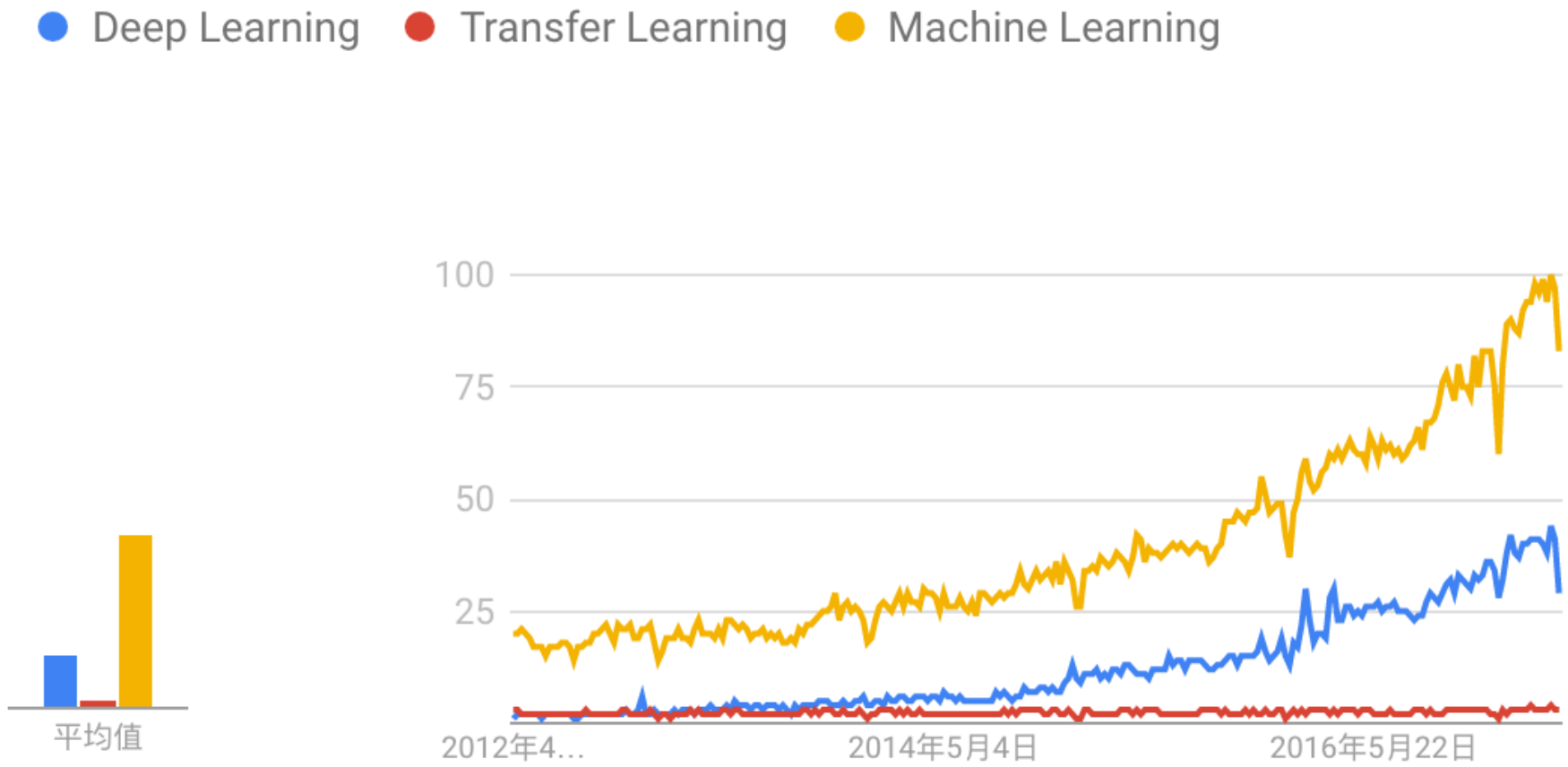
● 迁移学习

caicloud才云
©版权归才云所有，违反必究

迁移学习的流行趋势

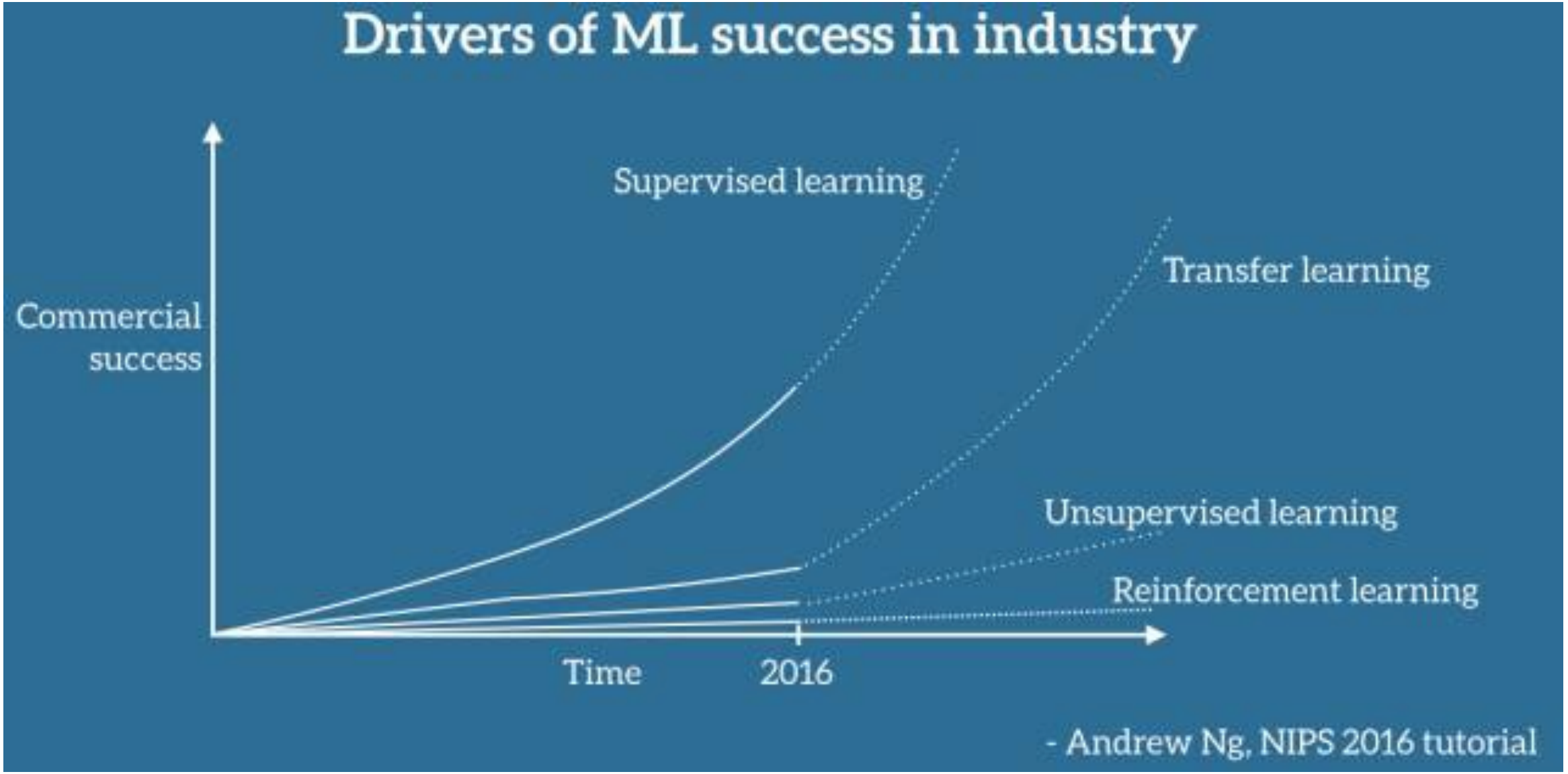
热度随时间变化的趋势

Google Trends



caicloud才云
©版权归才云所有，违反必究

迁移学习的发展趋势



caicloud 才云
©版权归才云所有，违反必究

迁移学习的优势

图像识别	不使用迁移学习	使用迁移学习
数据量	1万张图	300~500张图
开发时间	1-2周	1-2天
计算时间	半年	3-5天
准确率	95%	93%

caicloud才云
©版权归才云所有，违反必究

迁移学习的形式

迁移形式	操作过程
样本迁移	将任务A中部分适合任务B的数据抽取出来，与其它数据一起针对任务B进行训练
特征迁移	识别小轿车的神经网络中，用于表示车轮车窗的网络参数，可以直接迁移到卡车识别
模型迁移	类似特征迁移，整个模型的网络结构都被迁移。比如将Inception-V3卷积神经网络模型，迁移到其他图像识别任务。

caicloud才云
©版权归才云所有，违反必究

迁移学习的处理方法

复用卷积池化层

Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

特征向量

我们只训练全连接层

图像分类结果

caicloud才云
©版权归才云所有，违反必究

提纲

01 深度学习与TensorFlow

02 卷积神经网络

03 迁移学习

04 构建图像识别应用

caicloud才云

©版权归才云所有，违反必究

构建图像识别应用(一)

1. 加载Inception-V3模型，读取其中的瓶颈层、输入层张量名称（Tensor）

```
# 加载Inception-V3模型文件
with gfile.FastGFile(FLAGS.model_file, 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())

# 读取输入层张量、瓶颈层张量
image_data_tensor, bottleneck_tensor = tf.import_graph_def(
    graph_def, return_elements=[IMAGE_DATA_TENSOR_NAME, BOTTLENECK_TENSOR_NAME])
```

caicloud才云
©版权归才云所有，违反必究

2. 复用卷积池化层，生成图像特征向量（瓶颈层）

```
def run_bottleneck_on_image(sess, image_data, image_data_tensor, bottleneck_tensor):
    ''' 根据图像数据，复用卷积池化层，生成特征向量
    Args:
        sess: TensorFlow的会话(session)
        image_data: 图像数据
        image_data_tensor: 输入层张量
        bottleneck_tensor: 瓶颈层张量
    Returns:
        bottleneck_values: 特征向量
    '''
    bottleneck_values = sess.run(bottleneck_tensor, {image_data_tensor: image_data})
    bottleneck_values = np.squeeze(bottleneck_values)
    return bottleneck_values
```

caicloud才云
©版权归才云所有，违反必究

3. 定义神经网络的前向传播过程

```
def full_connect_with_relu(inputs, weights_shape, biases_shape):
    ''' 定义神经网络前向传播过程
    Args:
        inputs: 输入张量
        weights_shape: 本层连接的维度
        biases_shape: 本层偏置项的维度
    Returns:
        下层神经元的数值结果
    '''
    weights = tf.get_variable("weights", weights_shape,
                initializer=tf.truncated_normal_initializer(stddev=0.001))
    biases  = tf.get_variable("biases", biases_shape,
                initializer=tf.constant_initializer(0.0))
    result = tf.matmul(inputs, weights) + biases
    return tf.nn.relu(result)
```

caicloud才云
©版权归才云所有，违反必究

4. 定义双层全连接神经网络

```
def inference(inputs):
    ''' 定义双层全连接神经网络 '''
    # 输入层 => 第1层
    with tf.variable_scope("layer1"):
        layer1 = full_connect_with_relu(inputs, [BOTTLENECK_SIZE, LAYER1_SIZE], [LAYER1_SIZE])
    # 第1层 => 第2层
    with tf.variable_scope("layer2"):
        layer2 = full_connect_with_relu(layer1, [LAYER1_SIZE, LAYER2_SIZE], [LAYER2_SIZE])
    # 第2层 => 输出层
    with tf.variable_scope("logits"):
        logits = full_connect_with_relu(layer2, [LAYER2_SIZE, NUM_CLASSES], [NUM_CLASSES])
    return logits
```

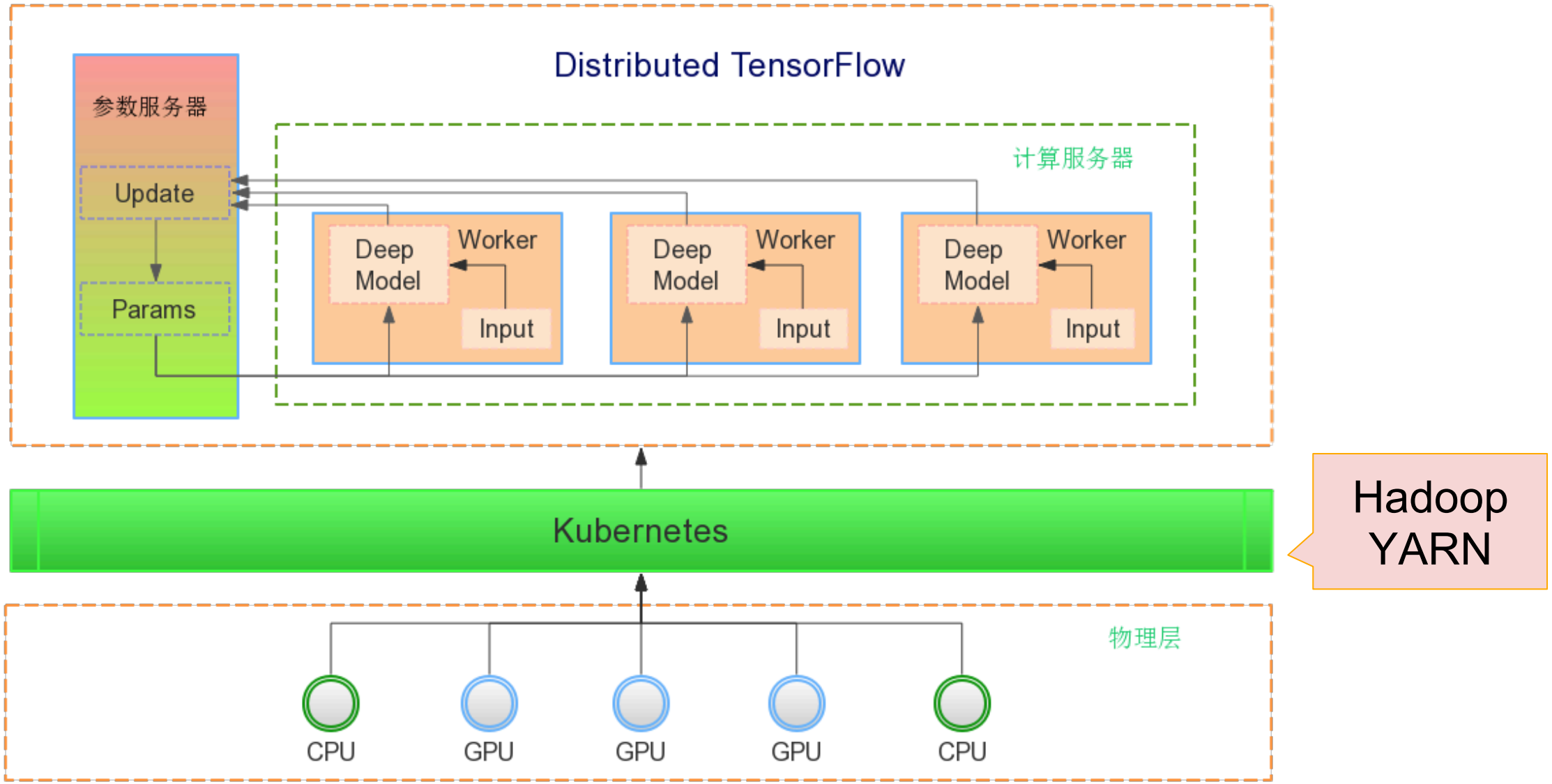
caicloud才云
©版权归才云所有，违反必究

5. 训练全连接神经网络

```
# 计算预测结果
logits = inference(bottleneck_input)
predict_answer = tf.nn.softmax(logits)

# 定义交叉熵损失函数
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits, ground_truth_input)
cross_entropy_mean = tf.reduce_mean(cross_entropy)

# 定义优化目标
train_step = tf.train.GradientDescentOptimizer(FLAGS.learning_rate).minimize(cross_entropy_mean)

with tf.Session() as sess:
    # 执行模型训练
    for i in range(FLAGS.max_steps):
        train_bottlenecks, train_ground_truth = get_random_cached_bottlenecks(
            n_classes, image_lists, FLAGS.batch_size)
        _, cross_entropy_cost = sess.run([train_step, cross_entropy_mean], feed_dict={
            bottleneck_input: train_bottlenecks, ground_truth_input: train_ground_truth})
```

caicloud才云
©版权归才云所有，违反必究

分布式TensorFlow深度学习平台



caicloud才云
©版权归才云所有，违反必究

caicloud

才云

Q&A

©版权归才云所有，违反必究