TensorFlow实践 - AlphaGo与天弈围棋

张卫鹏　天壤网络科技　深度学习工程师
wpzhang@tianrang-inc.com

AlphaGo 取胜之道？

Rank	Name	♂♀	Flag	Elo
1	Ke Jie	♂		3619
2	Google DeepMind AlphaGo			3602
3	Park Junghwan	♂		3576
4	Tuo Jiaxi	♂		3534
5	Mi Yuting	♂		3534
6	Lee Sedol	♂		3533
7	Kim Jiseok	♂		3526
8	Iyama Yuta	♂		3515
9	Shi Yue	♂		3511
10	Chen Yaoye	♂		3505
11	Zhou Ruiyang	♂		3492
12	Li Qincheng	♂		3485
13	Shin Jinseo	♂		3482
14	Huang Yunsong	♂		3482
15	Park Yeonghun	♂		3478
16	Kang Dongyun	♂		3476
17	Tan Xiao	♂		3473
18	Tang Weixing	♂		3472
19	Choi Cheolhan	♂		3471
20	Lian Xiao	♂		3467

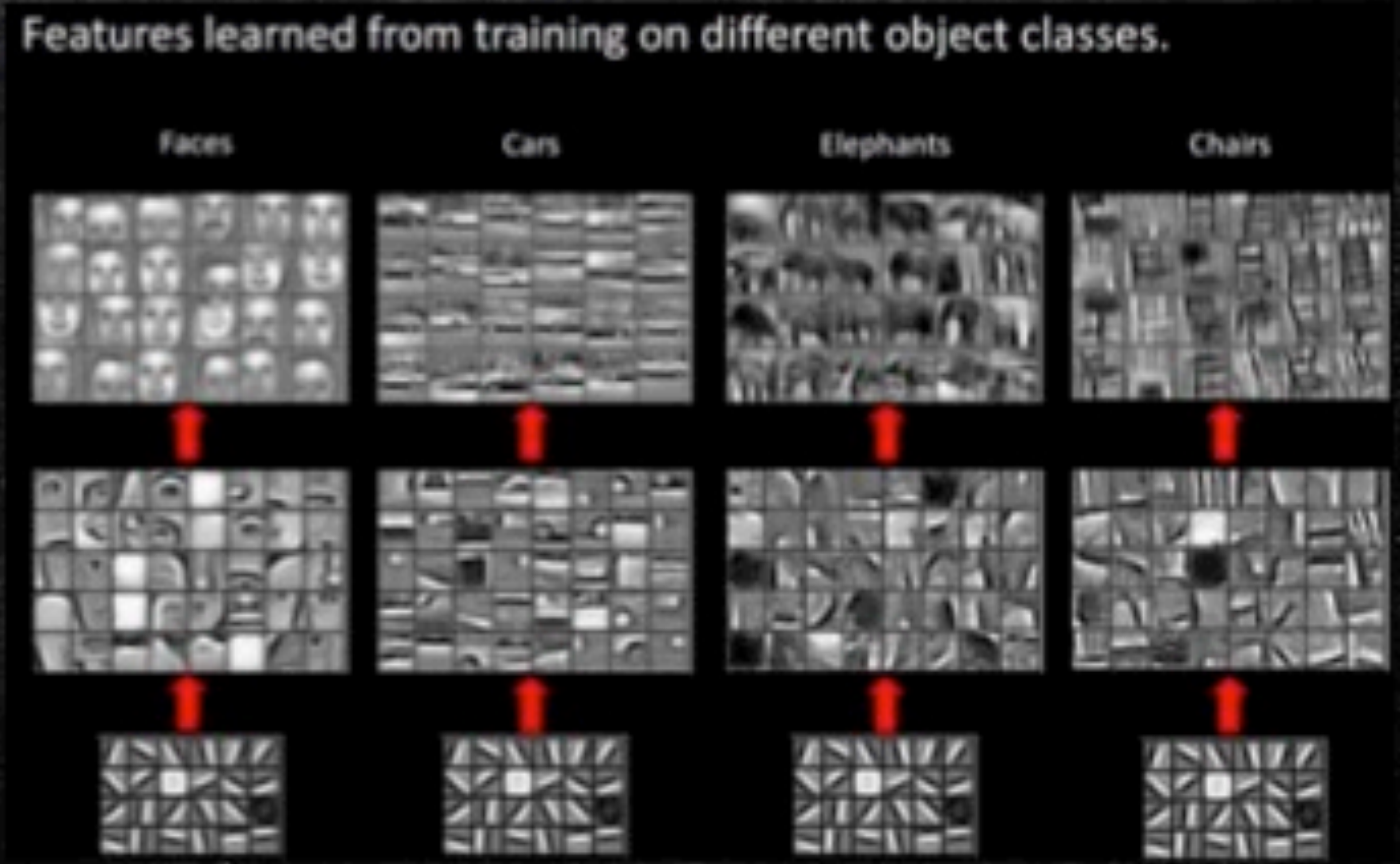
分享内容

- Deep Learning
- TensorFlow
- AlphaGo
- 天弈围棋
- Deep Q-Network

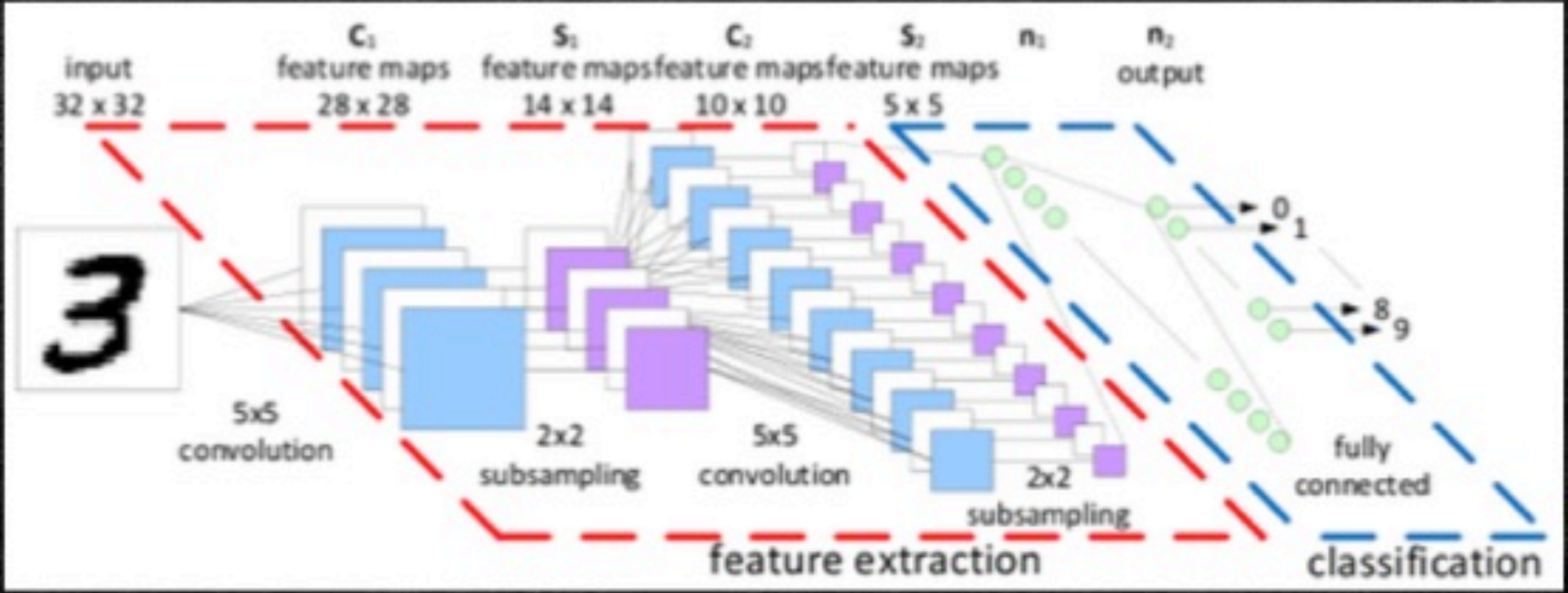
Deep Learning

- 不需要太多的feature work
- 拟合复杂函数的能力

- 需要强大的计算能力
- 不能简单的迁移知识



Deep Learning



Deep Learning Training

- 神经元数量
- 激活函数
- 损失函数
- 正则化参数
- Much parameters ......
- GPU Cluster

	Propagation	Back-propagation
Sigmoid	$y_s = \frac{1}{1+e^{-x_s}}$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \frac{1}{(1+e^{x_s})(1+e^{-x_s})}$
Tanh	$y_s = \tanh(x_s)$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \frac{1}{\cosh^2 x_s}$
ReLu	$y_s = \max(0, x_s)$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \mathbb{I}\{x_s > 0\}$
Ramp	$y_s = \min(-1, \max(1, x_s))$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \mathbb{I}\{-1 < x_s < 1\}$

TensorFlow

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express

Tensor（张量）意味着N维数组，Flow（流）意味着基于数据流图的计算，TensorFlow即为张量从图的一端流动到另一端。 TensorFlow一大亮点是支持异构设备分布式计算，它能够在各个平台上自动运行模型，从电话、单个CPU / GPU到成百上千GPU卡组成的分布式系统。

TensorFlow - 架构

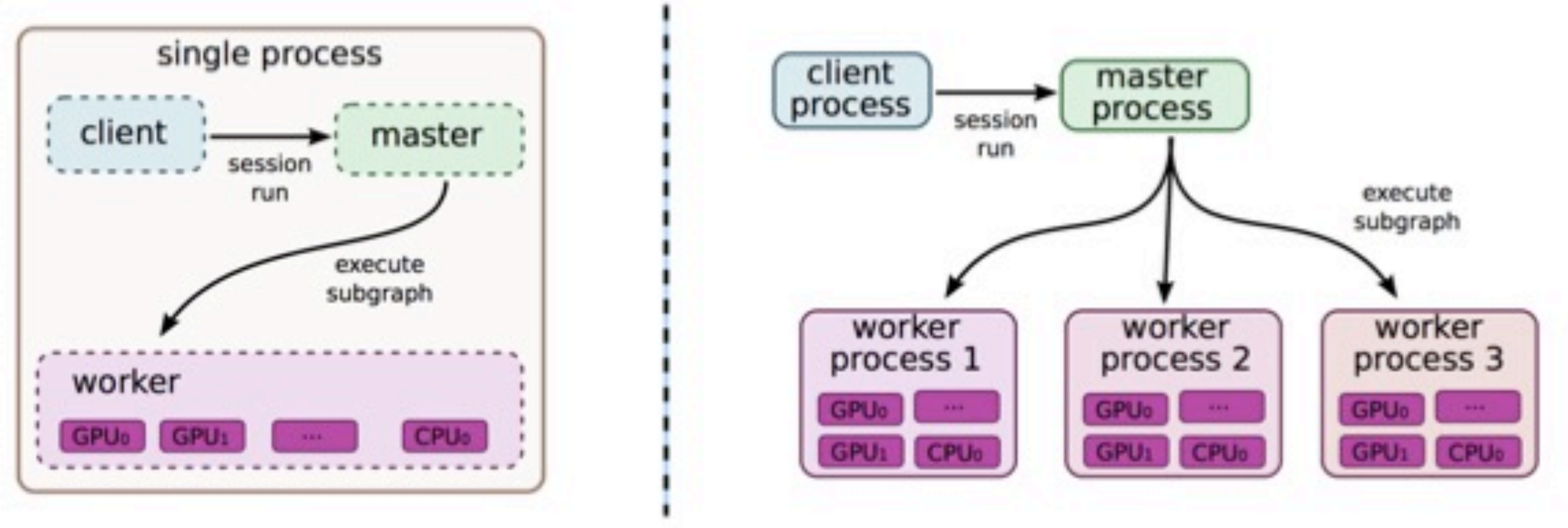


Figure 3: Single machine and distributed system structure

TensorFlow - 案例

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                      # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1))    # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                          # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)                # Relu(Wx+b)
C = [...]                                             # Cost computed as a function
                                                      # of Relu

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...          # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})             # Fetch cost, feeding x=input
  print step, result
```

Figure 1: Example TensorFlow code fragment

TensorFlow - 案例

```python
# Import the library
import tensorflow as tf

# Define the graph
hello_op = tf.constant('Hello, TensorFlow!')
a = tf.constant(10)
b = tf.constant(32)
compute_op = tf.add(a, b)

# Define the session to run graph
with tf.Session() as sess:
    print(sess.run(hello_op))
    print(sess.run(compute_op))
```

```python
import tensorflow as tf
import numpy as np

# Prepare train data
train_X = np.linspace(-1, 1, 100)
train_Y = 2 * train_X + np.random.randn(*train_X.shape) * 0.33 + 10

# Define the model
X = tf.placeholder("float")
Y = tf.placeholder("float")
w = tf.Variable(0.0, name="weight")
b = tf.Variable(0.0, name="bias")
loss = tf.square(Y - tf.mul(X, w) - b)
train_op = tf.train.GradientDescentOptimizer(0.01).minimize(loss)

# Create session to run
with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    epoch = 1
    for i in range(10):
        for (x, y) in zip(train_X, train_Y):
            _, w_value, b_value = sess.run([train_op, w, b], feed_dict={X: x, Y: y})
        print("Epoch: {}, w: {}, b: {}".format(epoch, w_value, b_value))
        epoch += 1
```

TensorFlow - 案例 build network

```
with tf.op_scope([inputs], scope, 'inception_v3'):
  with scopes.arg_scope([ops.conv2d, ops.fc, ops.batch_norm, ops.dropout], is_training=is_training):
    with scopes.arg_scope([ops.conv2d, ops.max_pool, ops.avg_pool], stride=1, padding='VALID'):

      # 299 x 299 x 3
      end_points['conv0'] = ops.conv2d(inputs, 32, [3, 3], stride=2, scope='conv0')

      # 149 x 149 x 32
      end_points['conv1'] = ops.conv2d(end_points['conv0'], 32, [3, 3], scope='conv1')

      # 147 x 147 x 32
      end_points['conv2'] = ops.conv2d(end_points['conv1'], 64, [3, 3], padding='SAME', scope='conv2')

      # 147 x 147 x 64
      end_points['pool1'] = ops.max_pool(end_points['conv2'], [3, 3], stride=2, scope='pool1')

      # 73 x 73 x 64
      end_points['conv3'] = ops.conv2d(end_points['pool1'], 80, [1, 1], scope='conv3')

      # 73 x 73 x 80.
      end_points['conv4'] = ops.conv2d(end_points['conv3'], 192, [3, 3], scope='conv4')

      # 71 x 71 x 192.
      end_points['pool2'] = ops.max_pool(end_points['conv4'], [3, 3], stride=2, scope='pool2')

      # 35 x 35 x 192.
      net = end_points['pool2']
```

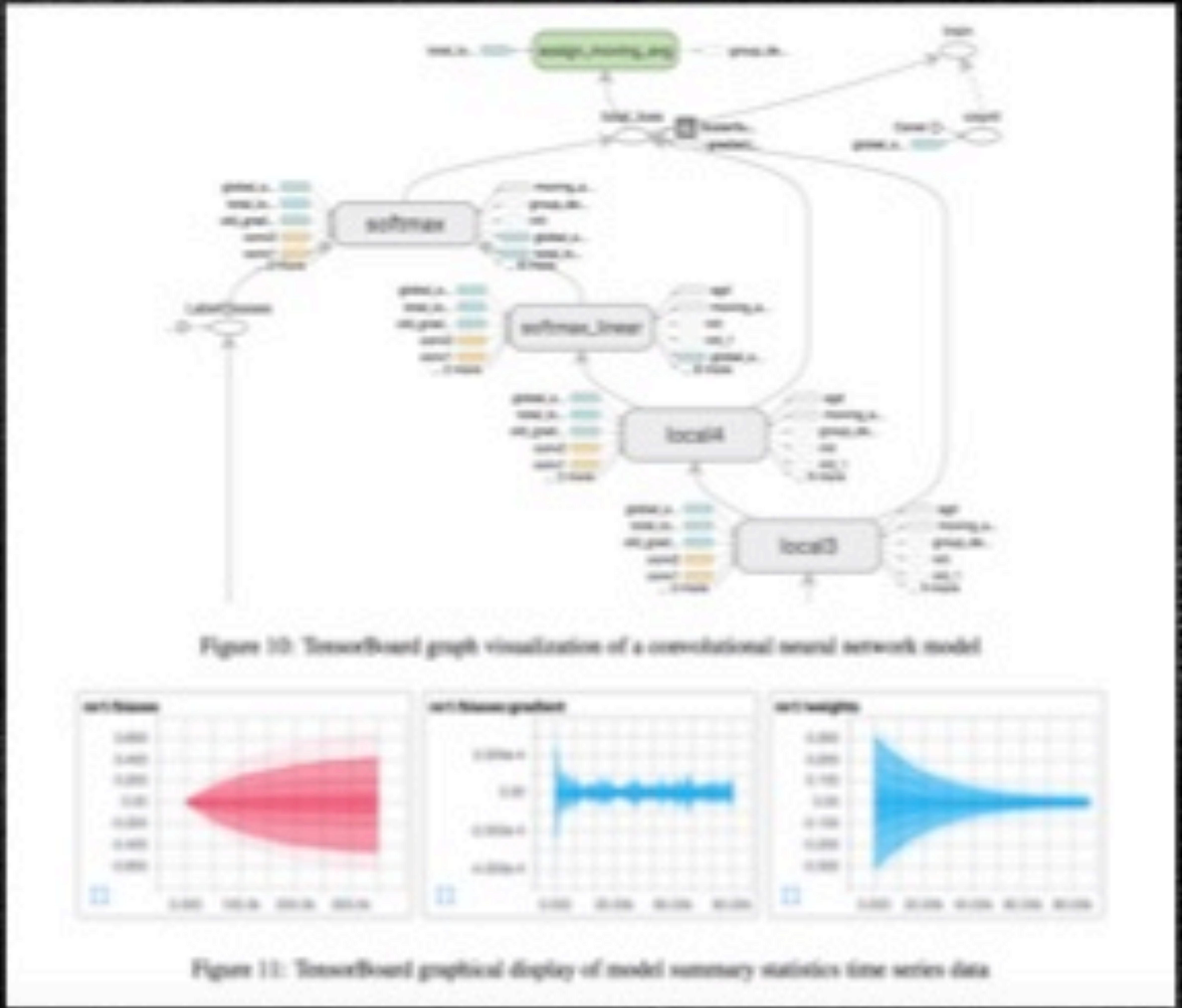
典型的CNN卷积网络

TensorFlow - 亮点

```python
# Create session to run graph
with tf.Session() as sess:
  summary_op = tf.merge_all_summaries()
  writer = tf.train.SummaryWriter(tensorboard_dir, sess.graph)
  sess.run(init_op)
  sess.run(tf.initialize_local_variables())

  if mode == "train" or mode == "train_from_scratch":
    if mode != "train_from_scratch":
      ckpt = tf.train.get_checkpoint_state(checkpoint_dir)
      if ckpt and ckpt.model_checkpoint_path:
        print("Continue training from the model {}".format(
            ckpt.model_checkpoint_path))
        saver.restore(sess, ckpt.model_checkpoint_path)
```

Continues Learning

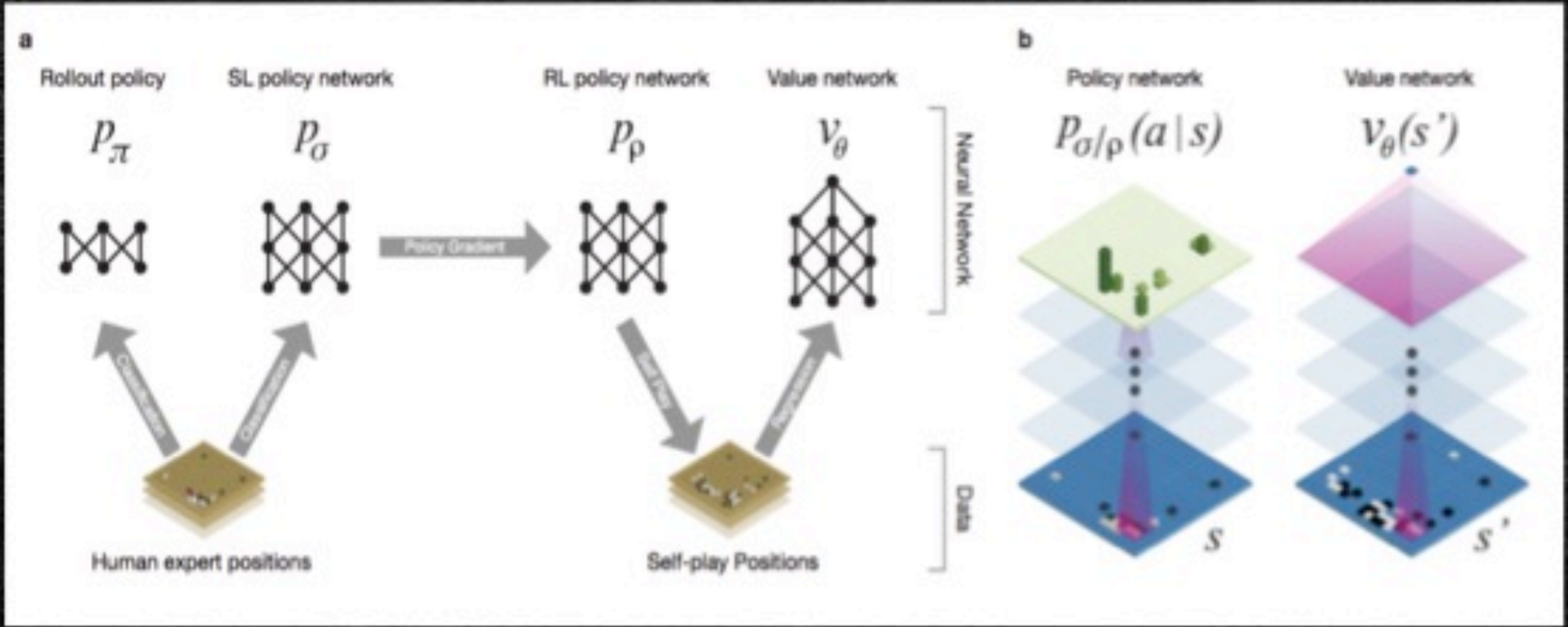


TensorBoard

深度学习框架对比

	网络与模型能力	接口	模型部署	性能	架构
Caffe	第一个主流的工业级深度学习工具	支持pycaffe接口，使用protobuf定义模型	跨平台	简单快速	平均水准
CNTK	通用的、平台独立	配置文件，没有高级语言接口	跨平台，不支持ARM架构	简单快速	
TensorFlow	工业级深度学习平台	支持python和C++两个接口	跨平台，不支持Windows	非常好	架构清晰，模块化设计
Theano	支持大部分先进网络，引领了符号图在编程网络中使用的趋势	支持python接口	跨平台，但是对工业用户缺少吸引力	启动时间慢	架构比较变态，全要打包为python字符串
Torch	对卷积网络支持非常好	运行在LuaJIT上	需要LuaJIT支持，限制部署	非常好	清晰的设计和模块化的接口

AlphaGo

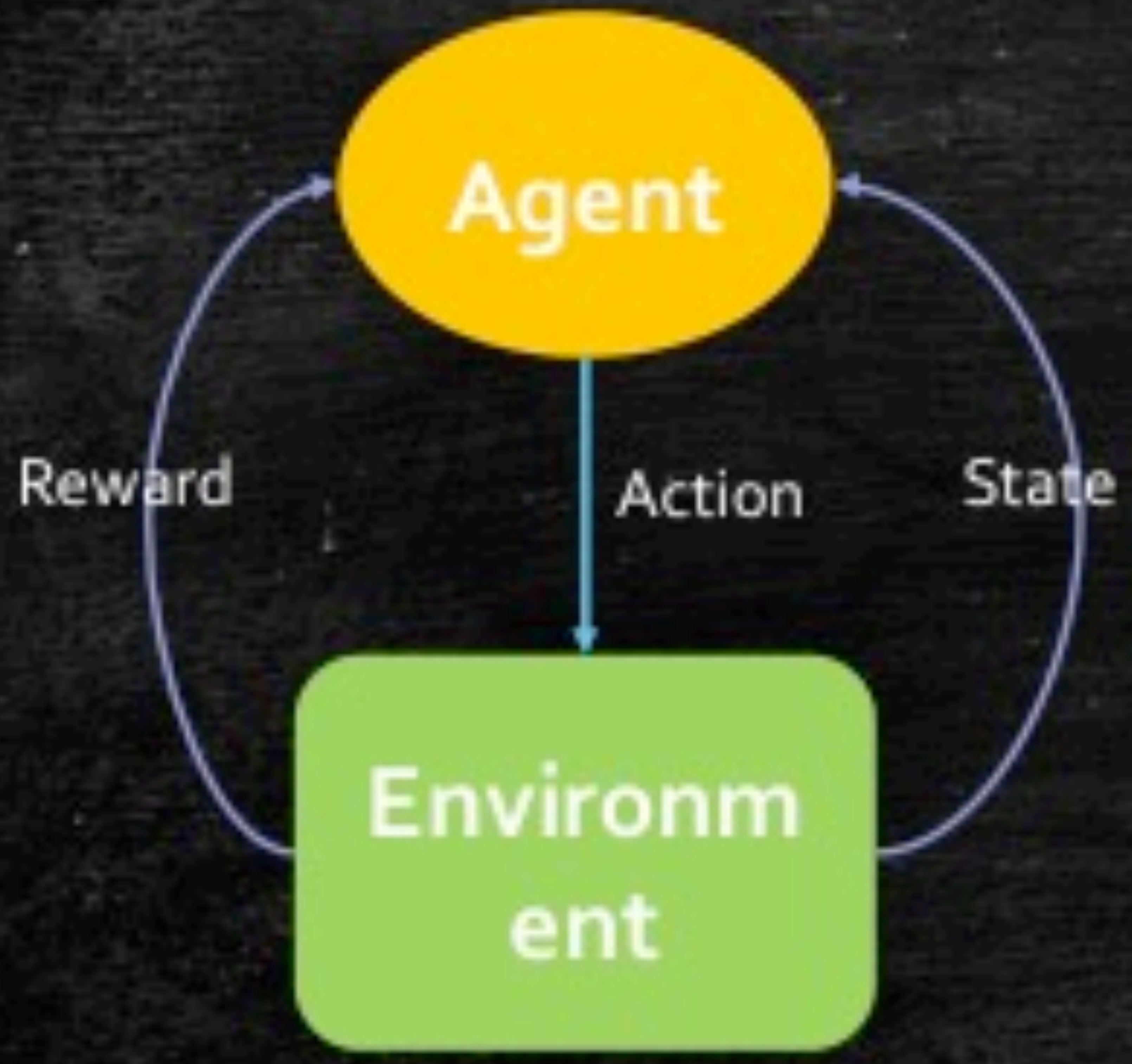


a
Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy Gradient
Neural Network
Data
Human expert positions
Self-play Positions
b
Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s'

天弈围棋

- http://yi.tianrang.com
- 基于TensorFlow实现，深度神经网络
- 围棋爱好者的训练营

DQN – Deep Q-Network



Deep Q-Network Algorithm

The pseudo-code for the Deep Q Learning algorithm, as given in [1], can be found below:

```
Initialize replay memory D to size N
Initialize action-value function Q with random weights
for episode = 1, M do
    Initialize state s_1
    for t = 1, T do
        With probability ϵ select random action a_t
        otherwise select a_t=max_a  Q(s_t,a; θ_i)
        Execute action a_t in emulator and observe r_t and s_(t+1)
        Store transition (s_t,a_t,r_t,s_(t+1)) in D
        Sample a minibatch of transitions (s_j,a_j,r_j,s_(j+1)) from D
        Set y_j:=
            r_j for terminal s_(j+1)
            r_j+γ*max_(a^' )  Q(s_(j+1),a'; θ_i) for non-terminal s_(j+1)
        Perform a gradient step on (y_j-Q(s_j,a_j; θ_i))^2 with respect to θ
    end for
end for
```

DQN - 案例

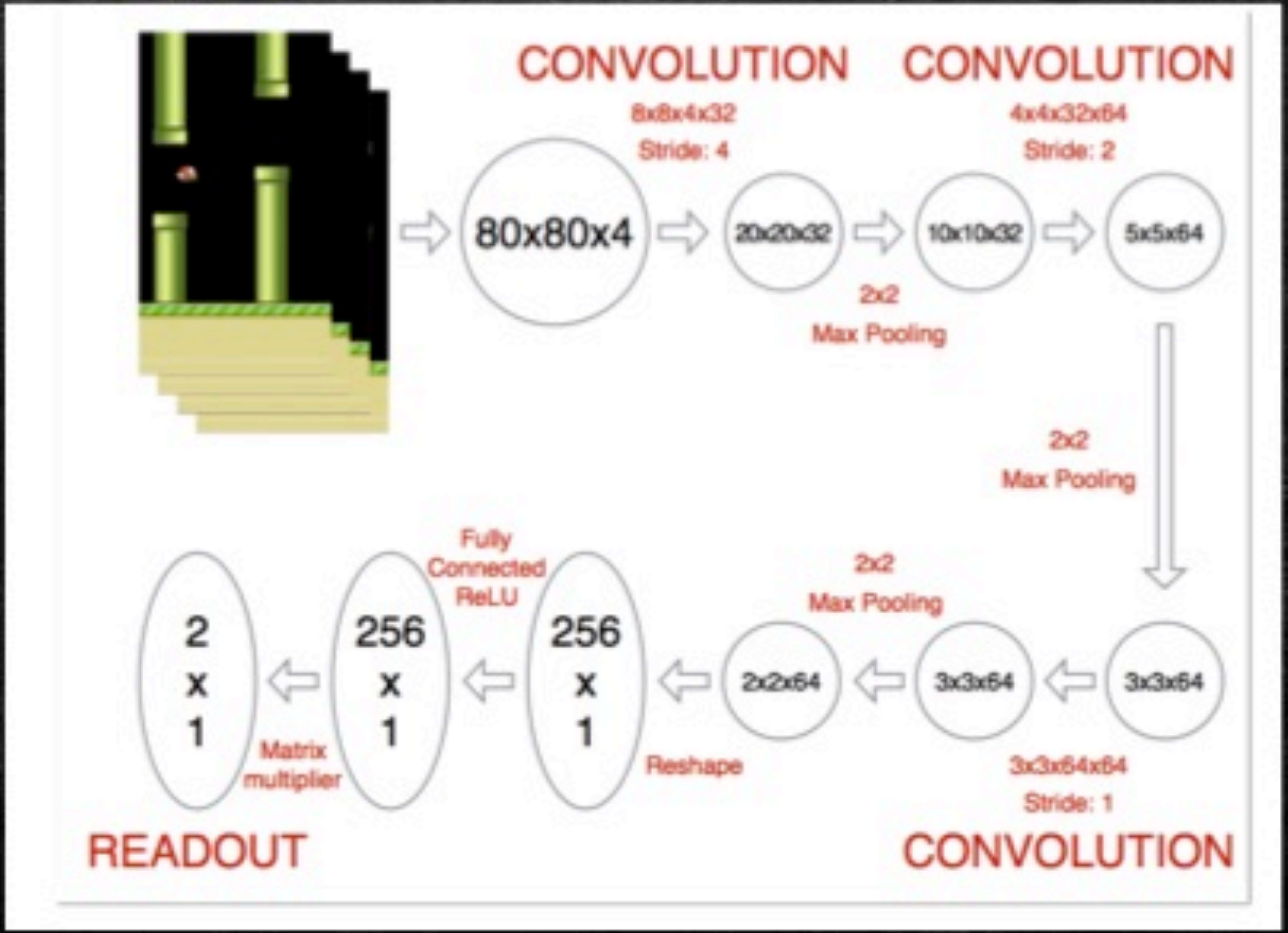
FlappyBird

BreakOut

GTA无人驾驶

DQN – Deep Q-Network

无需领域知识，输入为原始像素，输出为action对应的概率分布



Reference

- [1] Mastering the Game of Go with Deep Neural Networks and Tree Search, Nature 2015
- [2] TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, Google 2015
- [3] Generative Adversarial Nets, Arxiv 2014
- [4] Human-level Control through Deep Reinforcement Learning, Nature 2015
- [5] Playing Atari with Deep Reinforcement Learning, NIPS

谢谢！

天壤网络科技（上海）有限公司