GIAC BEIJING Dec.12.16-17

架構 ARCHNOTES 高可用架构

小米深度学习平台架构与实践

技术架构未来

thegiac.com

About me

- 陈迪豪，就职小米，深度学习领域
- 分布式存储 - HBase - Ceph
- 基础架构 - Docker - OpenStack
- 人工智能 - TensorFlow - K8S
- Seagull作者，1400+ stars

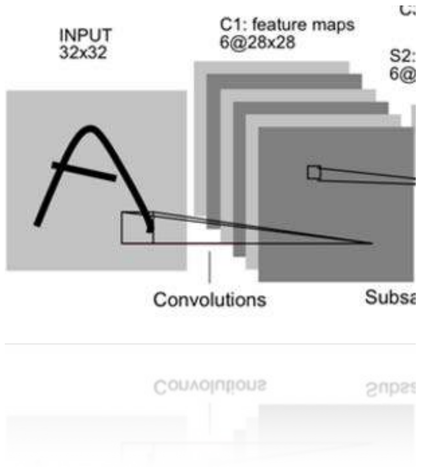
Agenda

- 机器学习与深度学习应用
- 深度学习平台架构与设计
- 深度学习平台应用与实践

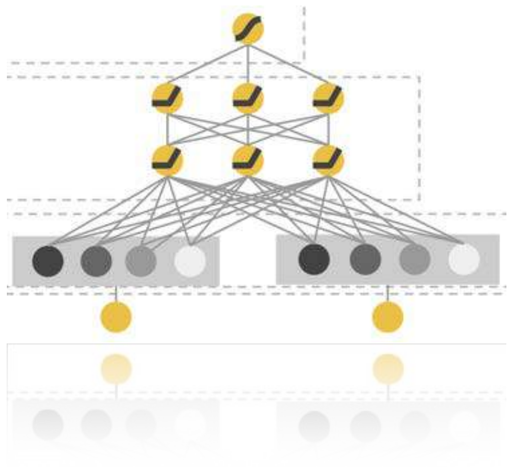
Agenda

- 机器学习与深度学习应用
- 深度学习平台架构与设计
- 深度学习平台应用与实践

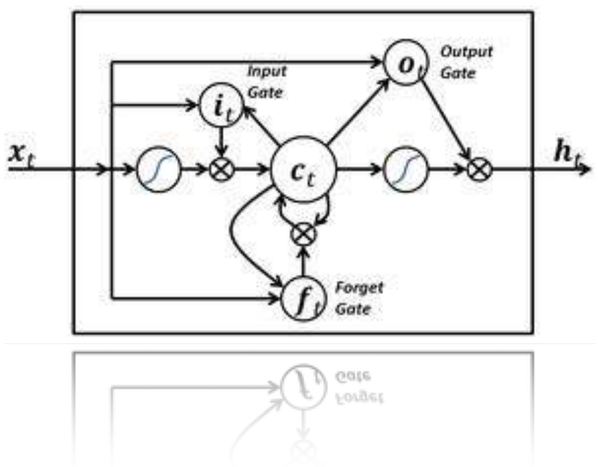
Introduce ML/DL



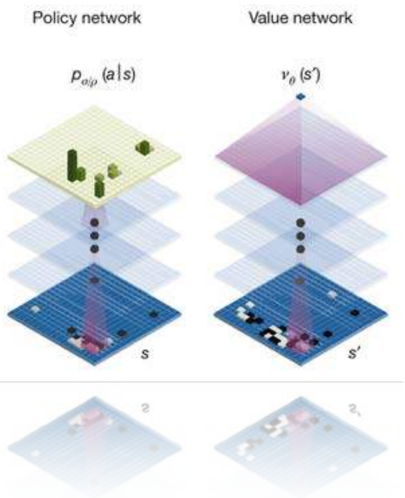
CNN



MLP

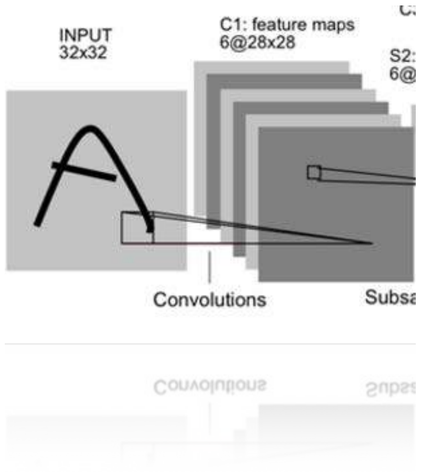


RNN/LSTM

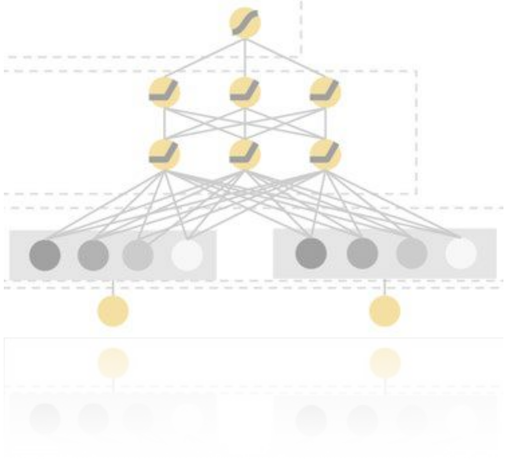


RL

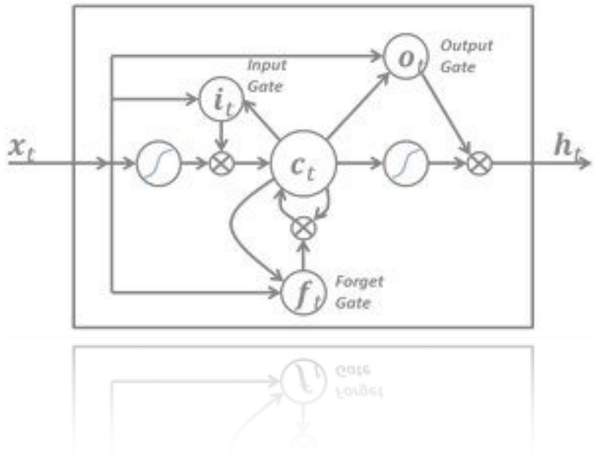
Introduce ML/DL



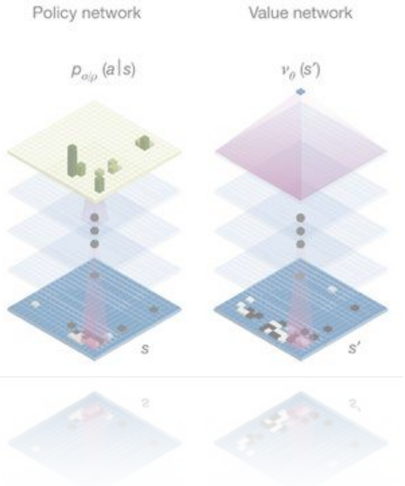
CNN



MLP

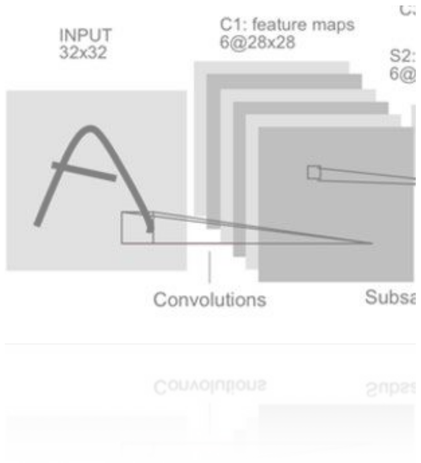


RNN/LSTM

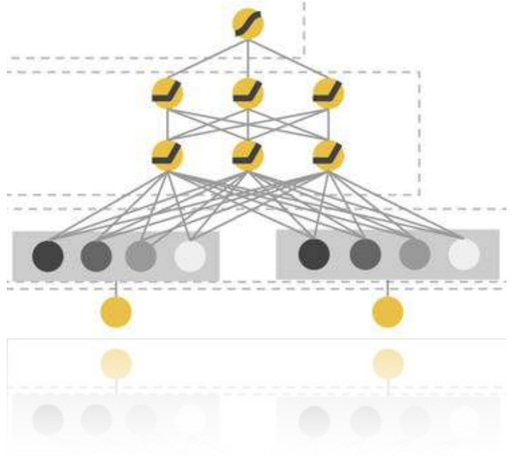


RL

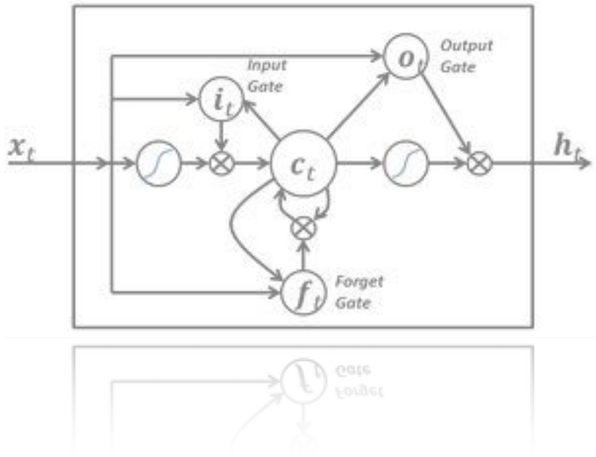
Introduce ML/DL



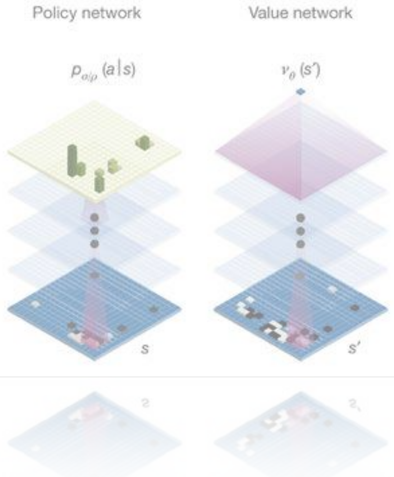
CNN



MLP

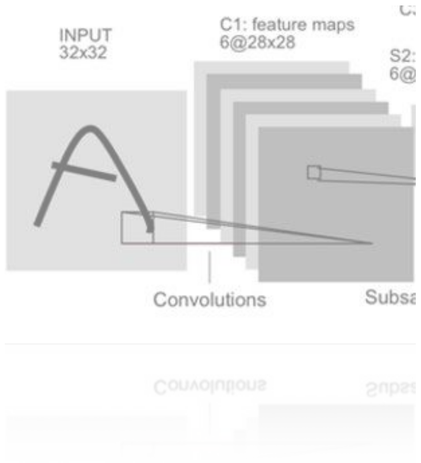


RNN/LSTM

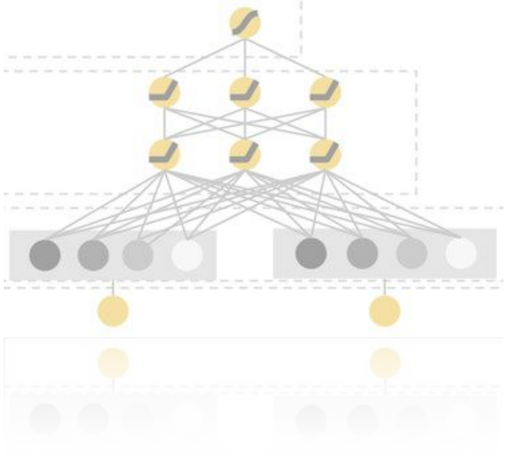


RL

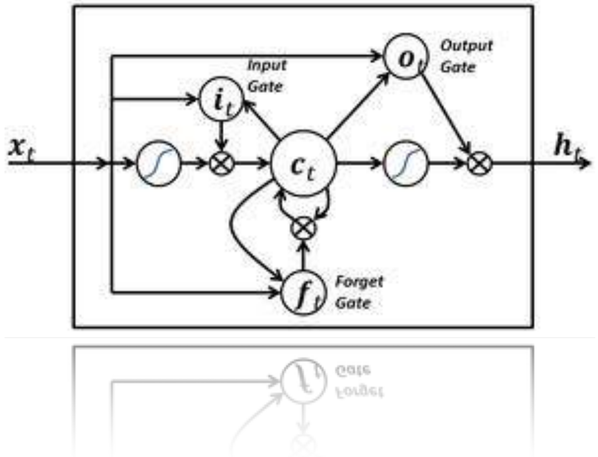
Introduce ML/DL



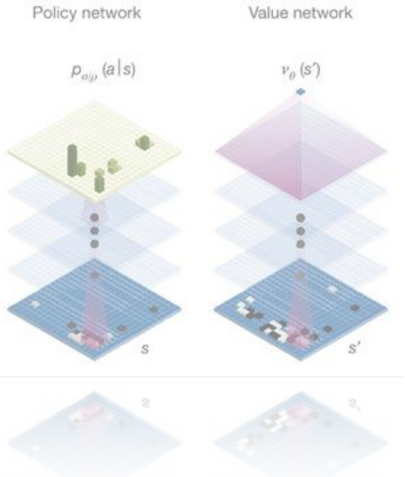
CNN



MLP

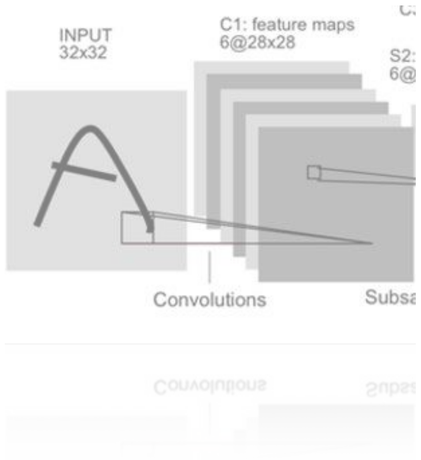


RNN/LSTM

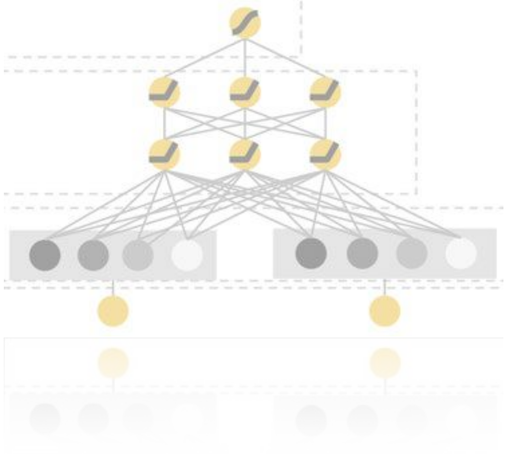


RL

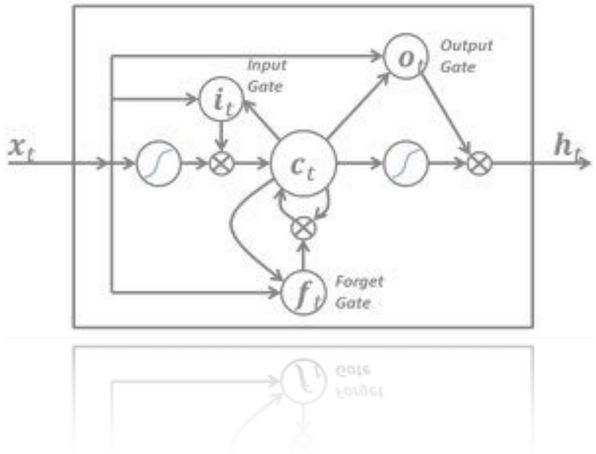
Introduce ML/DL



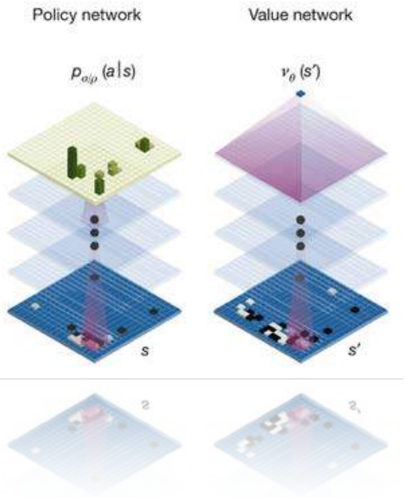
CNN



MLP

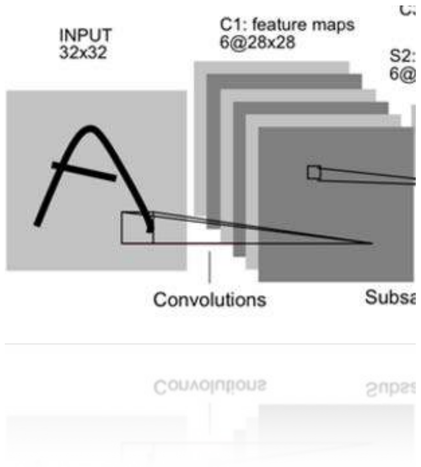


RNN/LSTM

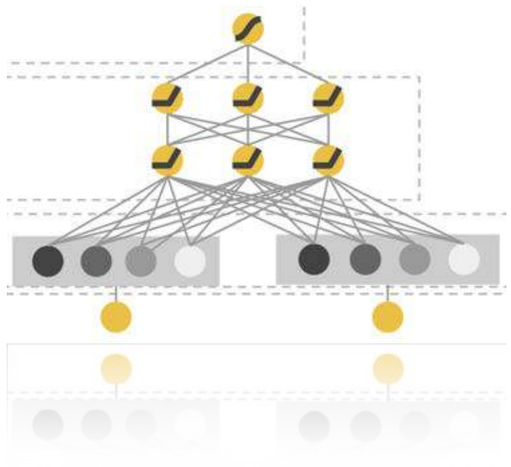


RL

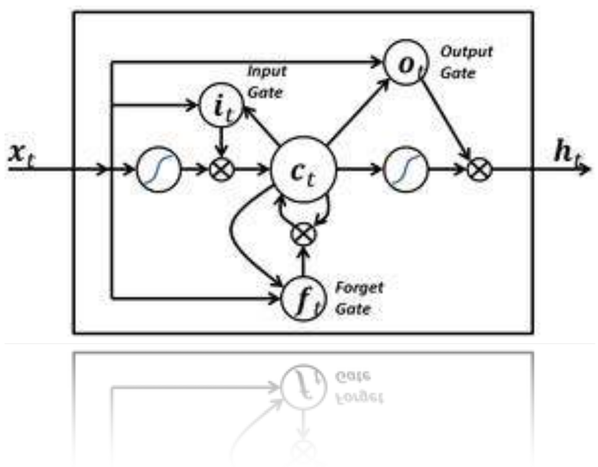
Introduce ML/DL



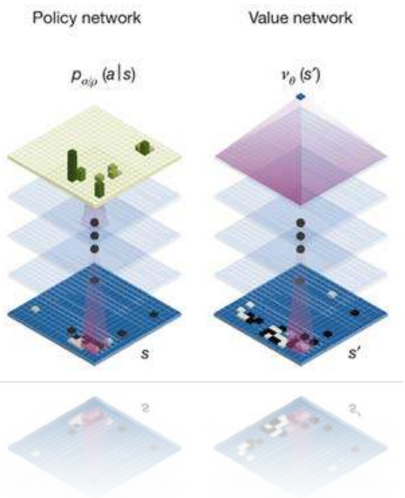
CNN



MLP

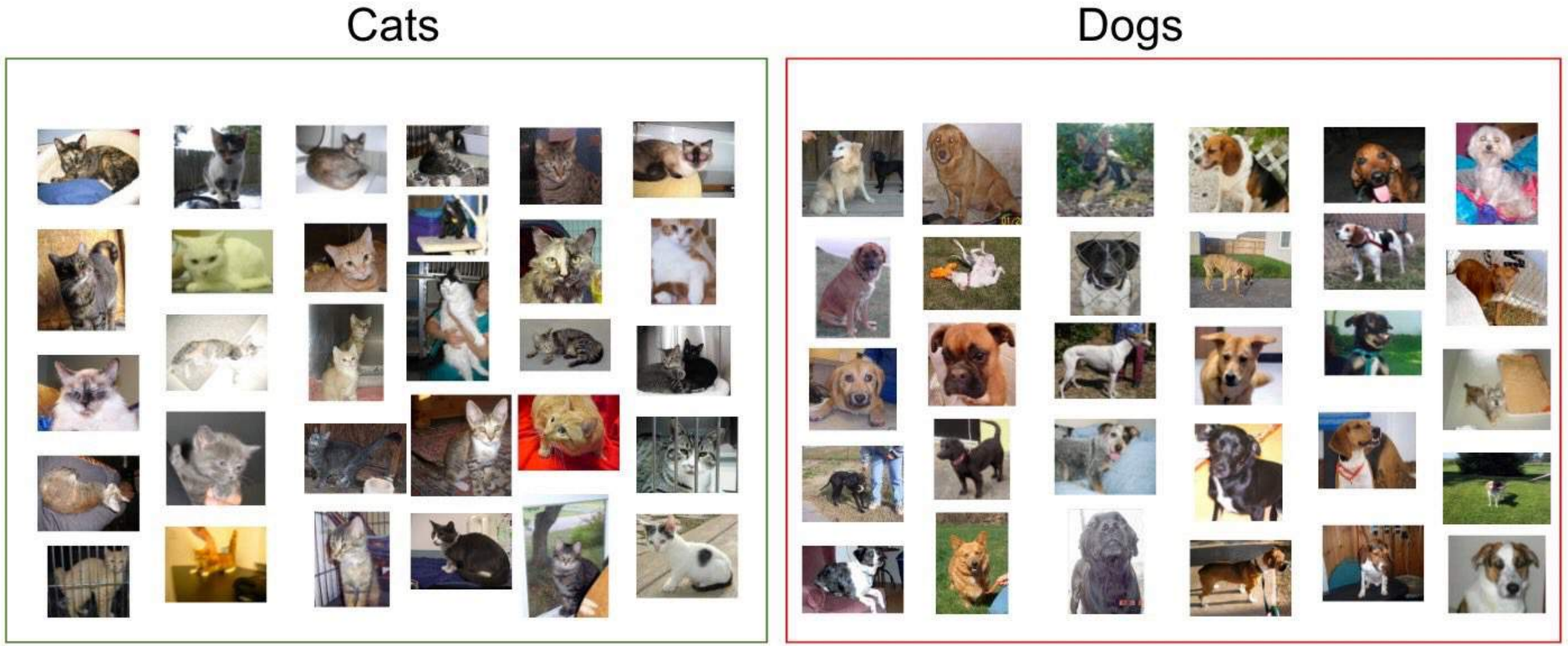


RNN/LSTM



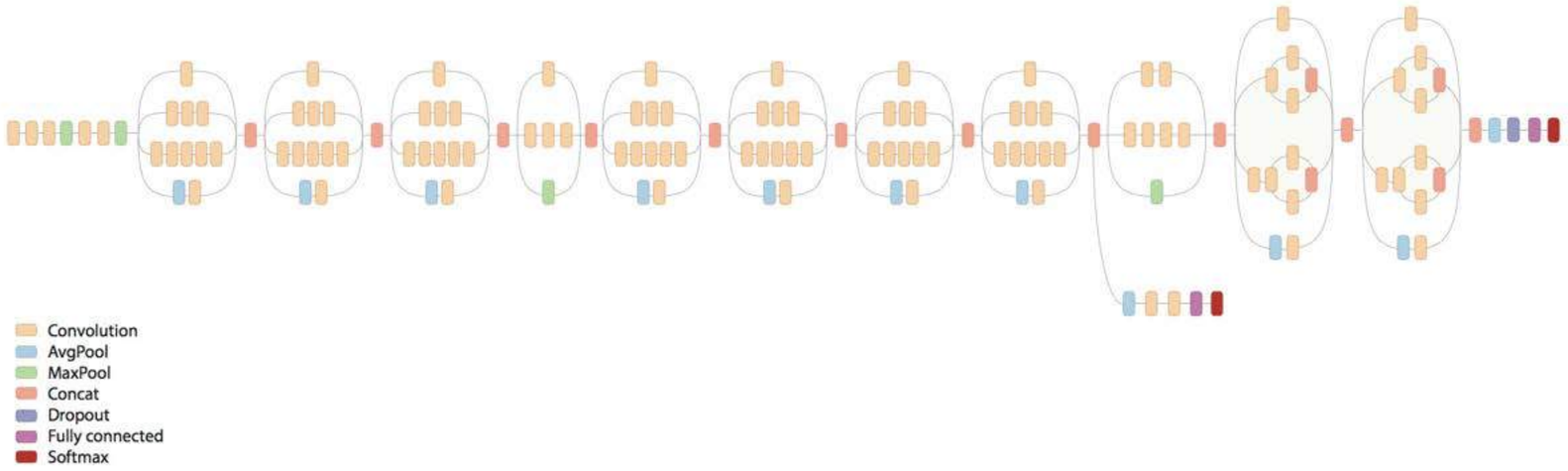
RL

Case Study: Image Classification

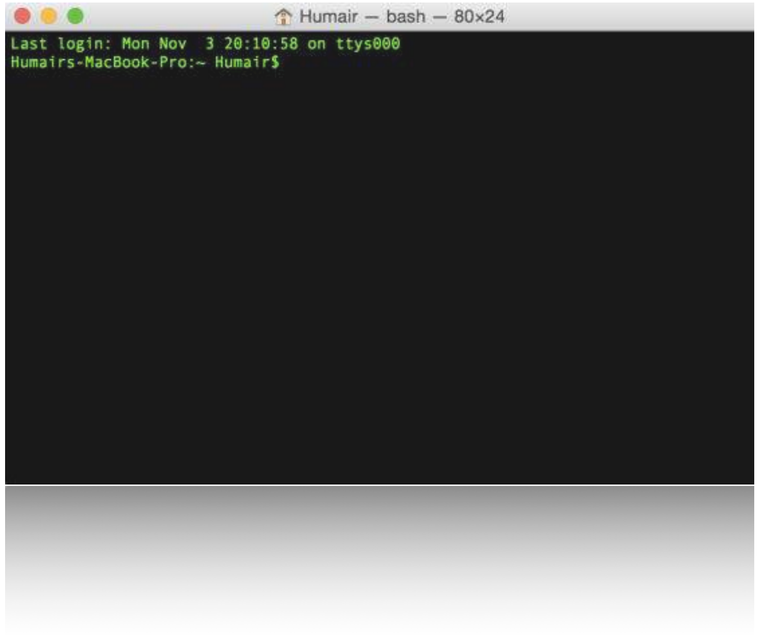


Sample of cats & dogs images from Kaggle Dataset

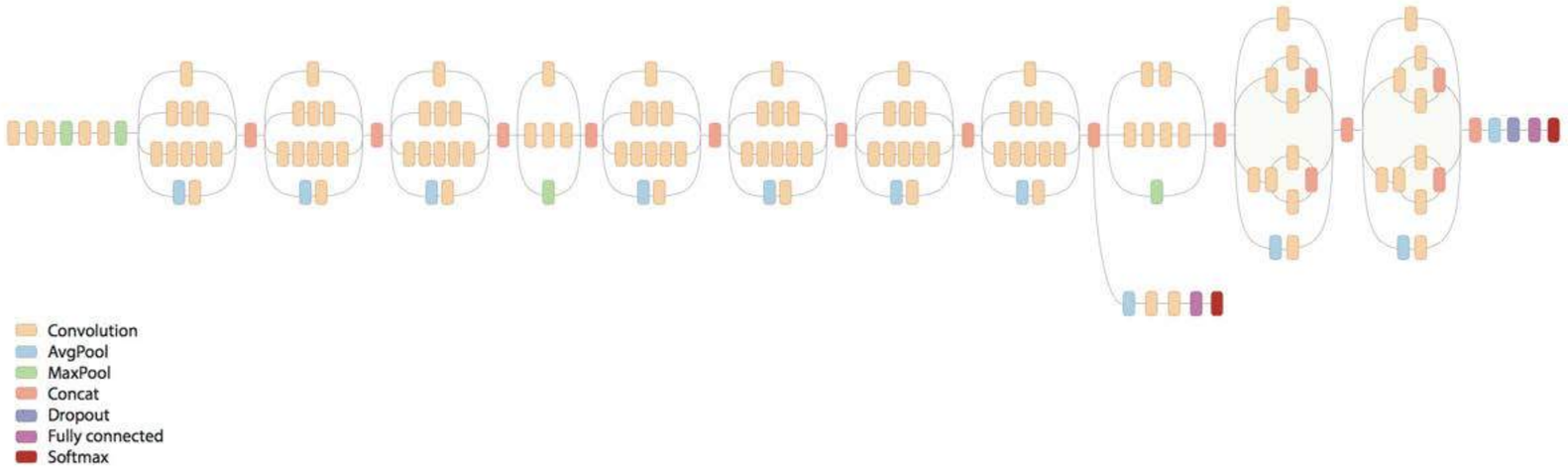
Case Study: Image Classification



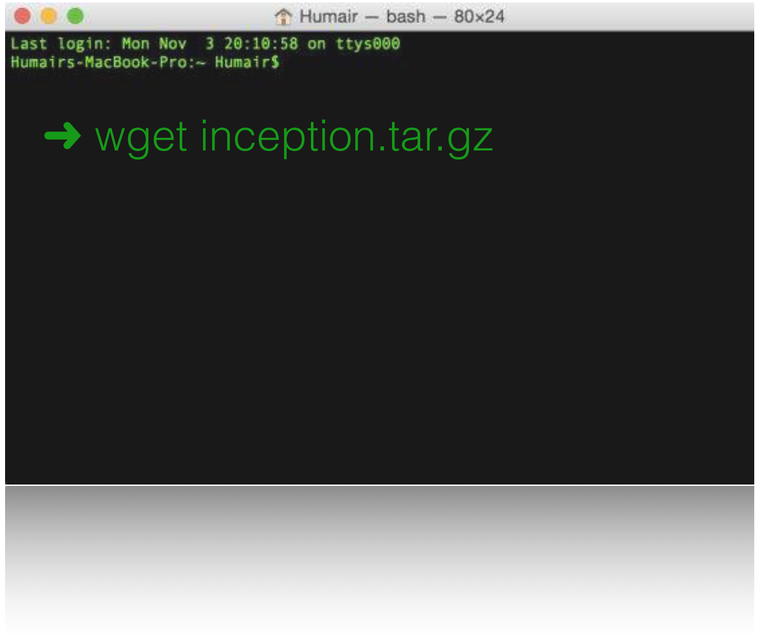
https://github.com/tensorflow/models/tree/master/inception



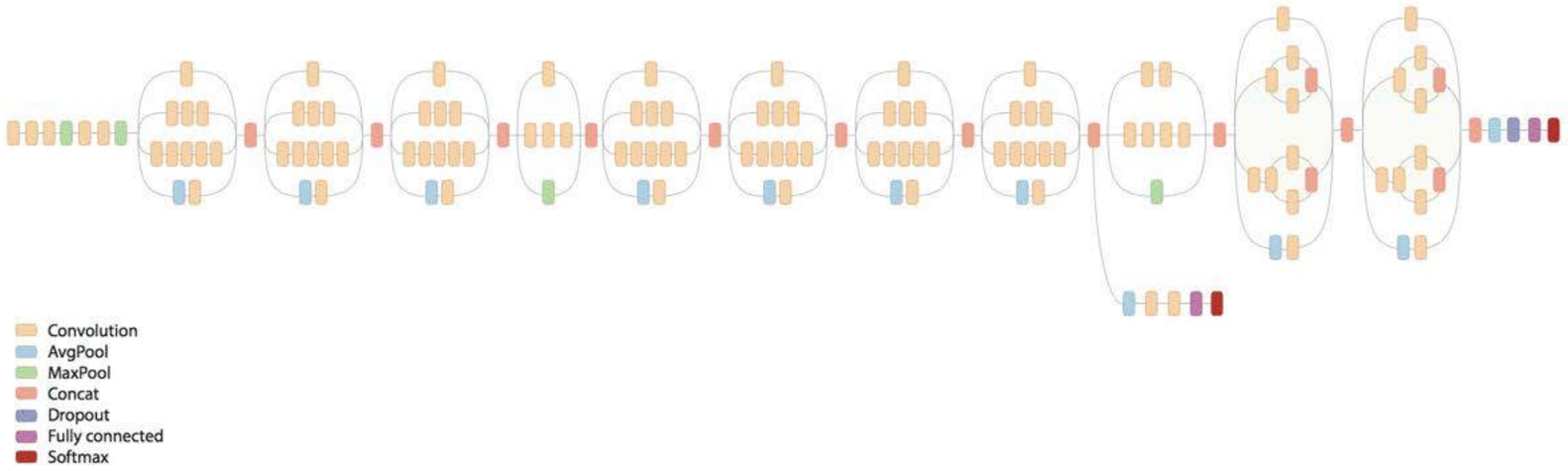
Case Study: Image Classification



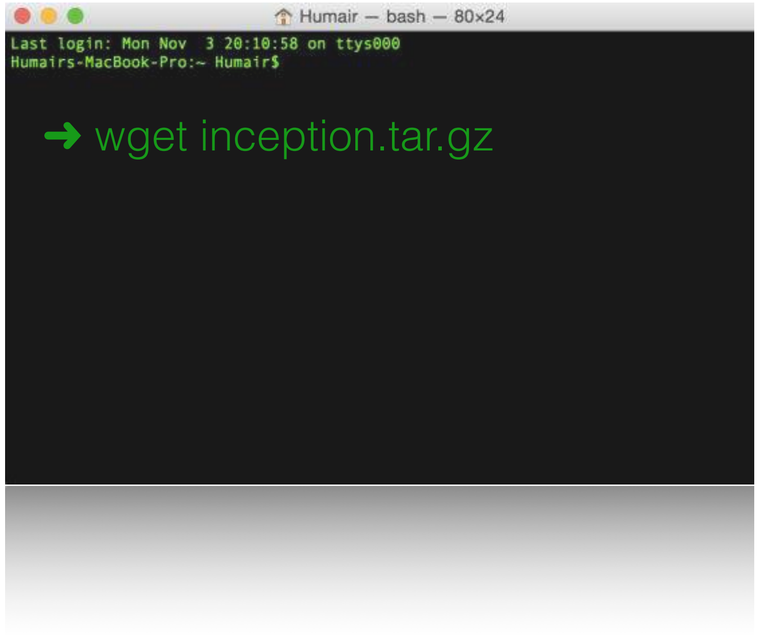
https://github.com/tensorflow/models/tree/master/inception



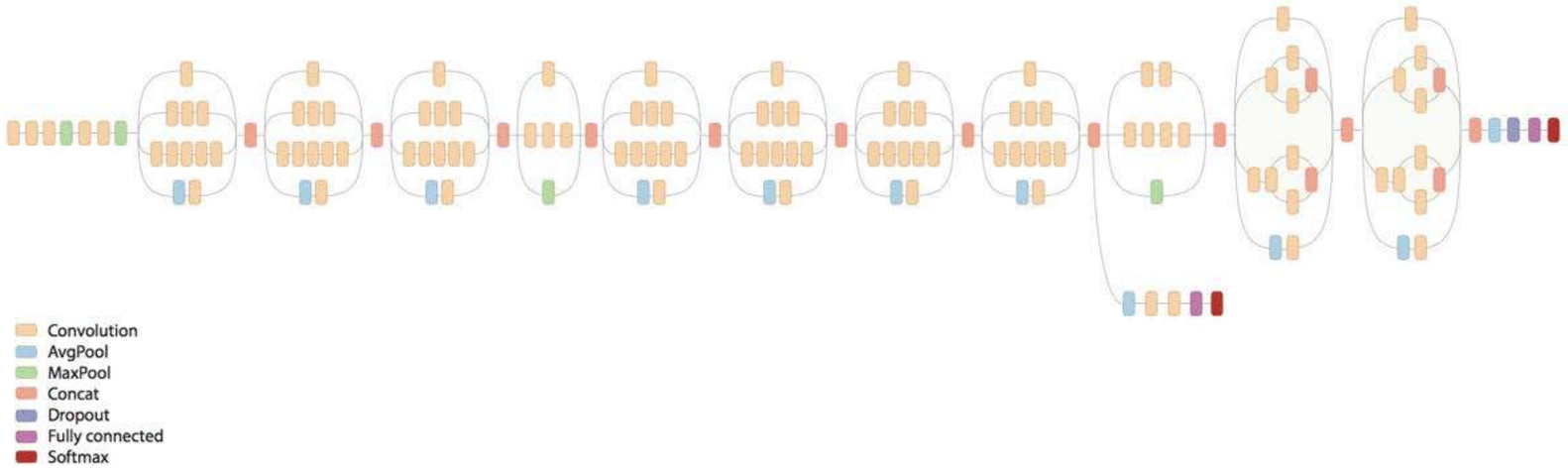
Case Study: Image Classification



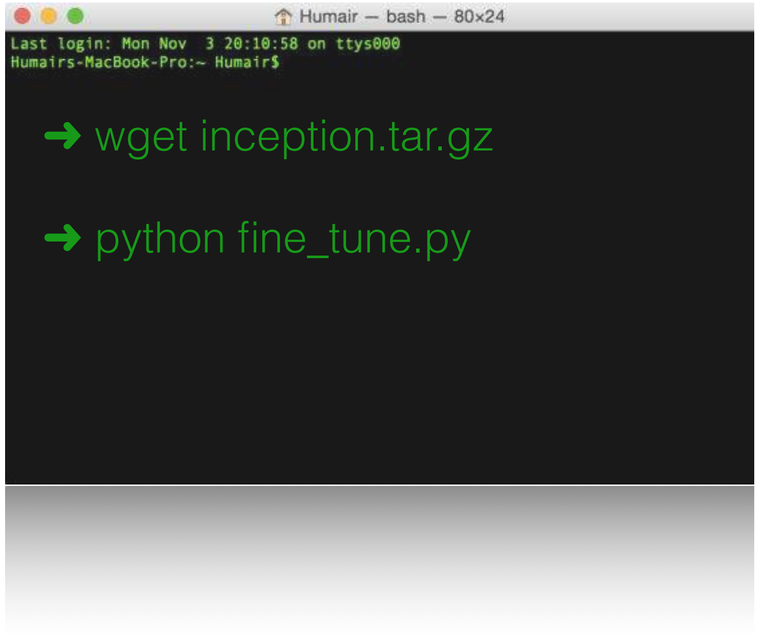
https://github.com/tensorflow/models/tree/master/inception



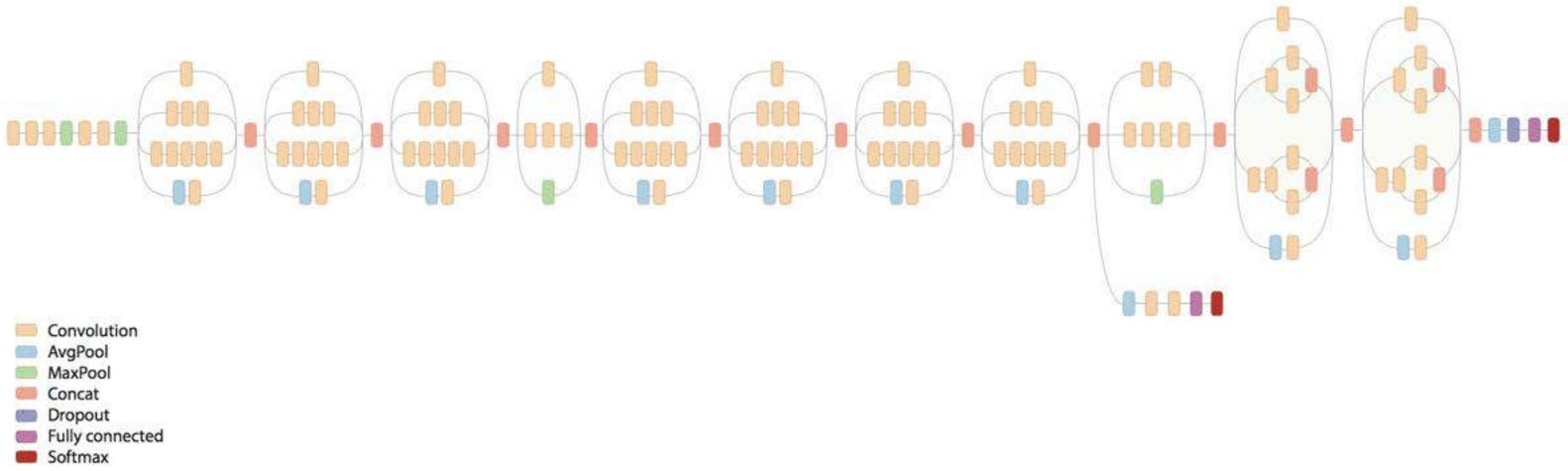
Case Study: Image Classification



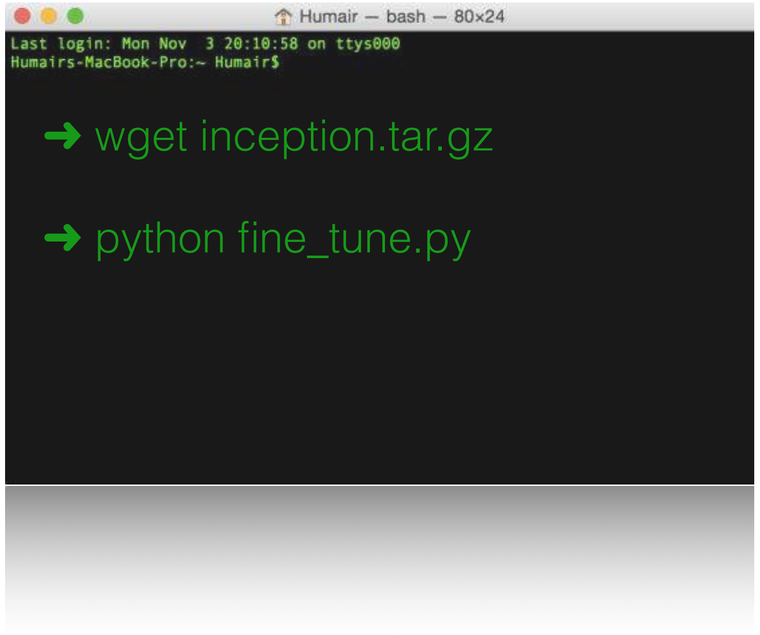
https://github.com/tensorflow/models/tree/master/inception



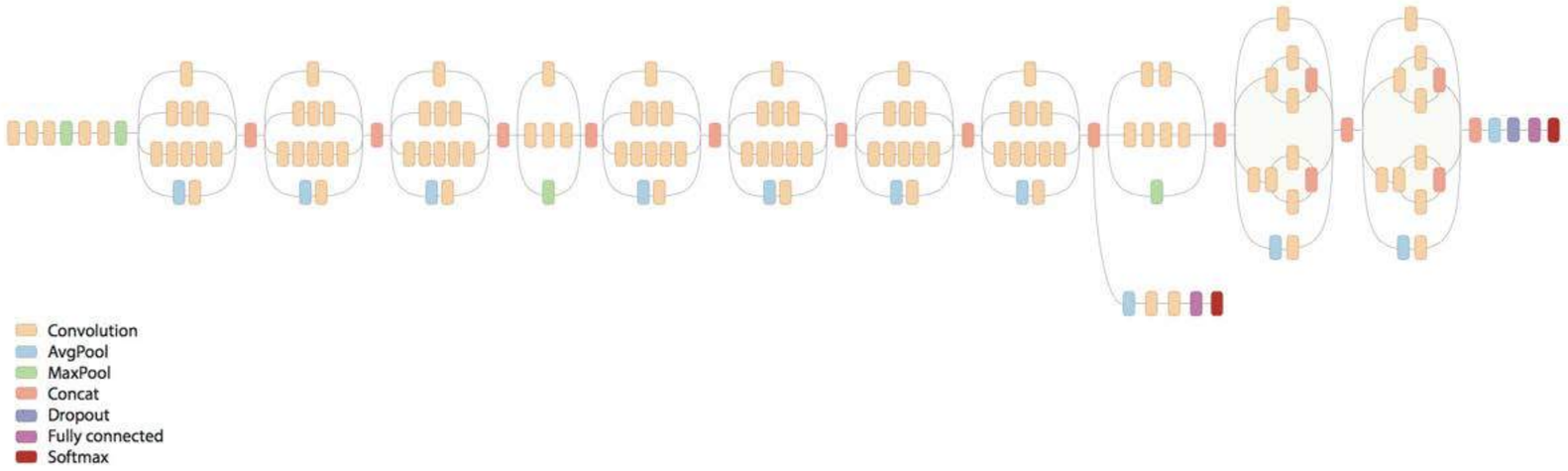
Case Study: Image Classification



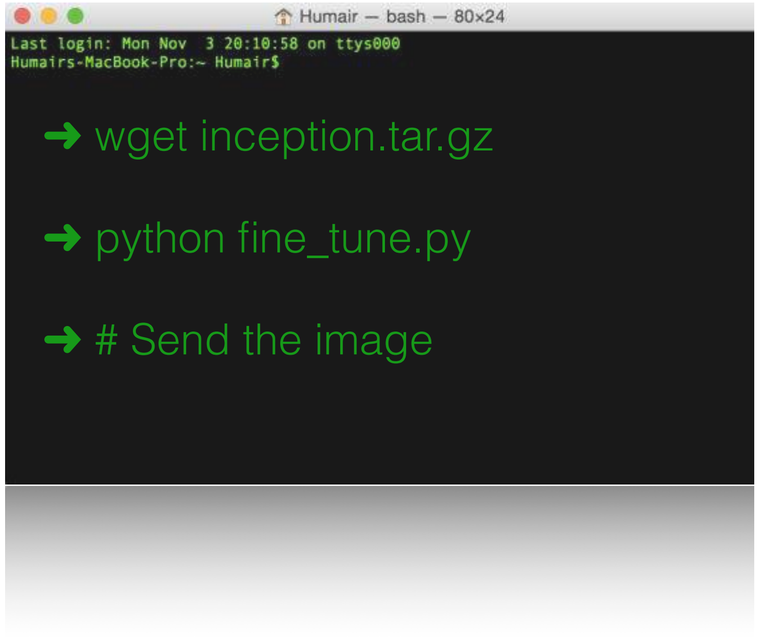
https://github.com/tensorflow/models/tree/master/inception



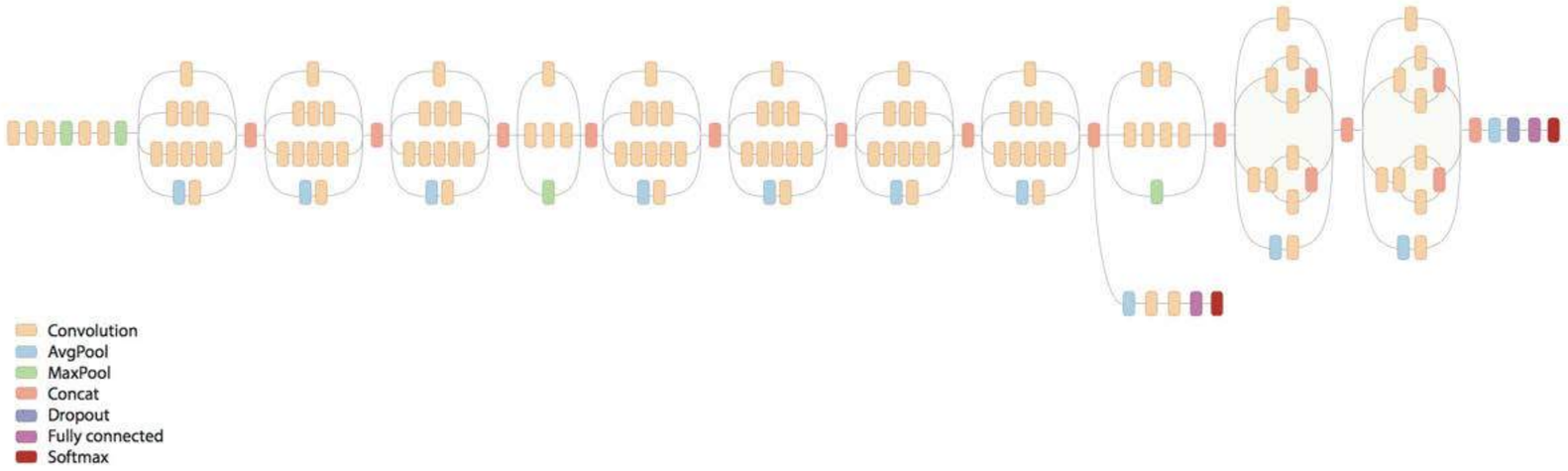
Case Study: Image Classification



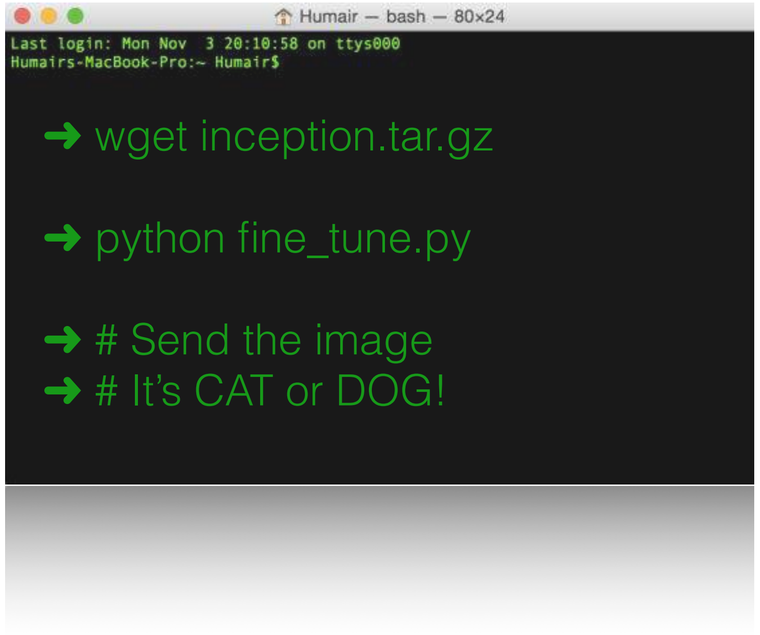
https://github.com/tensorflow/models/tree/master/inception



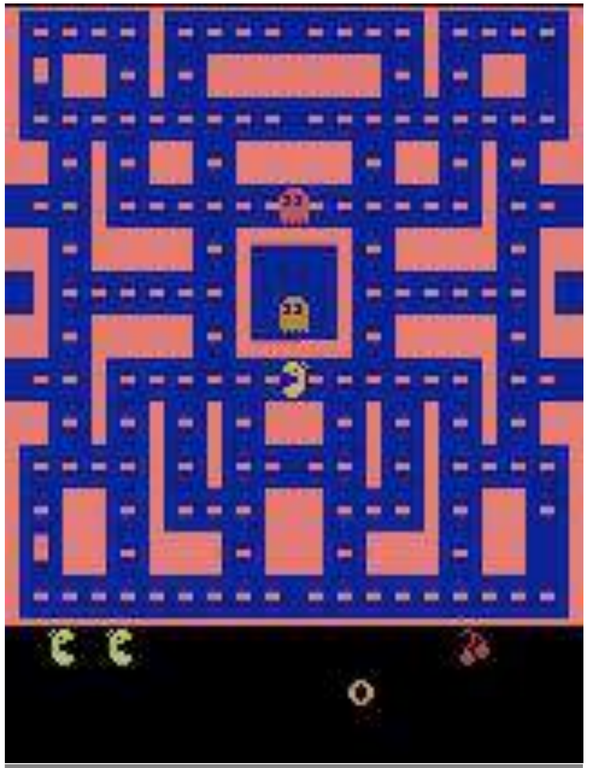
Case Study: Image Classification

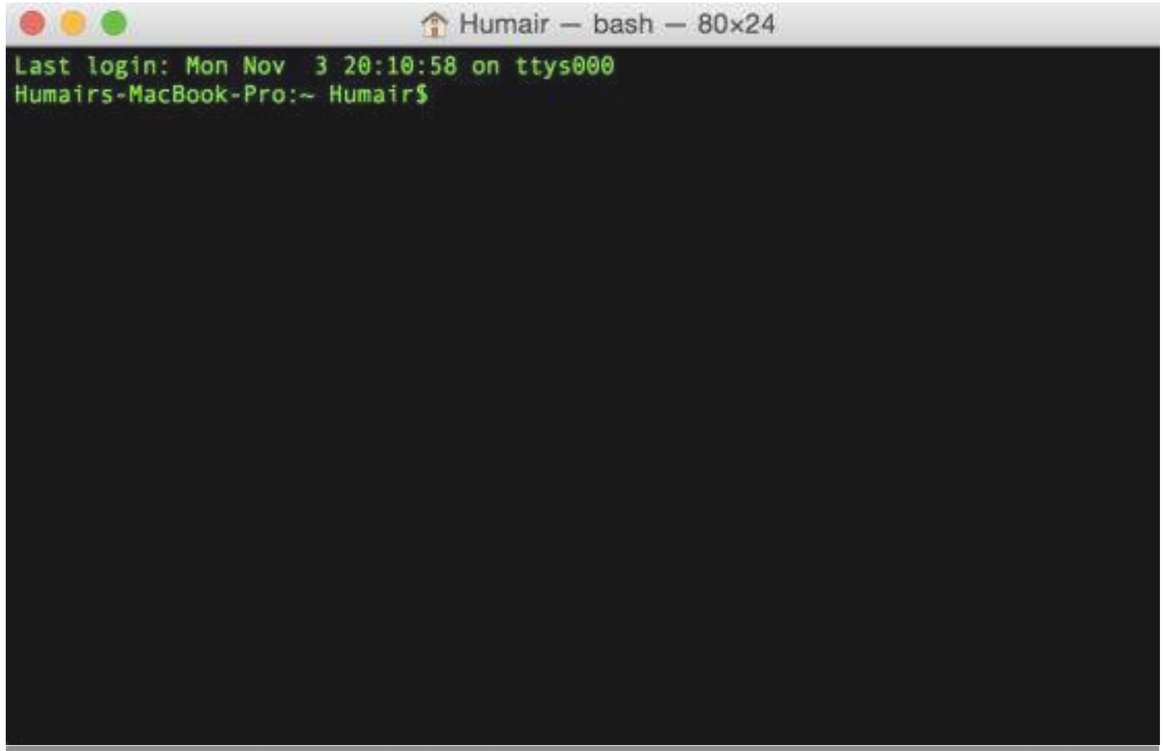


https://github.com/tensorflow/models/tree/master/inception

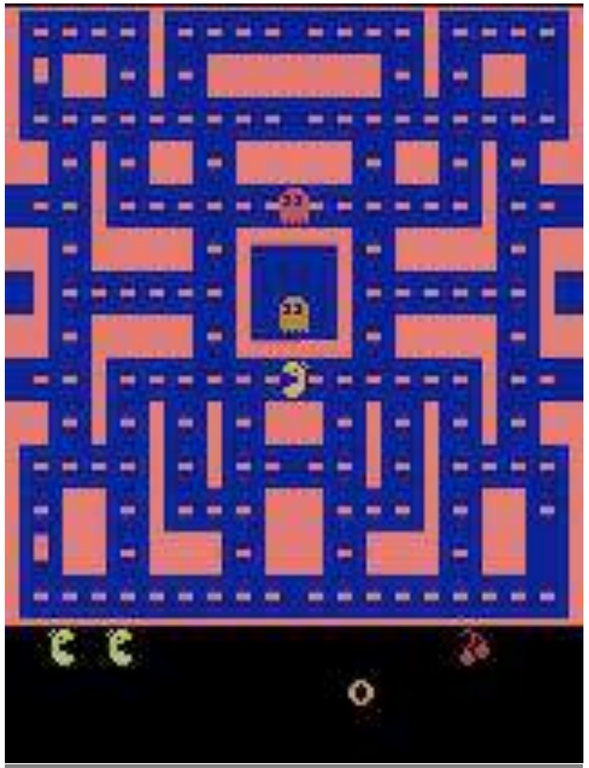


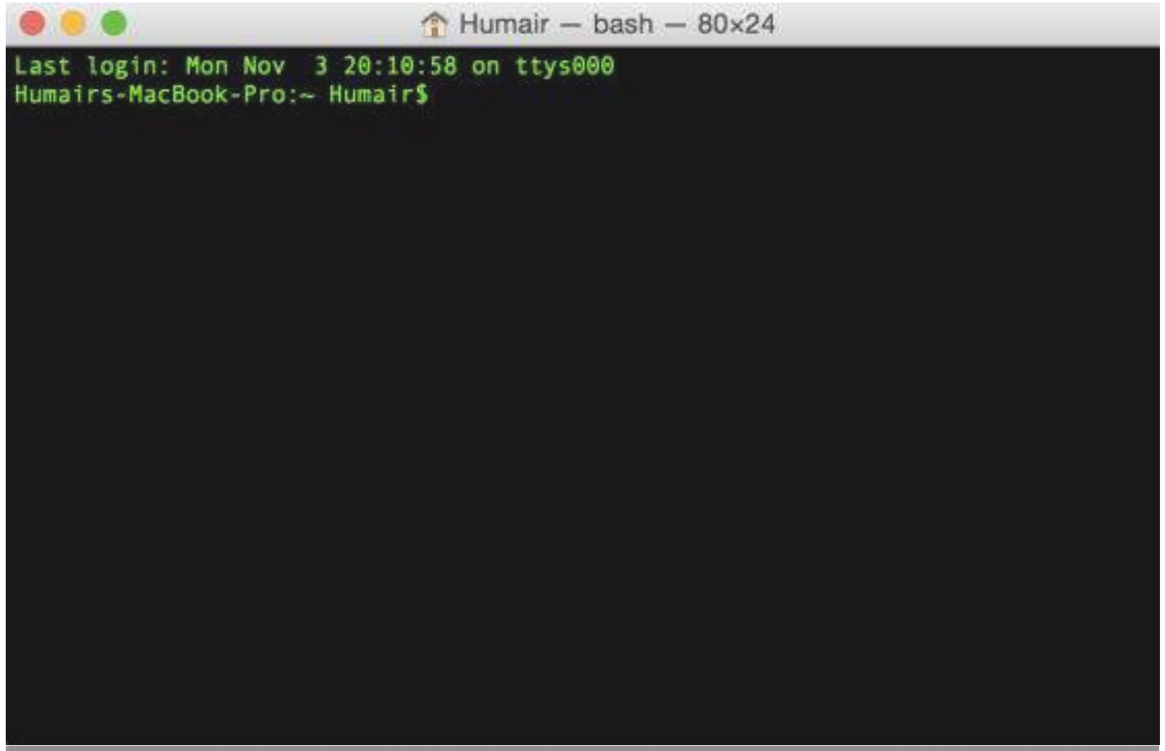
Case Study: Games AI



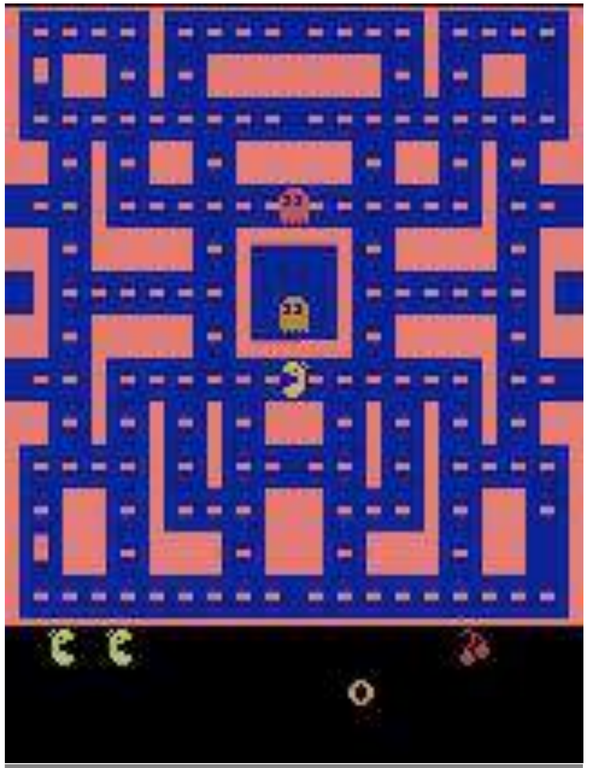


Case Study: Games AI





Case Study: Games AI

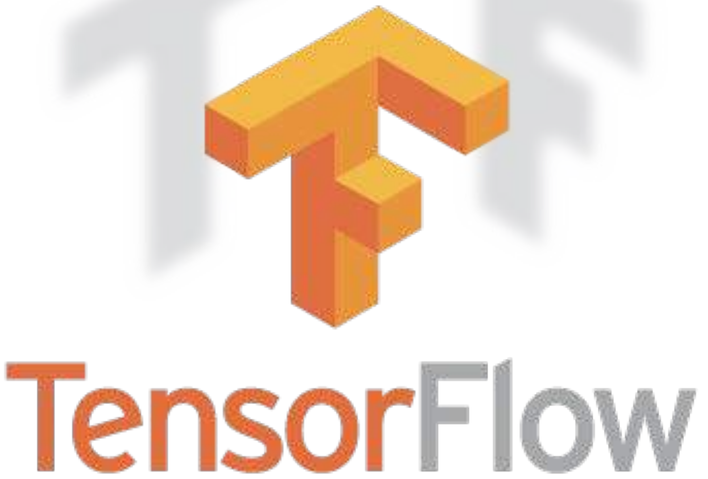


Humair — bash — 80×24

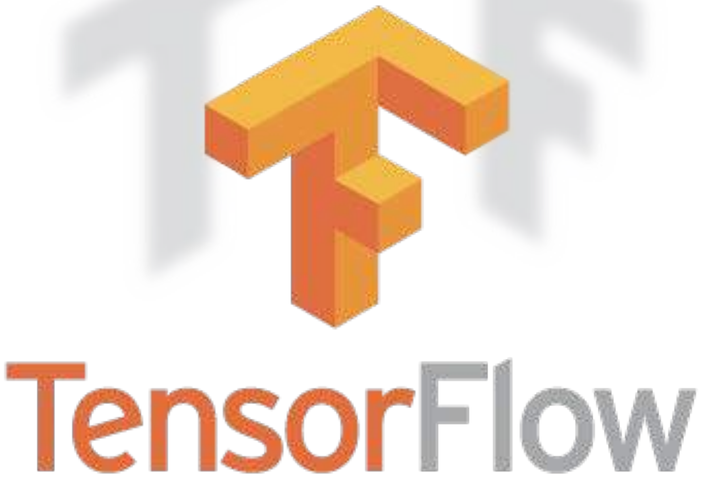
```
Last login: Mon Nov  3 20:10:58 on ttys000
Humairs-MacBook-Pro:~ Humair$
```

```
import gym

env = gym.make("MsPacman-v0")

observation = env.reset()

for _ in range(1000):
    env.render()
    action = env.action_space.sample()
    env.step(action)
```

Introduce TensorFlow

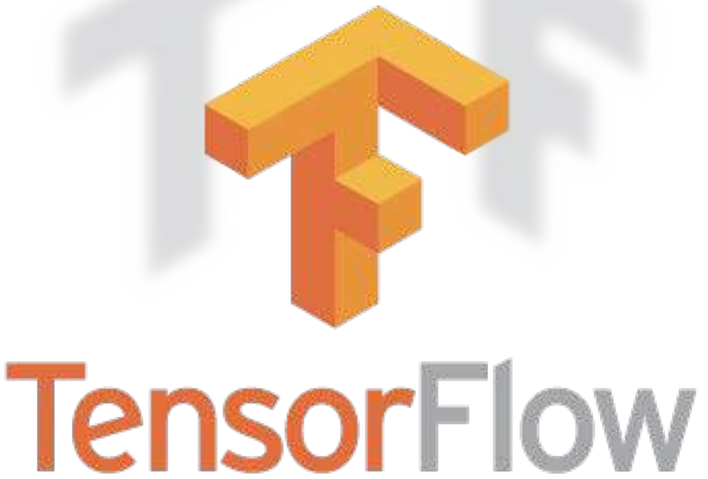


Introduce TensorFlow



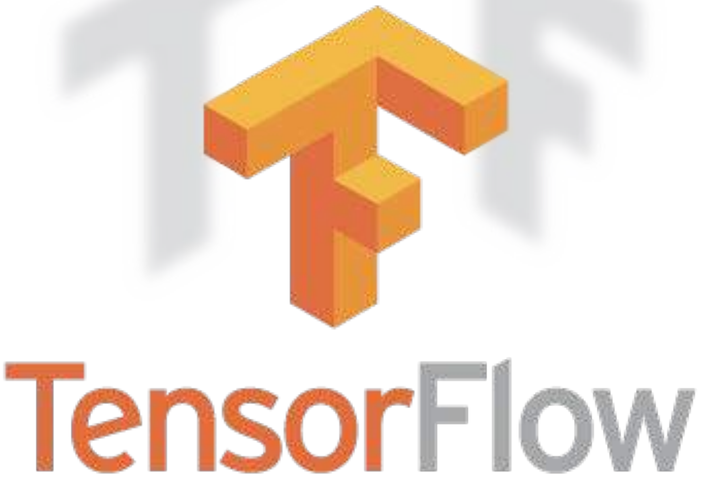
- Deep learning library

Introduce TensorFlow



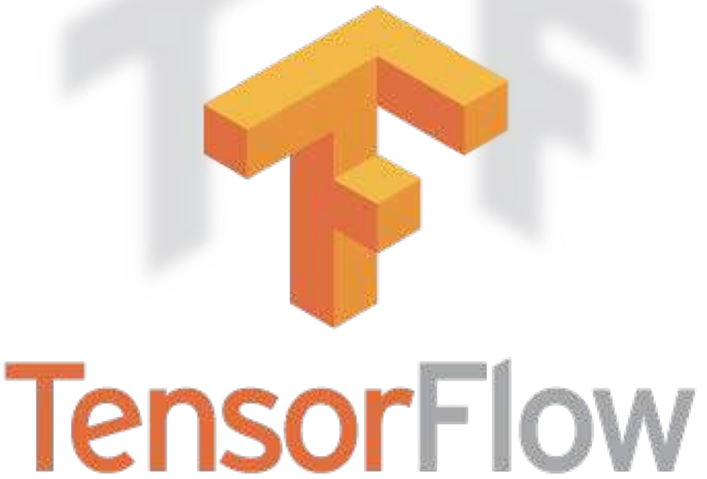
- Deep learning library
- C++/Python interfaces

Introduce TensorFlow



- Deep learning library
- C++/Python interfaces
- Training with CPU/GPU

Introduce TensorFlow



- Deep learning library
- C++/Python interfaces
- Training with CPU/GPU
- Distributed for large scale

Introduce TensorFlow

```
# 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))
```

Introduce TensorFlow

```
# 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))
```

Introduce TensorFlow

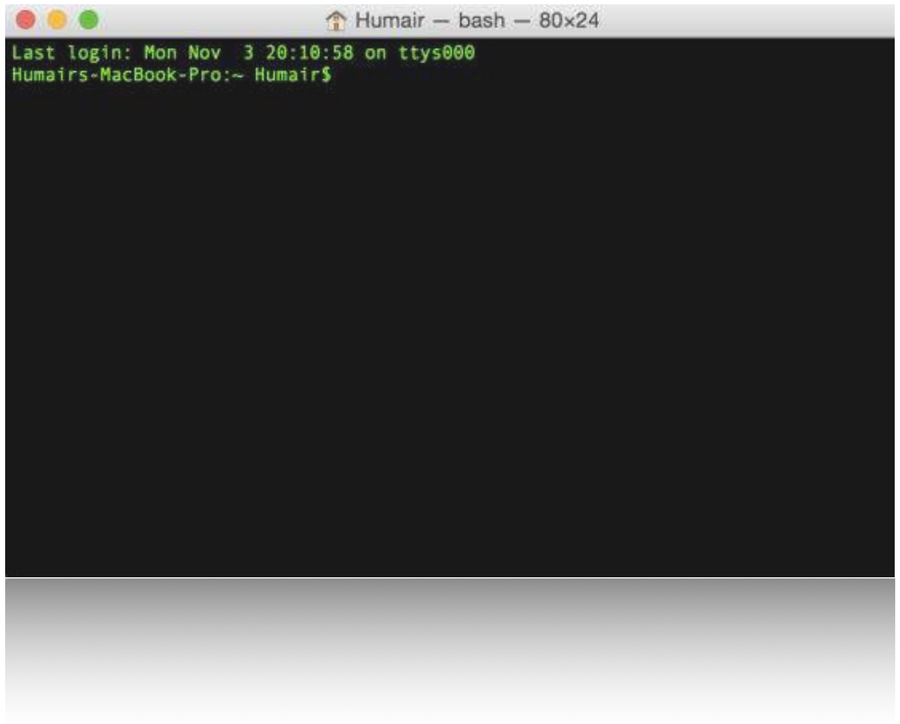
```
# Import the library
import tensorflow as tf

# 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))
```

Introduce TensorFlow

```
# 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))
```

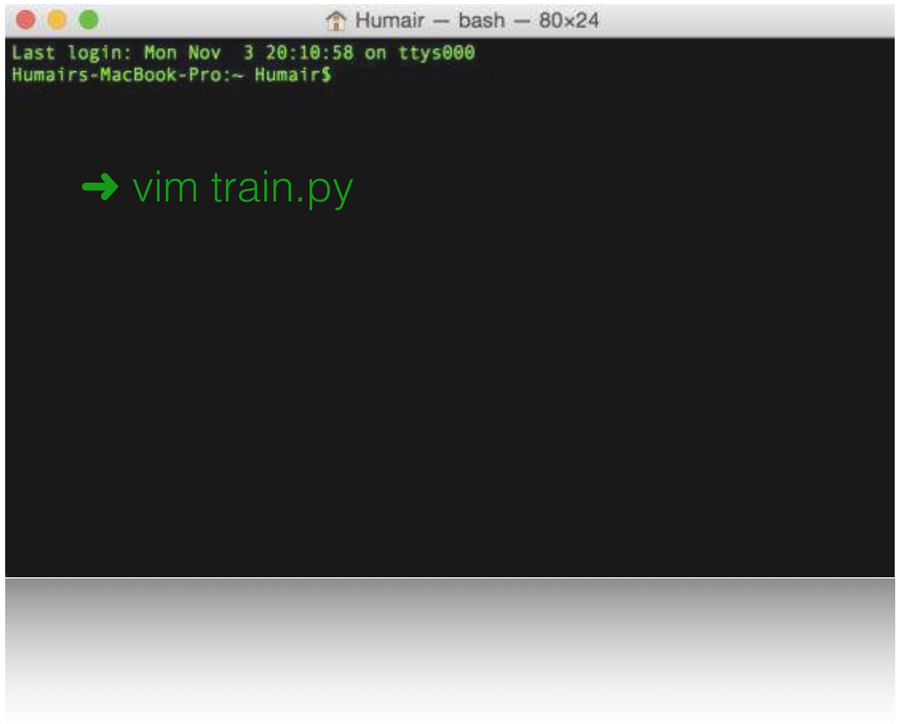
Introduce TensorFlow

```
# 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))
```



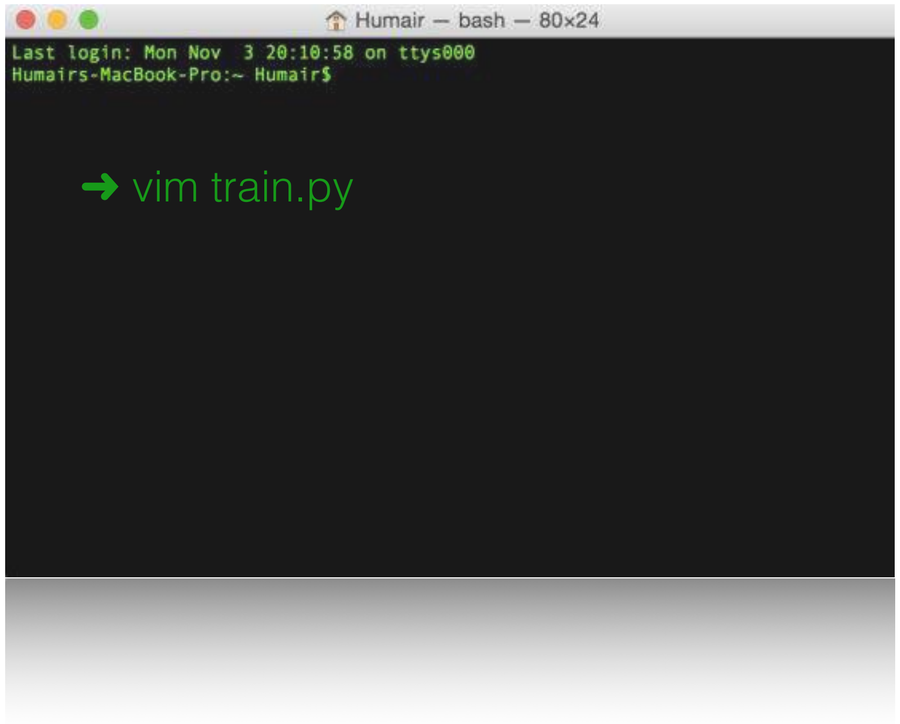
Introduce TensorFlow

```
# 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))
```



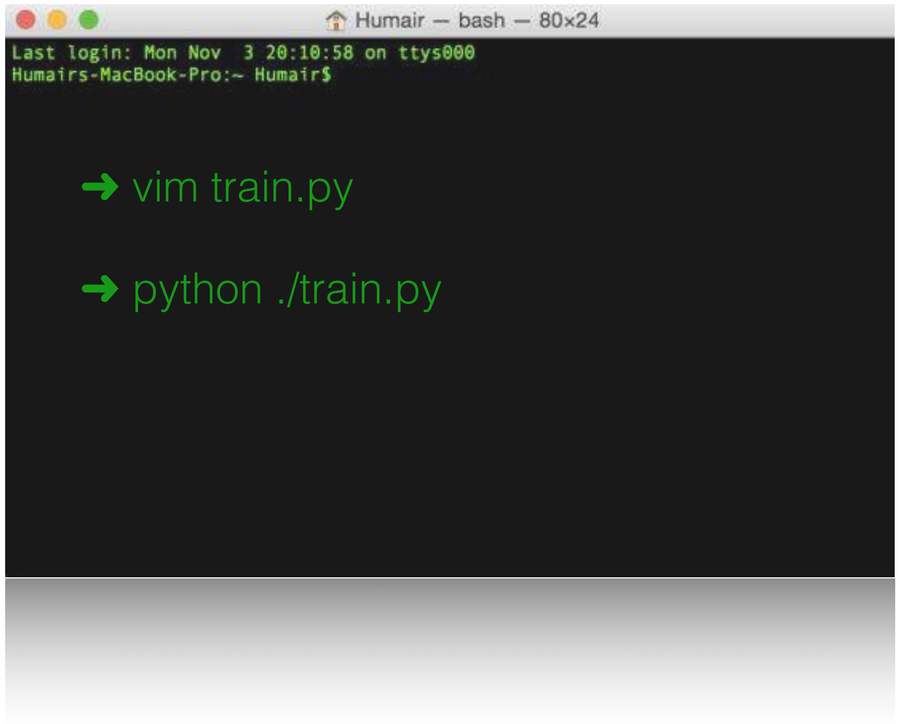
Introduce TensorFlow

```
# 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))
```



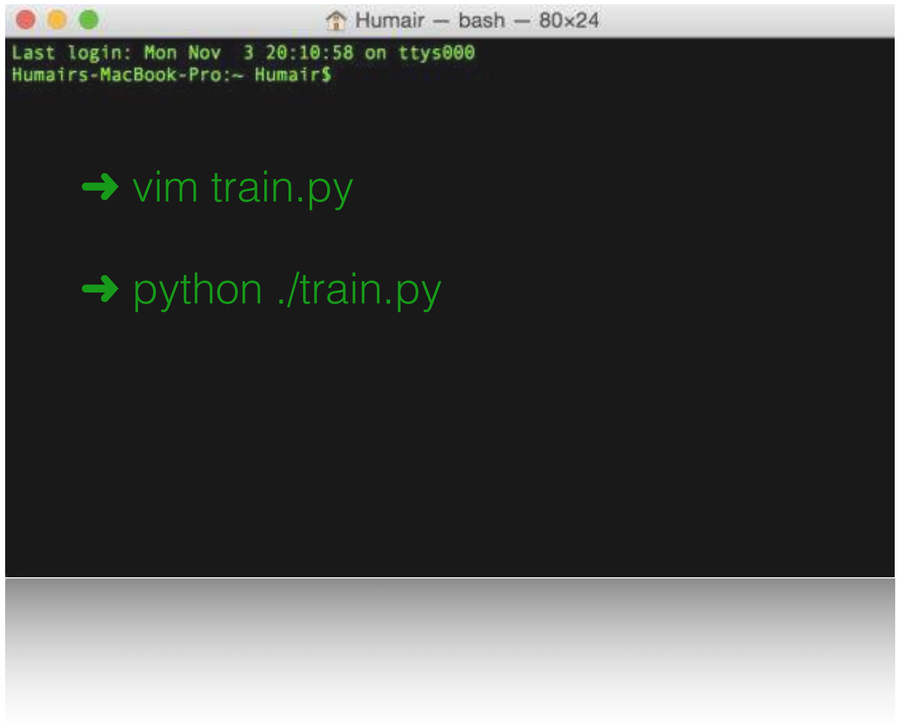
Introduce TensorFlow

```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))
```



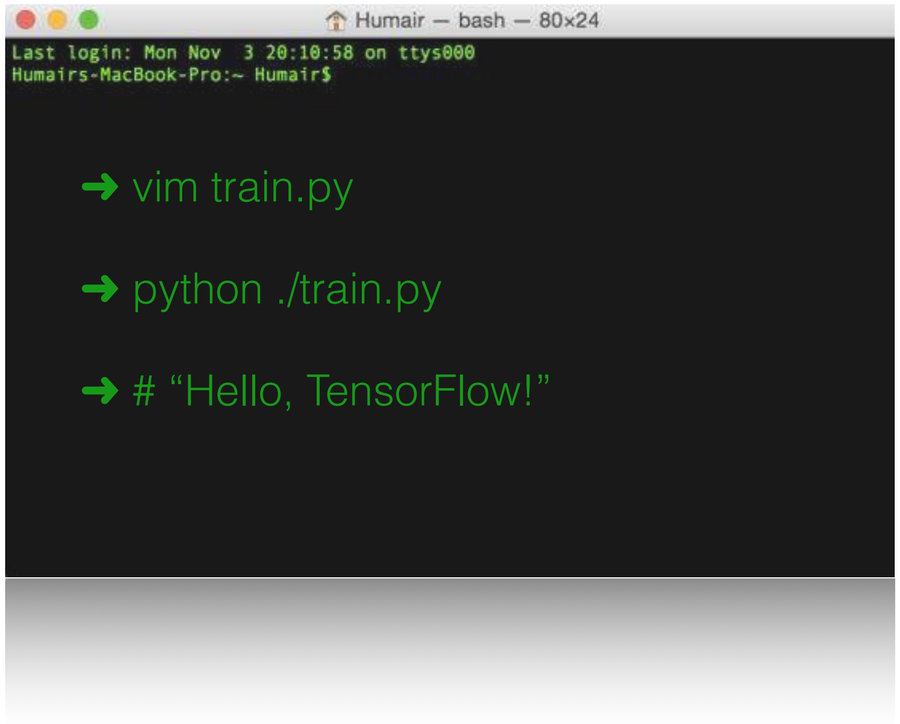
Introduce TensorFlow

```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))
```



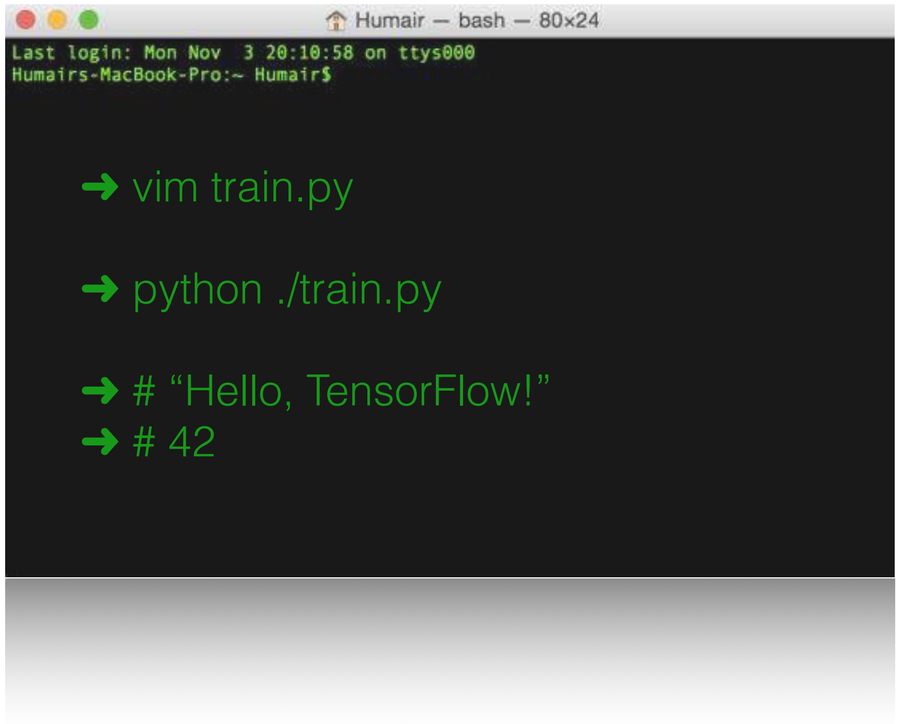
Introduce TensorFlow

```
# 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))
```



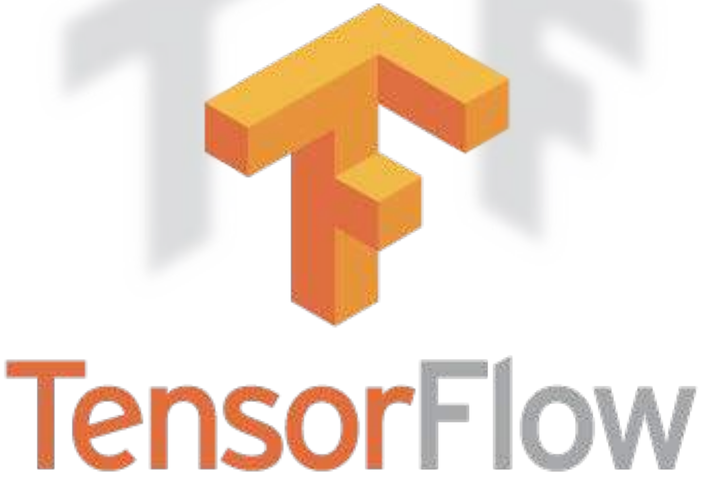
Introduce TensorFlow

```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))
```



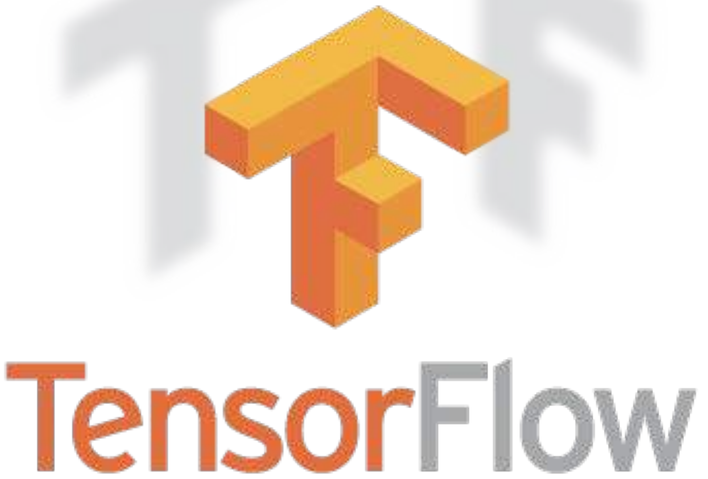
Summary of TensorFlow

Summary of TensorFlow



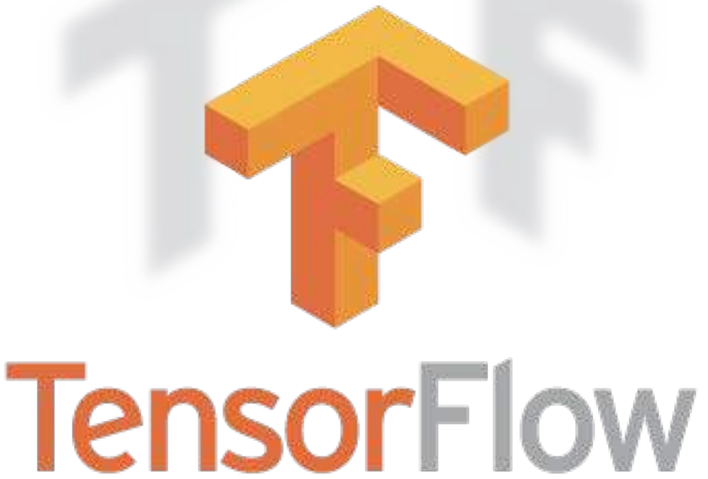
- TensorFlow is library

Summary of TensorFlow



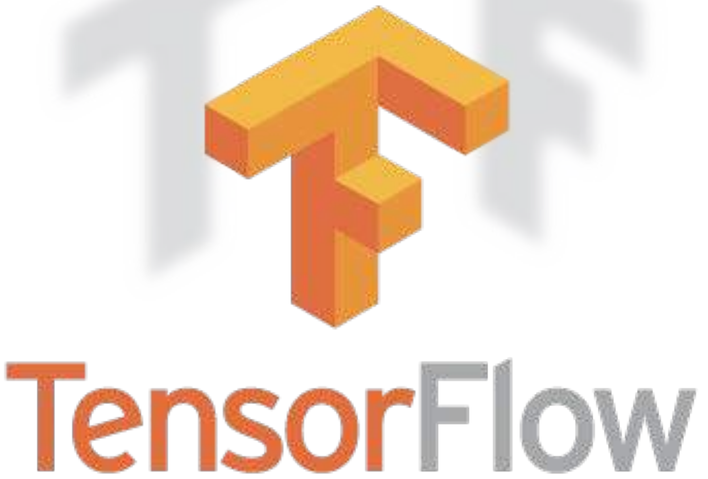
- TensorFlow is library
- Installed manually

Summary of TensorFlow



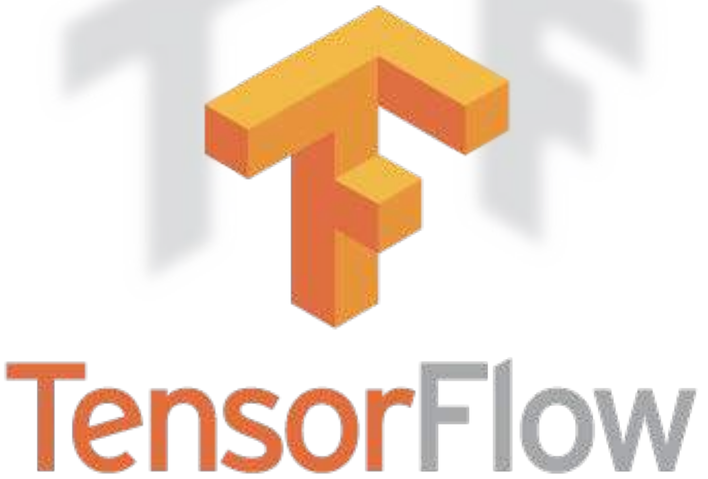
- TensorFlow is library
- Installed manually
- Configure manually

Summary of TensorFlow



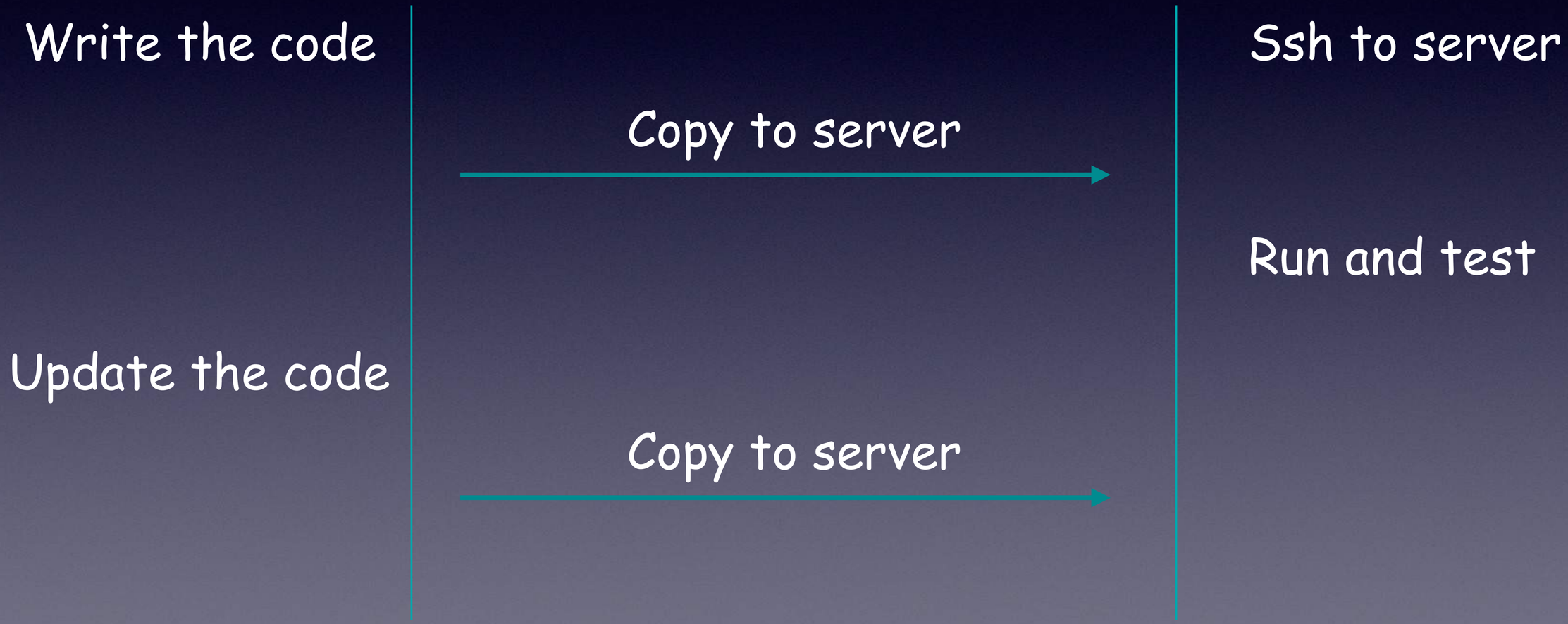
- TensorFlow is library
- Installed manually
- Configure manually
- Distributed manually

Summary of TensorFlow

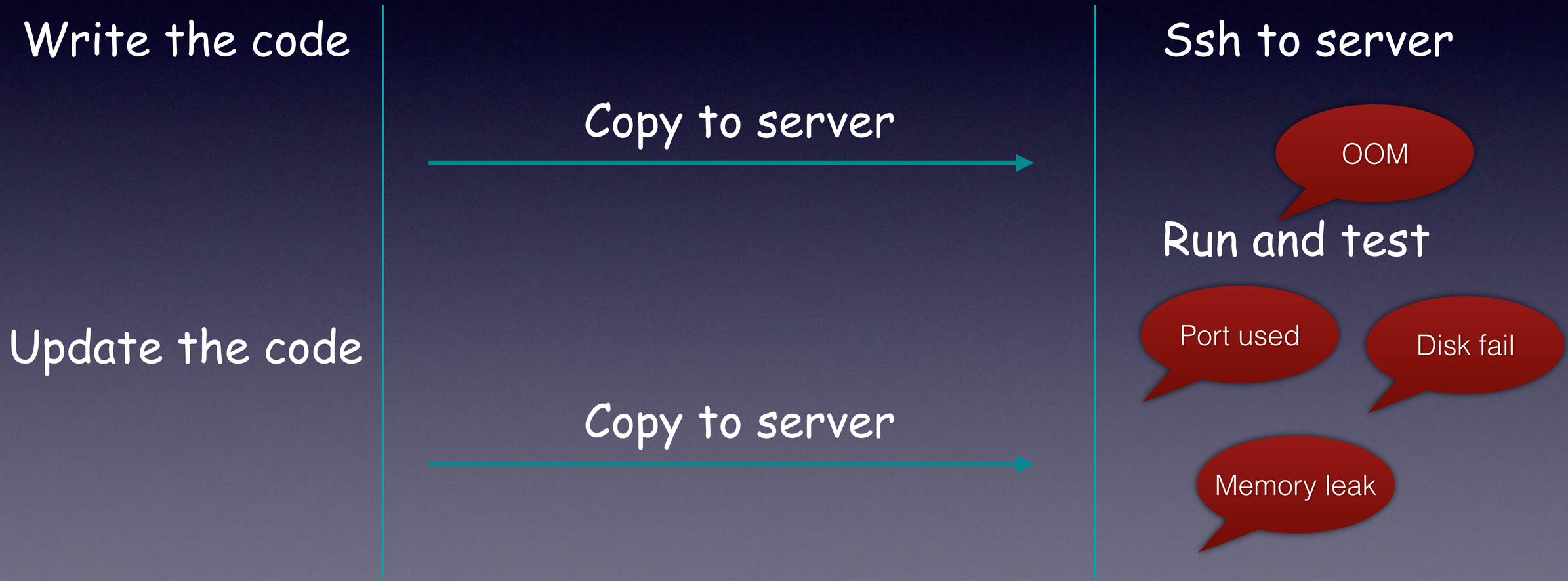


- TensorFlow is library
- Installed manually
- Configure manually
- Distributed manually
- Run and tune manually

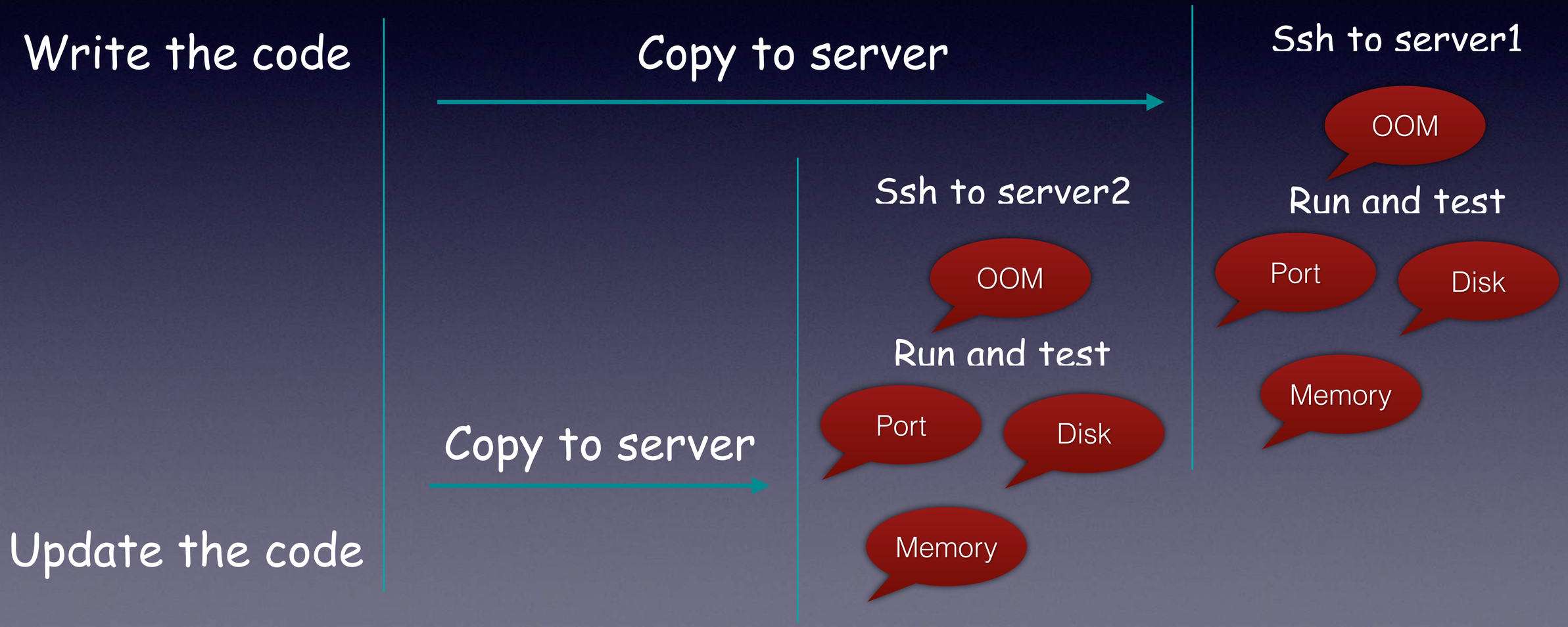
Standalone TensorFlow Workflow

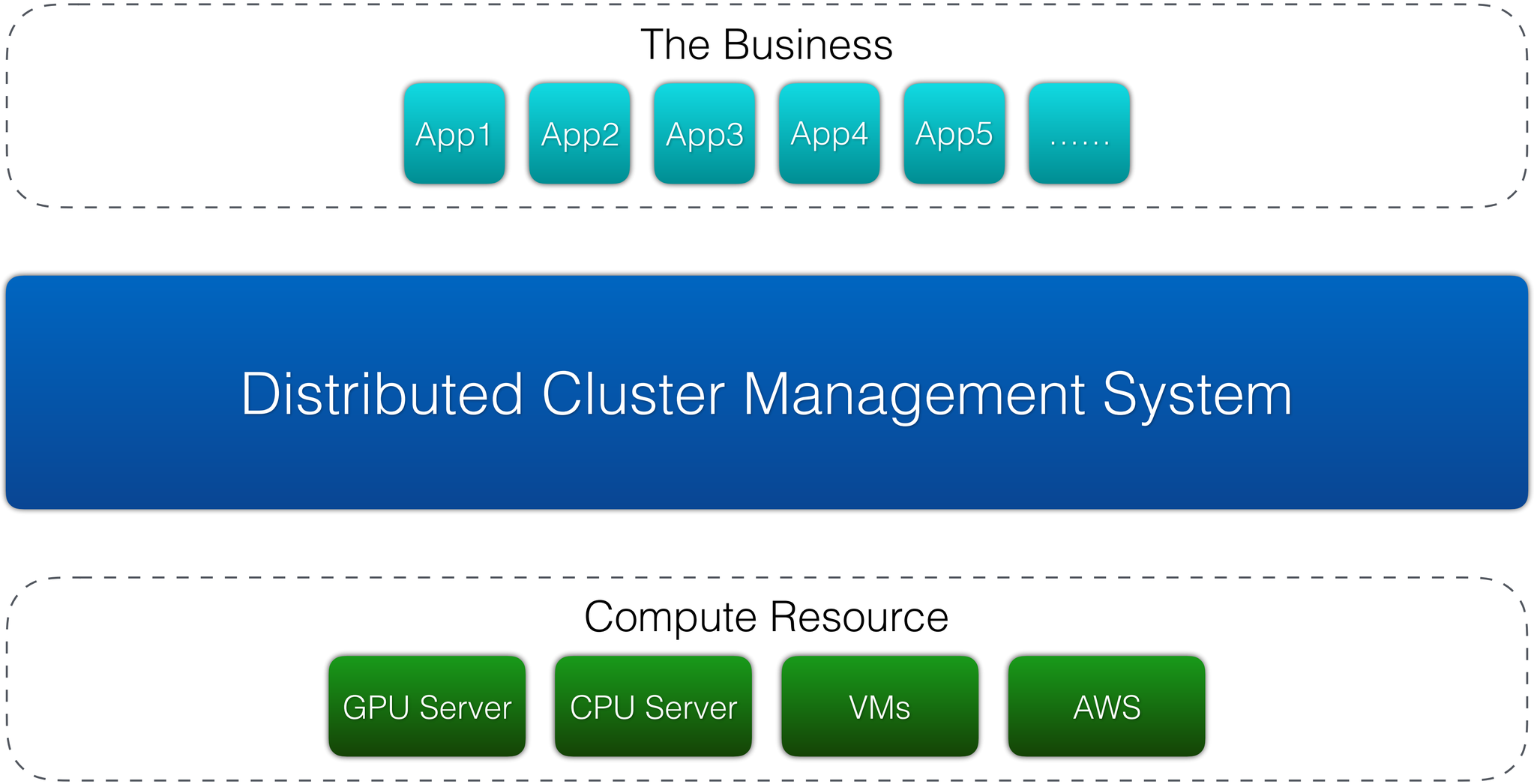


Standalone TensorFlow Workflow



Distributed TensorFlow Workflow




The Business
App1
App2
App3
App4
App5
.....
Distributed Cluster Management System
Compute Resource
GPU Server
CPU Server
VMs
AWS

Agenda

- 机器学习与深度学习应用
- **深度学习平台架构与设计**
- 深度学习平台应用与实践

Cloud-ml: The Principles

Cloud-ml: The Principles

- Cloud computing instead of bare metal

Cloud-ml: The Principles

- Cloud computing instead of bare metal
- Submit the jobs instead of ssh + script

Cloud-ml: The Principles

- Cloud computing instead of bare metal
- Submit the jobs instead of ssh + script
- Integrated with train and model service

Cloud-ml: The Principles

- Cloud computing instead of bare metal
- Submit the jobs instead of ssh + script
- Integrated with train and model service
- High availability and failover for train job

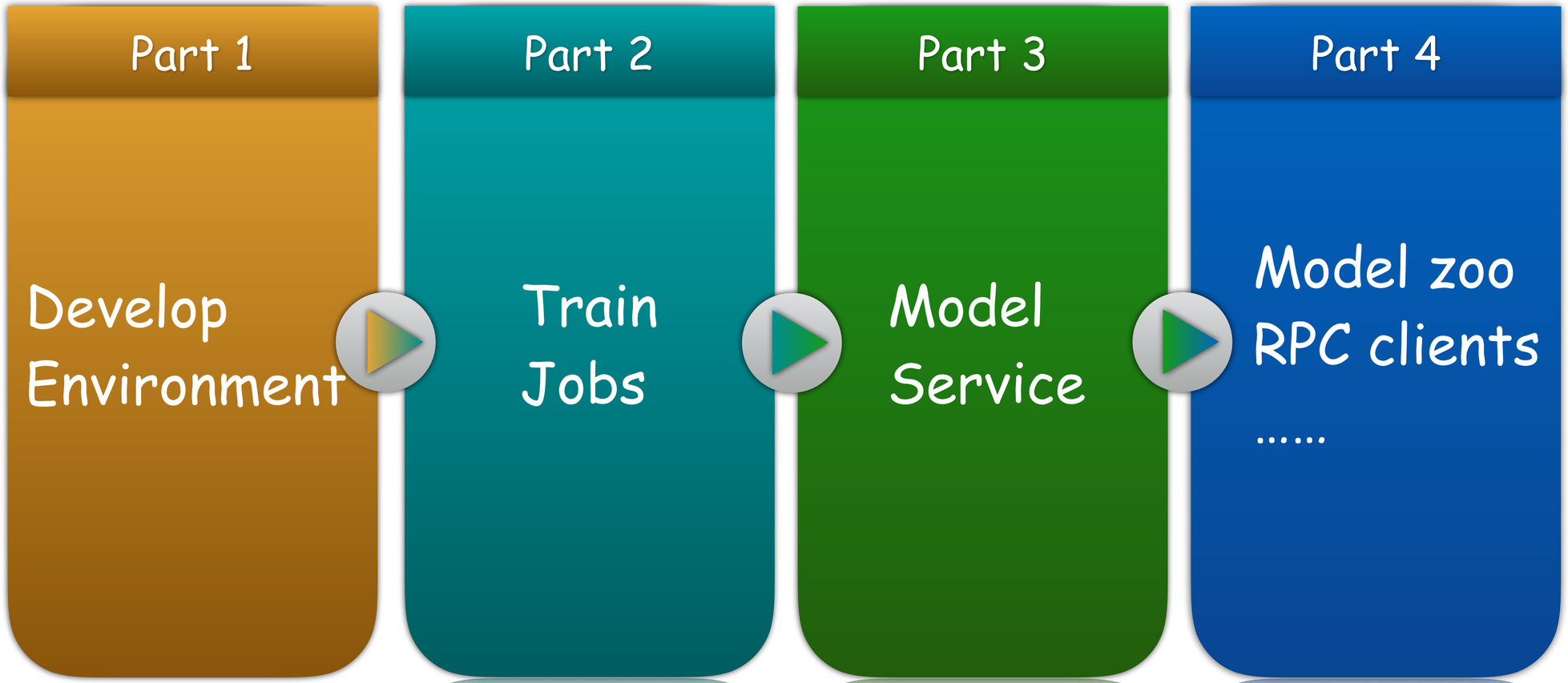
Cloud-ml: The Principles

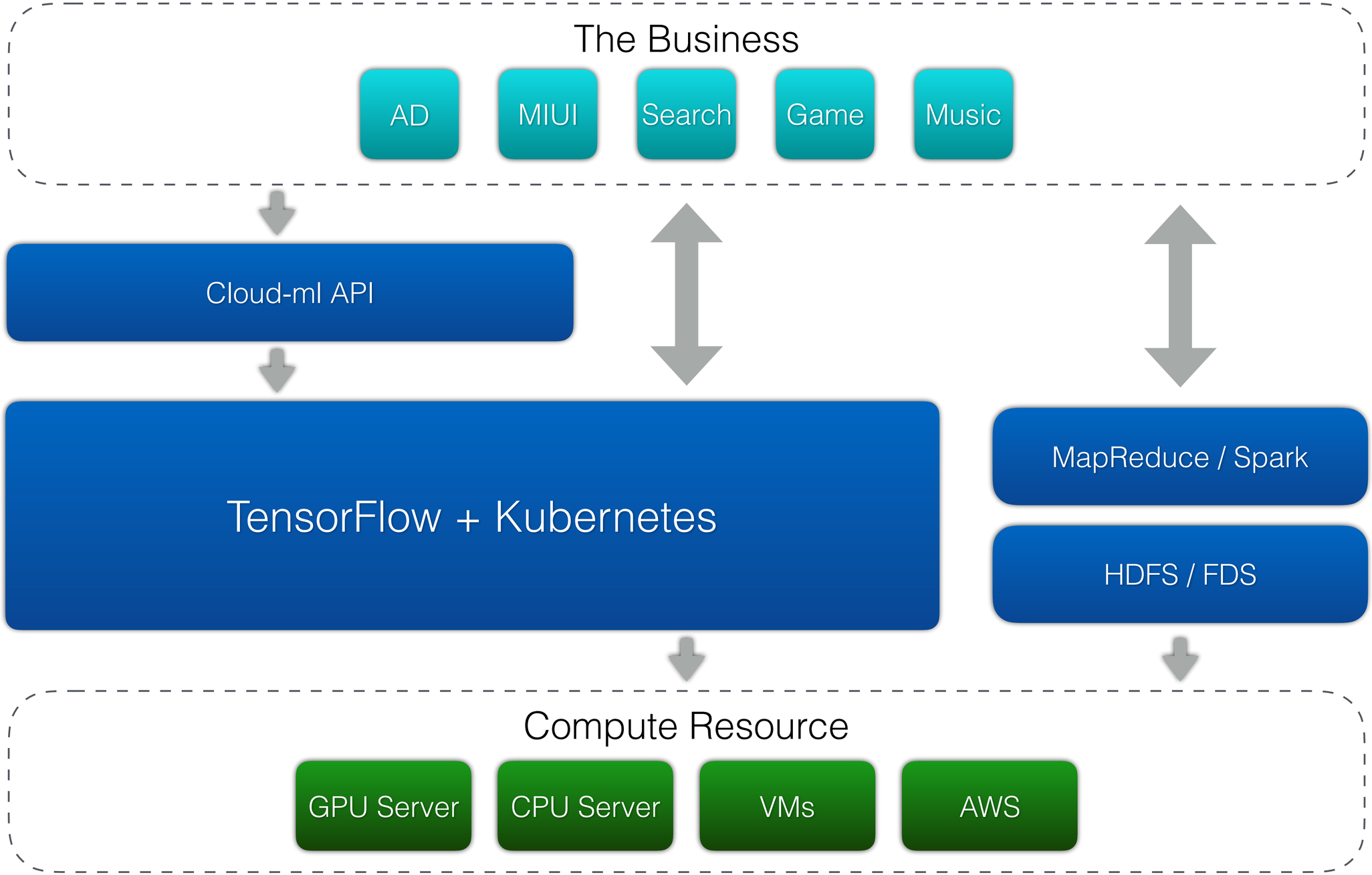
- Cloud computing instead of bare metal
- Submit the jobs instead of ssh + script
- Integrated with train and model service
- High availability and failover for train job
- Resource isolation and dynamic scheduling

Cloud-ml: The Principles

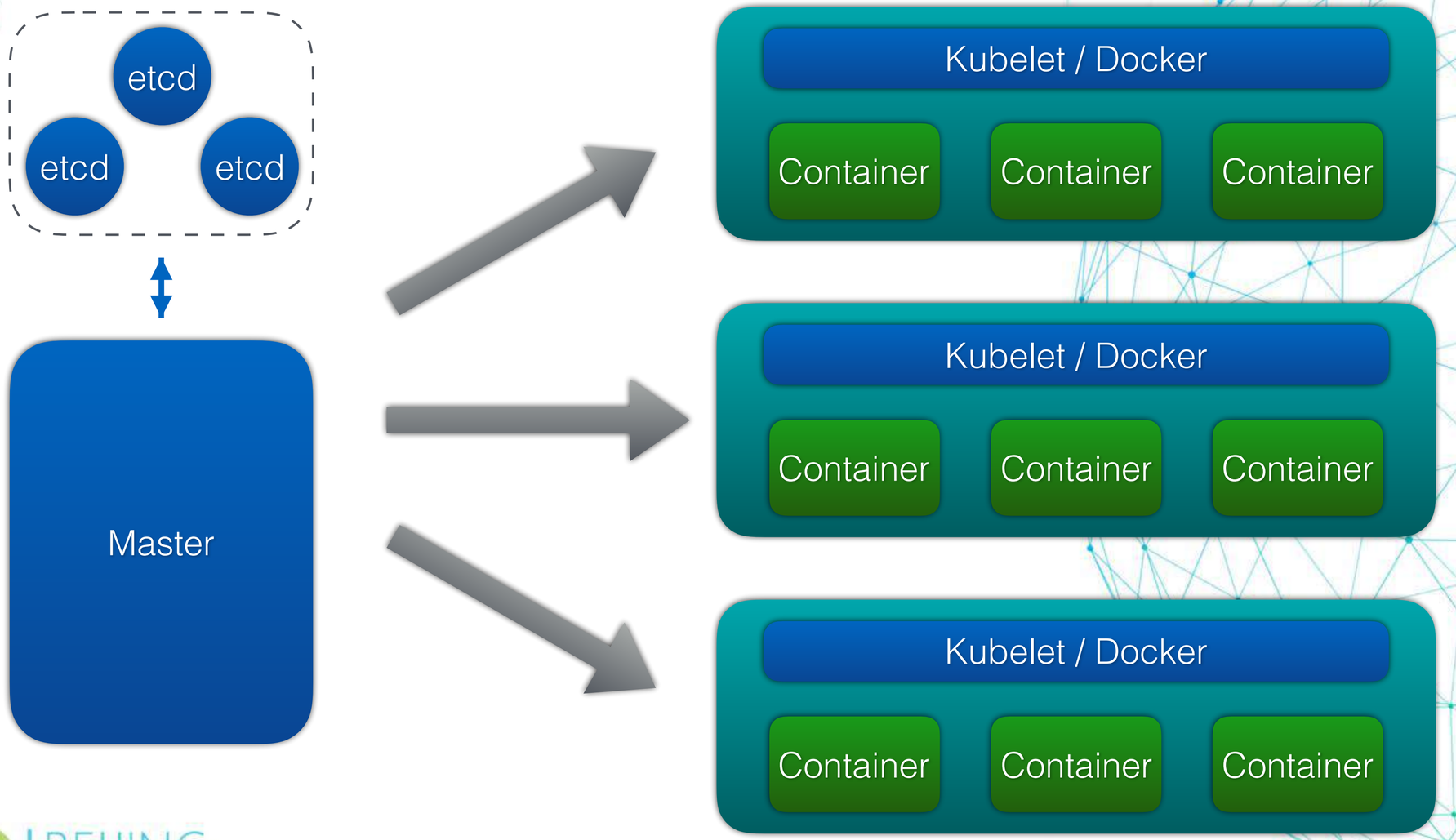
- Cloud computing instead of bare metal
- Submit the jobs instead of ssh + script
- Integrated with train and model service
- High availability and failover for train job
- Resource isolation and dynamic scheduling
- Parallel training and automatically tuning

Cloud-ml: All-in-one Platform




The Business
AD
MIUI
Search
Game
Music
Cloud-ml API
TensorFlow + Kubernetes
MapReduce / Spark
HDFS / FDS
Compute Resource
GPU Server
CPU Server
VMs
AWS

Cloud-ml: Kubernetes Inside



Cloud-ml: The Architecture