全国高校云计算大赛线下课程 TensorFlow 实验手册

实验一 手写数字识别程序

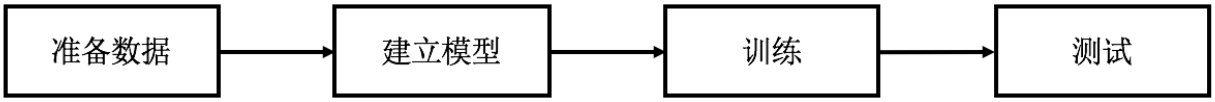
实验目的

1. 熟悉神经网络
2. 熟悉 python 语言
3. 熟悉 docker 命令
4. 掌握 TensorFlow 的张量，计算图，会话
5. 掌握使用 TensorFlow 构建神经网络解决手写数字识别问题

实验要求

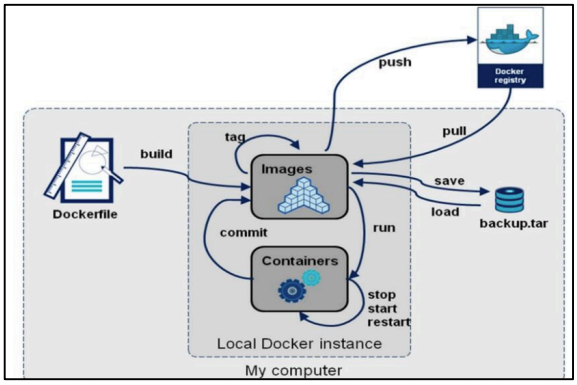
1. vi 编辑器编写 tensorflow 程序
2. 使用 docker 搭建 tensorflow 环境
3. 在容器中运行程序

实验原理



实验步骤

1. Load docker image into docker local image registration.
   docker load -i [docker-image-file]



2. Run a docker contrainer to build our lab environment.

```
# docker run -it --name lab1 slic/tensorflow:1.1
```

3. Coding

- datasets.py

```python
# !/bash/bin/env python
# -*- coding: utf-8 -*-

from tensorflow.examples.tutorials.mnist import input_data

path_to_mnist_data = '/root/data'
mnist = input_data.read_data_sets(path_to_mnist_data, one_hot=True)
```

- graph.py

```python
# !/bash/bin/env python
# -*- coding: utf-8 -*-

import tensorflow as tf
from datasets import mnist

# ---------- 参数的配置 ----------
# ---------- 输入层的节点数，图片的像素 ----------
INPUT_NODE = 784
# ---------- 输出层的节点数，图片表示的数字（0-9） ----------
OUTPUT_NODE = 10

# ---------- 隐藏层的节点数 ----------
LAYER1_NODE = 500
# ---------- 每次训练的数据个数 ----------
BATCH_SIZE = 100

# ---------- 基础学习率 ----------
LEARNING_RATE_BASE = 0.8
# ---------- 学习率的衰减率 ----------
LEARNING_RATE_DECAY = 0.99

# ---------- 正则化系数 ----------
REGULARAZTION_RATE = 0.0001
# ---------- 训练的总步数 ----------
TRAINING_STEPS = 5000
```

```
# ---------- 定义数据节点 ----------
x =
y_ =

# ---------- 定义隐藏层的参数 ----------
w1 =
b1 =

w2 =
b2 =

# ---------- f 映射关系 ----------
layer1 =
y = tf.matmul(layer1, w2) + b2

global_step = tf.Variable(0, trainable=False)

# ---------- 交叉熵损失函数 ----------
cross_entropy =

# ---------- 计算交叉熵的平均值 ----------
cross_entropy_mean = tf.reduce_mean(cross_entropy)

# ---------- L2 正则化损失函数 ----------
regularizer = tf.contrib.layers.l2_regularizer(REGULARAZTION_RATE)
regularaztion = regularizer(w1) + regularizer(w2)

# ---------- 总损失 ----------
loss = cross_entropy_mean + regularaztion

# ---------- 设置衰减的学习率 ----------
learning_rate =

# ---------- 使用梯度下降优化器来优化损失函数 ----------
train_step =

correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

- **train.py**

```
# !/bash/bin/env python
# -*- coding: utf-8 -*-
```

```
import tensorflow as tf

from datasets import mnist
from graph import TRAINING_STEPS, BATCH_SIZE, x, y_, loss, train_step,
accuracy

with tf.Session() as sess:
    # --------- 初始化变量 ---------
    init_op = tf.global_variables_initializer()
    sess.run(init_op)

    validate_feed = {
        x: mnist.validation.images,
        y_: mnist.validation.labels
    }

    test_feed = {
        x: mnist.test.images,
        y_: mnist.test.labels
    }

    for i in range(TRAINING_STEPS):
        if i % 1000 == 0:
            validate_acc = sess.run(accuracy,
                                    feed_dict=validate_feed)
            print("After %d training step(s), validation accuracy "
                  "using average model is %g" % (i, validate_acc))

        xs, ys = mnist.train.next_batch(BATCH_SIZE)
        sess.run(train_step,
                 feed_dict={x: xs, y_: ys})

    test_acc = sess.run(accuracy,
                        feed_dict=test_feed)
    print("After %d traing step(s), test accuracy using average "
          "model is %g" % (TRAINING_STEPS, test_acc))
```

4. Run & test

```
# docker exec -it [container id] /bin/bash
# python /path/to/train.py
```

全国高校云计算大赛线下课程 TensorFlow 实验手册

实验一 手写数字识别程序

实验结果

```
root@2bd7d031d1f2:/tmp/lab1# python train.py
Extracting /root/data/train-images-idx3-ubyte.gz
Extracting /root/data/train-labels-idx1-ubyte.gz
Extracting /root/data/t10k-images-idx3-ubyte.gz
Extracting /root/data/t10k-labels-idx1-ubyte.gz
After 0 training step(s), validation accuracy using average model is 0.0784
After 1000 training step(s), validation accuracy using average model is 0.973
After 2000 training step(s), validation accuracy using average model is 0.9766
After 3000 training step(s), validation accuracy using average model is 0.9782
After 4000 training step(s), validation accuracy using average model is 0.983
After 5000 traing step(s), test accuracy using average model is 0.9818
```