人工智能实践：TensorFlow 笔记

第四讲 功能扩展

本讲目标：神经网络八股功能扩展

一、回顾

1、tf.keras 搭建神经网络八股——六步法

1) import——导入所需的各种库和包

2) x_train, y_train——导入数据集、自制数据集、数据增强

3) model=tf.keras.models.Sequential

/class MyModel(Model) model=MyModel——定义模型

4) model.compile——配置模型

5) model.fit——训练模型、断点续训

6) model.summary——参数提取、acc/loss 可视化、前向推理实现应用

2、代码 mnist_train_baseline.py：

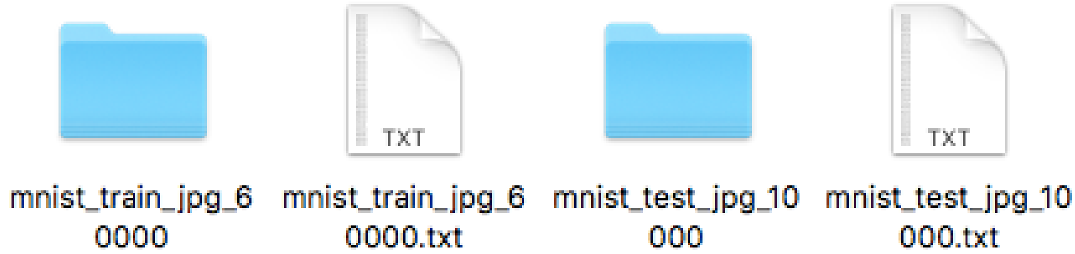
```python
import tensorflow as tf

mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
                                    tf.keras.layers.Flatten(input_shape=(28, 28)),
                                    tf.keras.layers.Dense(128, activation='relu'),
                                    tf.keras.layers.Dense(10, activation='softmax')
                                    ])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

model.fit(x_train, y_train, batch_size=32, epochs=10, validation_data=(x_test,y_test),validation_freq=1)
model.summary()
```

二、本讲用 tf.keras 完善功能模块

1、自制数据集，应对特定应用

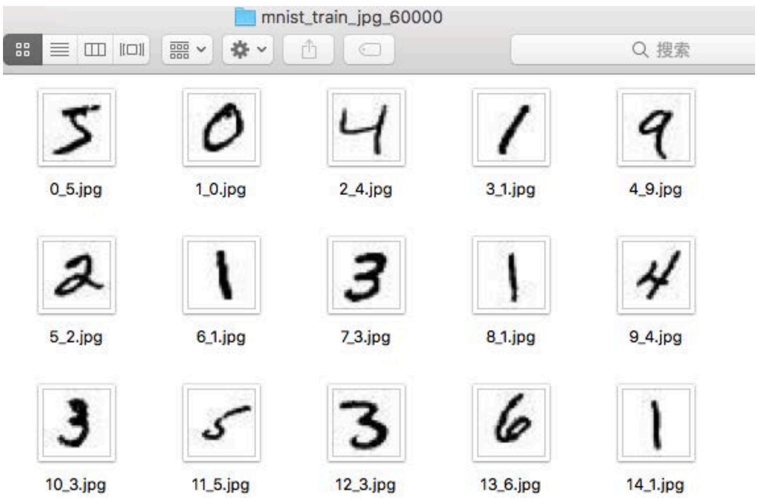
1.1、观察数据集数据结构，配成特征标签对

mnist_image_label 文件夹：



四个文件分别对应为训练集图片、训练集标签、测试集图片、测试集标签

图片文件夹：



标签文件：

```
28755_0.jpg 0
13360_5.jpg 5
57662_5.jpg 5
21455_5.jpg 5
59351_5.jpg 5
39461_3.jpg 3
22720_5.jpg 5
52282_6.jpg 6
10872_3.jpg 3
17077_3.jpg 3
23650_8.jpg 8
56239_0.jpg 0
26079_8.jpg 8
52645_6.jpg 6
12727_2.jpg 2
```

代码 mnist_train_ex1.py：

```python
import tensorflow as tf
from PIL import Image
import numpy as np

train_path = './mnist_image_label/mnist_train_jpg_60000/'
train_txt = './mnist_image_label/mnist_train_jpg_60000.txt'

test_path = './mnist_image_label/mnist_test_jpg_10000/'
test_txt = './mnist_image_label/mnist_test_jpg_10000.txt'

def generateds(path, txt):
    f = open(txt, 'r')
    contents = f.readlines()  # 按行读取
    f.close()
    x, y_ = [], []
    for content in contents:
        value = content.split()  # 以空格分开，存入数组
        img_path = path + value[0]
        img = Image.open(img_path)
        img = np.array(img.convert('L'))
        img = img / 255.
        x.append(img)
        y_.append(value[1])
        print('loading : ' + content)

    x = np.array(x)
    y_ = np.array(y_)
    y_ = y_.astype(np.int64)
    return x, y_

print('-------------Generate Data sets-----------------')
x_train, y_train = generateds(train_path, train_txt)
x_test, y_test = generateds(test_path, test_txt)
```

def generateds(输入特征路径,标签路径文件名)

读取图片

存入数组

调整格式

返回输入特征和标签

获得训练集和测试集数据

```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

model.fit(x_train, y_train, batch_size=32, epochs=5, validation_data=(x_test, y_test), validation_freq=1)
model.summary()
```

2、数据增强，增大数据量

2.1、数据增强（增大数据量）

image_gen_train=tf.keras.preprocessing.image.ImageDataGenerator(增 强 方法)

image_gen_train.fit(x_train)

常用增强方法：

缩放系数：rescale=所有数据将乘以提供的值

随机旋转：rotation_range=随机旋转角度数范围

宽度偏移：width_shift_range=随机宽度偏移量

高度偏移：height_shift_range=随机高度偏移量

水平翻转：horizontal_flip=是否水平随机翻转

随机缩放：zoom_range=随机缩放的范围［1-n，1+n]

例：image_gen_train = ImageDataGenerator(

rescale=1./255, #原像素值 0～255 归至 0～1

rotation_range=45, #随机 45 度旋转

width_shift_range=.15, #随机宽度偏移 [-0.15,0.15)

height_shift_range=.15, #随机高度偏移 [-0.15,0.15)

horizontal_flip=True, #随机水平翻转

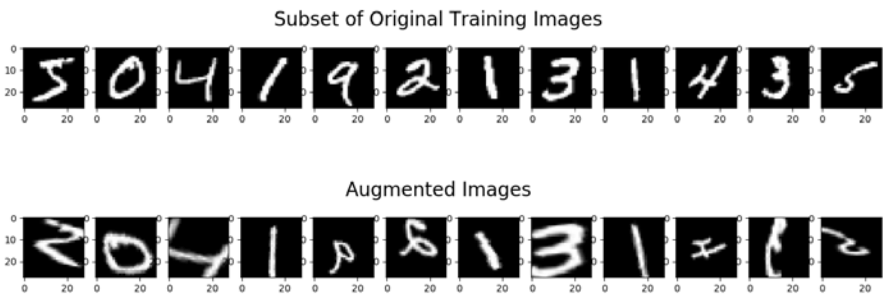
zoom_range=0.5 #随机缩放到 [1-50％，1+50%]

代码 mnist_train_ex2.py：

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)  # 给数据增加一个维度，使数据和网络结构匹配
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

image_gen_train = ImageDataGenerator(
                                     rescale=1./1.,#如为图像，分母为255时，可归至0～1
                                     rotation_range=45,#随机45度旋转
                                     width_shift_range=.15,#宽度偏移
                                     height_shift_range=.15,#高度偏移
                                     horizontal_flip=True,#水平翻转
                                     zoom_range=0.5#将图像随机缩放阈量50%
                                     )
image_gen_train.fit(x_train)

model = tf.keras.models.Sequential([
                                    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
                                    tf.keras.layers.Dense(128, activation='relu'),
                                    tf.keras.layers.Dense(10, activation='softmax')
                                    ])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

model.fit(image_gen_train.flow(x_train, y_train, batch_size=32), epochs=5, validation_data=(x_test,y_test),validation_freq=1)
model.summary()
```

注：1、model.fit(x_train,y_train,batch_size=32,……)变为 model.fit(image_gen_train.flow(x_train, y_train,batch_size=32), ……)；2、数据增强函数的输入要求是 4 维，通过 reshape 调整；3、如果报错：缺少 scipy 库，pip install scipy 即可。

2.2、数据增强可视化（代码 show_augmented _images.py）



3、断点续训，存取模型

3.1、读取模型

load_weights(路径文件名)

```
checkpoint_save_path = "./checkpoint/mnist.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    print('-------------load the model-----------------')
    model.load_weights(checkpoint_save_path)
```

3.2、保存模型

借助 tensorflow 给出的回调函数，直接保存参数和网络

```
tf.keras.callbacks.ModelCheckpoint(
            filepath=路径文件名,
            save_weights_only=True,
            monitor='val_loss',   # val_loss or loss
            save_best_only=True)
history = model.fit(x_train, y_train, batch_size=32, epochs=5,
validation_data=(x_test, y_test), validation_freq=1,
callbacks=[cp_callback])
```

注：monitor 配合 save_best_only 可以保存最优模型，包括：训练损失最小模型、测试损失最小模型、训练准确率最高模型、测试准确率最高模型等。

代码 mnist_train_ex3.py：

```python
import tensorflow as tf
import os

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])


checkpoint_save_path = "./checkpoint/mnist.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    print('-------------load the model-----------------')
    model.load_weights(checkpoint_save_path)

cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_save_path,
                                                 save_weights_only=True,
                                                 monitor='val_loss',  # val_loss or loss
                                                 save_best_only=True)

history = model.fit(x_train, y_train, batch_size=32, epochs=5, validation_data=(x_test, y_test), validation_freq=1,
                    callbacks=[cp_callback])
model.summary()
```

4、参数提取，写至文本

4.1、提取可训练参数

model.trainable_variables 模型中可训练的参数

4.2、设置 print 输出格式

np.set_printoptions(precision=小数点后按四舍五入保留几位,threshold=数组元素数量少于或等于门槛值，打印全部元素；否则打印门槛值+1 个元素，中间用省略号补充)

```
>>> np.set_printoptions(precision=5)
>>> print(np.array([1.123456789]))
[1.12346]
>>> np.set_printoptions(threshold=5)
>>> print(np.arange(10))
 [0 1 2 … , 7 8 9]
```

注：precision=np.inf 打印完整小数位；threshold=np.nan 打印全部数组元素。

代码 mnist_train_ex4.py：

设置显示全部内容

np.inf 表示无穷大

打印模型参数

存入文本

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

np.set_printoptions(threshold=np.inf)

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

checkpoint_save_path = "./checkpoint/mnist.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    print('-------------load the model-----------------')
    model.load_weights(checkpoint_save_path)

cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_save_path,
                                                 save_weights_only=True,
                                                 monitor='val_loss',  # val_loss or loss
                                                 save_best_only=True)

history = model.fit(x_train, y_train, batch_size=32, epochs=5, validation_data=(x_test, y_test), validation_freq=1,
                    callbacks=[cp_callback])
model.summary()

print(model.trainable_variables)
file = open('./weights.txt', 'w')
for v in model.trainable_variables:
    file.write(str(v.name) + '\n')
    file.write(str(v.shape) + '\n')
    file.write(str(v.numpy()) + '\n')
file.close()
```

模型参数打印结果:

```
         1.97993845e-01, -2.78407305e-01, -8.17760266e-03,
        -7.00537682e-01, -4.84343439e-01,  3.32634568e-01,
         3.56431931e-01],
       [-4.86615181e-01,  1.56931654e-01,  3.42359006e-01,
         4.70145404e-01, -7.82909155e-01, -7.54599690e-01,
        -3.46825391e-01,  4.61666703e-01,  1.52707055e-01,
        -1.02171159e+00],
       [ 2.49814410e-02, -3.94866347e-01, -2.38858745e-01,
        -4.76421565e-01, -8.29285800e-01,  4.69416261e-01,
        -2.57994473e-01,  1.95362136e-01,  2.25063980e-01,
         1.16042025e-01],
       [-1.83508769e-01,  3.88530493e-02, -4.87567544e-01,
        -2.24406436e-01, -3.49434733e-01,  4.11300808e-01,
        -1.88447893e-01, -2.36360729e-02, -5.55139363e-01,
         1.78482935e-01]], dtype=float32)>, <tf.Variable 'dense_1/bias:0' shape=(10,) dtype=float32, numpy=
array([-0.11309464, -0.21033764, -0.02997856, -0.0904652 ,  0.12025899,
       -0.00598264, -0.09277644, -0.22755557,  0.3746695 ,  0.07525402],
      dtype=float32)>]
```

weights.txt:

```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
dense/kernel:0
(784, 128)
[[-5.28208241e-02  1.77696347e-02  5.36200777e-02 -8.02464187e-02
  -7.40158111e-02  2.43436918e-02  7.37475380e-02 -4.93402593e-02
  -5.28204963e-02 -1.22503266e-02  3.49320099e-02 -6.66563958e-03
   4.27991748e-02  4.25620303e-02  3.26388329e-02 -6.82831556e-03
  -3.72390486e-02  6.31327555e-02  1.49547681e-02 -4.88554873e-02
  -7.91044012e-02  2.88710743e-02  7.29896501e-02 -1.50888264e-02
  -4.08900231e-02 -4.23861295e-02  1.30817220e-02 -3.70090567e-02
   7.55885169e-02  5.81764653e-02 -2.59572975e-02 -4.89420667e-02
  -8.05656984e-02 -2.90985331e-02 -6.55247271e-03 -6.37586638e-02
   1.10701323e-02 -7.90546238e-02 -4.18584831e-02  7.35382959e-02
   8.01183209e-02  8.07191208e-02 -4.53560278e-02 -7.35454261e-04
  -4.34553251e-02 -5.70111275e-02 -3.52828428e-02  1.22970864e-02
  -7.03834444e-02 -2.02142075e-02  4.55207303e-02 -1.19082928e-02
   3.77118587e-04 -1.51321813e-02  7.47007057e-02  5.36825508e-03
  -6.94774091e-02  1.67139769e-02  7.05072358e-02  2.72734016e-02
   2.63411999e-02  7.09173158e-02 -7.11789280e-02 -7.05774128e-03
  -1.76456347e-02  3.48469615e-02 -2.76021659e-03  6.56074658e-02
  -7.71965832e-03  7.47766420e-02 -4.43984345e-02  1.25611573e-03
  -5.33864722e-02 -4.23931293e-02 -7.32599348e-02 -3.05883363e-02
  -7.33513385e-03  6.56326190e-02  4.32138294e-02 -6.28611743e-02
  -2.38207243e-02 -1.37094408e-02  2.65896618e-02  7.17325583e-02
```

5、acc/loss 可视化，查看效果

5.1、acc 曲线与 loss 曲线

history=model.fit(训练集数据， 训练集标签， batch_size=， epochs=，

validation_split=用作测试数据的比例,validation_data=测试集,

validation_freq=测试频率)

history:

loss：训练集 loss

val_loss：测试集 loss

sparse_categorical_accuracy：训练集准确率

val_sparse_categorical_accuracy：测试集准确率

```
acc = history.history['sparse_categorical_accuracy']
val_acc = history.history['val_sparse_categorical_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
#########     show    #########

# 显示训练集和验证集的acc和loss曲线
```

history 中读取所需数据

```
acc = history.history['sparse_categorical_accuracy']
val_acc = history.history['val_sparse_categorical_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

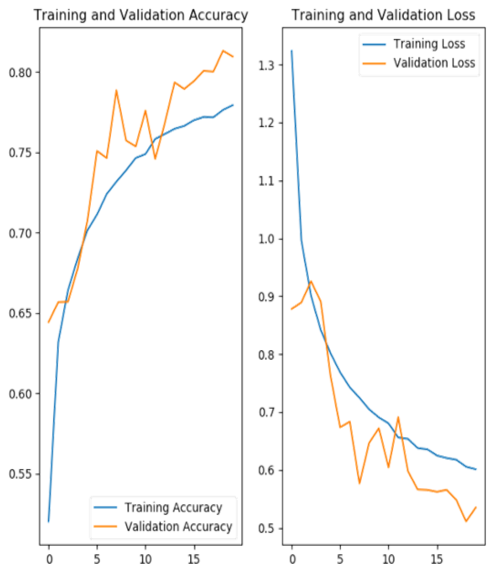
画图

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
```

画图

```
plt.subplot(1, 2, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

acc 和 loss 曲线图:



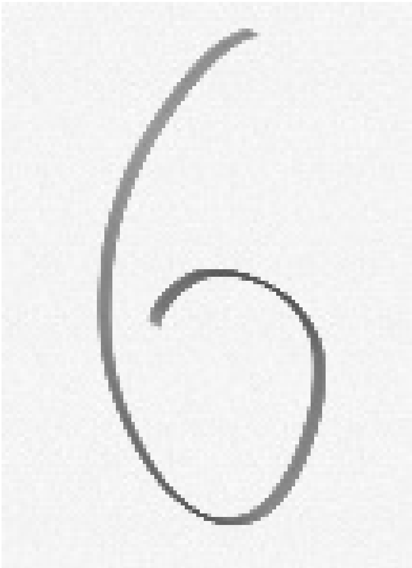
代码 mnist_train_ex5.py：

```python
import tensorflow as tf
import os
import numpy as np
from matplotlib import pyplot as plt

np.set_printoptions(threshold=np.inf)

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

checkpoint_save_path = "./checkpoint/mnist.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    print('-------------load the model-----------------')
    model.load_weights(checkpoint_save_path)

cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_save_path,
                                                 save_weights_only=True,
                                                 monitor='val_loss',  # val_loss or loss
                                                 save_best_only=True)

history = model.fit(x_train, y_train, batch_size=32, epochs=5, validation_data=(x_test, y_test), validation_freq=1,
                    callbacks=[cp_callback])
model.summary()

print(model.trainable_variables)
file = open('./weights.txt', 'w')
for v in model.trainable_variables:
    file.write(str(v.name) + '\n')
    file.write(str(v.shape) + '\n')
    file.write(str(v.numpy()) + '\n')
file.close()

#########    show    #########

# 显示训练集和验证集的acc和loss曲线
acc = history.history['sparse_categorical_accuracy']
val_acc = history.history['val_sparse_categorical_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

6、应用程序，给图识物

6.1、给图识物

输入一张手写数字图片：



神经网络自动识别出值：

6

手写十个数，正确率 90%以上合格。

6.2、前向传播执行应用

predict(输入数据, batch_size=整数) 返回前向传播计算结果

注：predict 参数详解。(1)x：输入数据，Numpy 数组（或者 Numpy 数组的列表，如果模型有多个输出)；(2)batch_size：整数，由于 GPU 的特性，batch_size 最好选用 8，16，32，64……，如果未指定，默认为 32；(3)verbose: 日志显示模式,0 或 1;(4)steps: 声明预测结束之前的总步数(批次样本),默认值 None；(5)返回：预测的 Numpy 数组（或数组列表）。

代码 mnist_app1.py：

加载模型

预测图片数量

预测图片路径

打开图片

调整尺寸和类型

二值化

预测

输出结果

```python
from PIL import Image
import numpy as np
import tensorflow as tf
import os

model_save_path = './checkpoint/mnist.ckpt'
model = tf.keras.models.Sequential([
                                    tf.keras.layers.Flatten(input_shape=(28, 28)),
                                    tf.keras.layers.Dense(128, activation='relu'),
                                    tf.keras.layers.Dense(10, activation='softmax')
                                    ])

model.load_weights(model_save_path)

preNum = int(input("input the number of test pictures:"))
for i in range(preNum):
    image_path = input("the path of test picture:")

    img = Image.open(image_path)

    img=img.resize((28,28),Image.ANTIALIAS)
    img_arr = np.array(img.convert('L'))

    for i in range(28):
        for j in range(28):
            if img_arr[i][j]<200:
                img_arr[i][j]=255
            else:
                img_arr[i][j]=0

    img_arr=img_arr/255.0

    x_predict = img_arr[tf.newaxis,...]

    result = model.predict(x_predict)
    pred=tf.argmax(result, axis=1)
    print('\n')
    tf.print(pred)
```

注:1、输出结果 pred 是张量,需要用 tf.print,print 打印出来是 tf.Tensor([1], shape=(1,), dtype=int64); 2、去掉二值化, 出现无法收敛问题, 需要对数据集进行归一化。

输出结果:

```
input the number of test pictures:10
the path of test picture:./pic/1.png


[1]
the path of test picture:./pic/2.png

[2]
```

代码 mnist_app2.py：

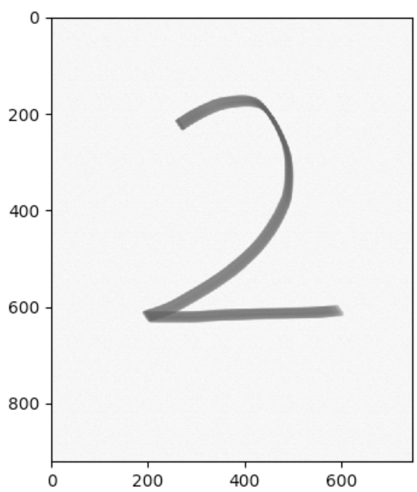
```
for i in range(preNum):
    image_path = input("the path of test picture:")
    img = Image.open(image_path)

    image = plt.imread(image_path)
    plt.set_cmap('gray')
    plt.imshow(image)

    img=img.resize((28,28),Image.ANTIALIAS)
    img_arr = np.array(img.convert('L'))


    for i in range(28):
        for j in range(28):
            if img_arr[i][j]<200:
                img_arr[i][j]=255
            else:
                img_arr[i][j]=0

    img_arr=img_arr/255.0
    x_predict = img_arr[tf.newaxis,...]
    result = model.predict(x_predict)
    pred=tf.argmax(result, axis=1)

    print('\n')
    tf.print(pred)

    plt.pause(1)
    plt.close()
```

显示图片

暂停 1 秒后

关闭图片

可视化图片：



三、补充

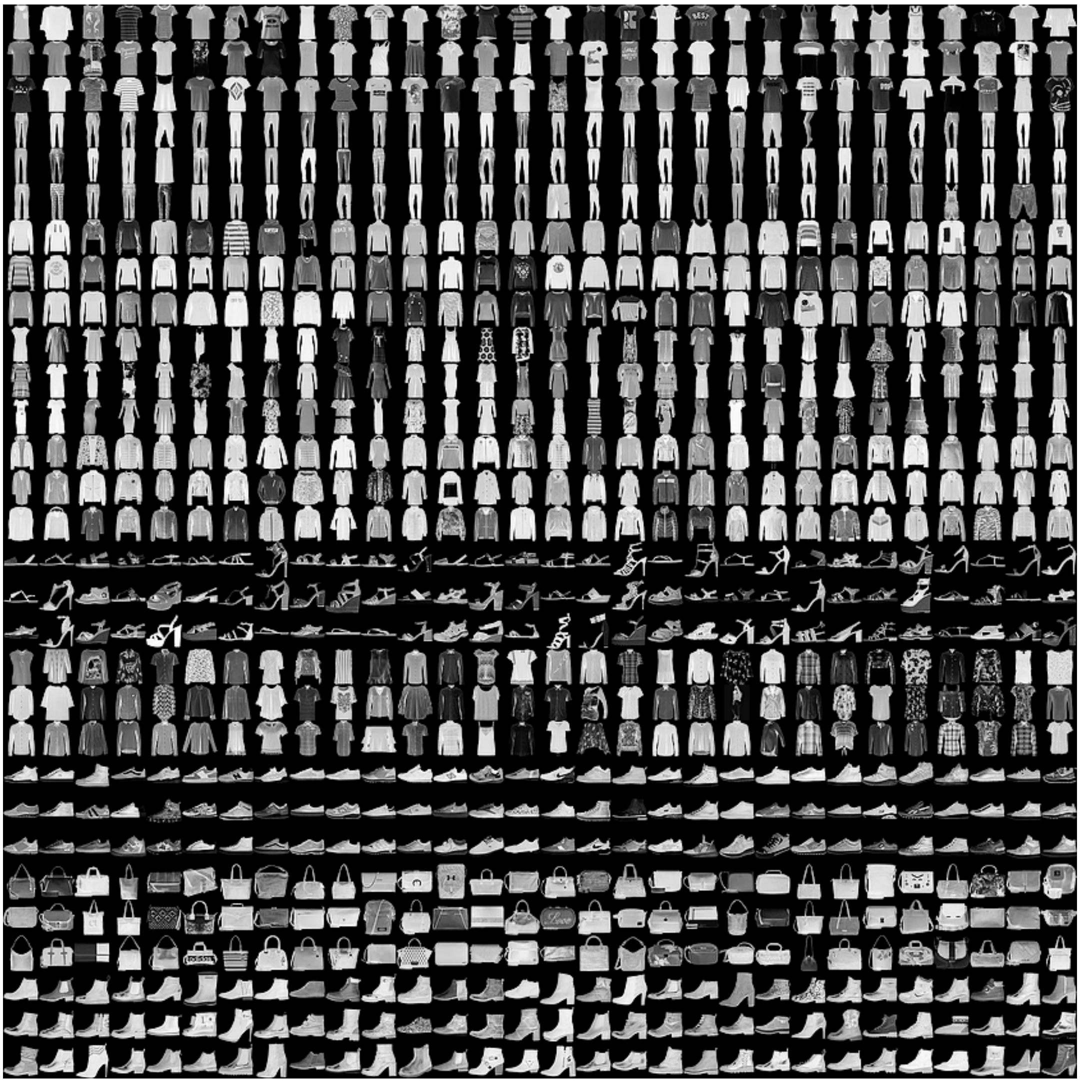
1、数据集 Fashion_mnist

Label	Description
0	T恤（T-shirt/top）
1	裤子（Trouser）
2	套头衫（Pullover）
3	连衣裙（Dress）
4	外套（Coat）
5	凉鞋（Sandal）
6	衬衫（Shirt）
7	运动鞋（Sneaker）
8	包（Bag）
9	靴子（Ankle boot）

60000 张训练图像和对应标签；

10000 张测试图像和对应标签；

每张图像 28x28 的分辨率。



注：load fashion_mnist 数据集 x_train.shape: (60000, 28, 28)无通道数；送入卷积前先 reshape 成 x_train.shape: (60000, 28, 28, 1)