人工智能实践：Tensorflow笔记

曹 健

北京大学

软件与微电子学院

tf.keras搭建神经网络八股

六步法

import

自制数据集　　**train, test**　　　　数据增强

Sequential / Class

model.compile

断点续训　**model.fit**

model.summary

参数提取　　　　　　　**acc/loss可视化**

前向推理实现应用

本讲目标：神经网络八股功能扩展

① 自制数据集，解决本领域应用

② 数据增强，扩充数据集

③ 断点续训，存取模型

④ 参数提取，把参数存入文本

⑤ acc/loss可视化，查看训练效果

⑥ 应用程序，给图识物

源码：class3\p14_mnist_sequential.py ＝ class4\ MNIST_FC\p4_mnist_train_baseline.py

import

train test

models.Sequential

model.compile

model.fit
model.summary

```python
1  import tensorflow as tf
2
3  mnist = tf.keras.datasets.mnist
4  (x_train, y_train), (x_test, y_test) = mnist.load_data()
5  x_train, x_test = x_train / 255.0, x_test / 255.0
6
7  model = tf.keras.models.Sequential([
8      tf.keras.layers.Flatten(),
9      tf.keras.layers.Dense(128, activation='relu'),
10     tf.keras.layers.Dense(10, activation='softmax')
11 ])
12
13 model.compile(optimizer='adam',
14               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
15               metrics=['sparse_categorical_accuracy'])
16
17 model.fit(x_train, y_train, batch_size=32, epochs=5, validation_data=(x_test, y_test), validation_freq=1)
18 model.summary()
```

import

train, test

Sequential

model.compile

model.fit

model.summary

4

本讲目标：神经网络八股功能扩展

① 自制数据集，解决本领域应用

② 数据增强，扩充数据集

③ 断点续训，存取模型

④ 参数提取，把参数存入文本

⑤ acc/loss可视化，查看训练效果

⑥ 应用程序，给图识物

观察数据集数据结构，给x_train、y_train、x_test、y_test 赋值

mnist_image_label文件夹：

AI > class4 > MNIST_FC > mnist_image_label

mnist_train_jpg_60000

图片

源码：class3\p13_mnist_datasets.py

mnist_test_jpg_10000

0_5.jpg
图片文件(.jpg)
851 字节

1_0.jpg
图片文件(.jpg)
799 字节

2_4.jpg
图片文件(.jpg)
772 字节

3_1.jpg
图片文件(.jpg)
633 字节

4_9.jpg
图片文件(.jpg)
813 字节

5_2.jpg
图片文件(.jpg)
784 字节

x_train.shape:

(60000, 28, 28)

y_train.shape:

(60000,)

x_test.shape:

(10000, 28, 28)

y_test.shape:

(10000,)

mnist_train_jpg_60000.txt

标签

mnist_test_jpg_10000.txt

```
0_5.jpg    5
1_0.jpg    0
2_4.jpg    4
3_1.jpg    1
4_9.jpg    9
```

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

```
def generateds(图片路径，标签文件)：
```

def generateds(图片路径，标签文件):

value[0]	value[1]
0_5.jpg	5
1_0.jpg	0
2_4.jpg	4
3_1.jpg	1
4_9.jpg	9

```python
def generateds(path, txt):
    f = open(txt, 'r')
    contents = f.readlines()
    f.close()
    x, y_ = [], []
    for content in contents:
        value = content.split()
        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_
```

import

```python
1  import tensorflow as tf
2  from PIL import Image
3  import numpy as np
4  import os
5
6  train_path = './mnist_image_label/mnist_train_jpg_60000/'
7  train_txt = './mnist_image_label/mnist_train_jpg_60000.txt'
8  x_train_savepath = './mnist_image_label/mnist_x_train.npy'
9  y_train_savepath = './mnist_image_label/mnist_y_train.npy'
10
11  test_path = './mnist_image_label/mnist_test_jpg_10000/'
12  test_txt = './mnist_image_label/mnist_test_jpg_10000.txt'
13  x_test_savepath = './mnist_image_label/mnist_x_test.npy'
14  y_test_savepath = './mnist_image_label/mnist_y_test.npy'
15
16  def generateds(path, txt):
17      f = open(txt, 'r')        #以只读形式打开txt文件
18      contents = f.readlines()    # 读取文件中所有行
19      f.close()         #关闭txt文件
20      x, y_ = [], []    #建立空列表
21      for content in contents:   #逐行取出
22          value = content.split()   # 以空格分开，图片路径为value[0] ，标签文件为value[1] ，存入列表
23          img_path = path + value[0]    #拼出图片路径和文件名
24          img = Image.open(img_path)     #读入图片
25          img = np.array(img.convert('L'))     #图片变为8位宽灰度值的np.array格式
26          img = img / 255.     #数据归一化 （实现预处理）
27          x.append(img)    #归一化后的数据，贴到列表x
28          y_.append(value[1])   #标签贴到列表y_
29          print('loading : ' + content)    #打印状态提示
30
31      x = np.array(x)   #变为np.array格式
32      y_ = np.array(y_)   #变为np.array格式
33      y_ = y_.astype(np.int64)   #变为64位整型
34      return x, y_   #返回输入特征x，返回标签y_
```

```python
if os.path.exists(x_train_savepath) and os.path.exists(y_train_savepath) and os.path.exists(
        x_test_savepath) and os.path.exists(y_test_savepath):
    print('-------------Load Datasets-----------------')
    x_train_save = np.load(x_train_savepath)
    y_train = np.load(y_train_savepath)
    x_test_save = np.load(x_test_savepath)
    y_test = np.load(y_test_savepath)
    x_train = np.reshape(x_train_save, (len(x_train_save), 28, 28))
    x_test = np.reshape(x_test_save, (len(x_test_save), 28, 28))
else:
    print('-------------Generate Datasets-----------------')
    x_train, y_train = generateds(train_path, train_txt)
    x_test, y_test = generateds(test_path, test_txt)

    print('-------------Save Datasets-----------------')
    x_train_save = np.reshape(x_train, (len(x_train), -1))
    x_test_save = np.reshape(x_test, (len(x_test), -1))
    np.save(x_train_savepath, x_train_save)
    np.save(y_train_savepath, y_train)
    np.save(x_test_savepath, x_test_save)
    np.save(y_test_savepath, y_test)

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

model.compile(optimizer='adam',
                loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                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()
```

train test

models.Sequential

model.compile

model.fit
model.summary

本讲目标：神经网络八股功能扩展

①自制数据集，解决本领域应用

②数据增强，扩充数据集

③断点续训，存取模型

④参数提取，把参数存入文本

⑤acc/loss可视化，查看训练效果

⑥应用程序，给图识物

数据增强（增大数据量）

image_gen_train = tf.keras.preprocessing.image.ImageDataGenerator(

      rescale = 所有数据将乘以该数值

      rotation_range = 随机旋转角度数范围

      width_shift_range = 随机宽度偏移量

      height_shift_range = 随机高度偏移量

      水平翻转：**horizontal_flip =** 是否随机水平翻转

      随机缩放：**zoom_range =** 随机缩放的范围 **[1-n，1+n])**

image_gen_train.fit(x_train)

```
例：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=False,   # 水平翻转
      zoom_range=0.5   # 将图像随机缩放阈量50%)

   image_gen_train.fit(x_train)
```

源码：p11_show_augmented _images.py

数据增强（增大数据量）

image_gen_train = tf.keras.preprocessing.image.ImageDataGenerator(

       rescale = 所有数据将乘以该数值

       rotation_range = 随机旋转角度数范围

       width_shift_range = 随机宽度偏移量

       height_shift_range = 随机高度偏移量

       水平翻转：**horizontal_flip =** 是否随机水平翻转

       随机缩放：**zoom_range =** 随机缩放的范围 **[1-n，1+n])**

image_gen_train.fit(x_train)

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
```

 (60000, 28, 28) ⟹ (60000, 28, 28, 1)

model.fit(x_train, y_train,batch_size=32, ……)

⇓

model.fit(image_gen_train.flow(x_train, y_train,batch_size=32), ……)

```
 1  import tensorflow as tf
 2  from tensorflow.keras.preprocessing.image import ImageDataGenerator
 3
 4  fashion = tf.keras.datasets.fashion_mnist
 5  (x_train, y_train), (x_test, y_test) = fashion.load_data()
 6  x_train, x_test = x_train / 255.0, x_test / 255.0
 7  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
 8
 9  image_gen_train = ImageDataGenerator(
10          rescale=1. / 1.,    # 如为图像, 分母为255时, 可归至0~1
11          rotation_range=45,   # 随机45度旋转
12          width_shift_range=.15,   # 宽度偏移
13          height_shift_range=.15,   # 高度偏移
14          horizontal_flip=True,    # 水平翻转
15          zoom_range=0.5   # 将图像随机缩放阈量50%
16  )
17  image_gen_train.fit(x_train)
18
19  model = tf.keras.models.Sequential([
20          tf.keras.layers.Flatten(),
21          tf.keras.layers.Dense(128, activation='relu'),
22          tf.keras.layers.Dense(10, activation='softmax')
23  ])
24
25  model.compile(optimizer='adam',
26                loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
27                metrics=['sparse_categorical_accuracy'])
28
29  model.fit(image_gen_train.flow(x_train, y_train, batch_size=32), epochs=5, validation_data=(x_test, y_test),
30          validation_freq=1)
31  model.summary()
```

import

train test

models.Sequential

model.compile

model.fit

model.summary

本讲目标：神经网络八股功能扩展

①自制数据集，解决本领域应用

②数据增强，扩充数据集

③断点续训，存取模型

④参数提取，把参数存入文本

⑤acc/loss可视化，查看训练效果

⑥应用程序，给图识物

读取保存模型

读取模型：

load_weights(路径文件名）

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

保存模型：

tf.keras.callbacks.ModelCheckpoint(

filepath=路径文件名,

save_weights_only=True/False,

save_best_only=True/False)

history = model.fit（ callbacks=[cp_callback] ）

```python
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_save_path,
                                                 save_weights_only=True,
                                                 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])
```

```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(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              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,
                                                 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()
```

import

train test

models.Sequential

model.compile

model.fit

model.summary

本讲目标：神经网络八股功能扩展

①自制数据集，解决本领域应用

②数据增强，扩充数据集

③断点续训，存取模型

④参数提取，把参数存入文本

⑤acc/loss可视化，查看训练效果

⑥应用程序，给图识物

提取可训练参数

model.trainable_variables 返回模型中可训练的参数

设置print输出格式

np.set_printoptions(threshold=超过多少省略显示)

```
np.set_printoptions(threshold=np.inf)   # np.inf表示无限大
```

```python
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()
```

import

train test

models.Sequential

model.compile

model.fit

model.summary

```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(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              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,
                                                 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()
```

本讲目标：神经网络八股功能扩展

①自制数据集，解决本领域应用

②数据增强，扩充数据集

③断点续训，存取模型

④参数提取，把参数存入文本

⑤acc/loss可视化，查看训练效果

⑥应用程序，给图识物

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：

训练集loss： loss

测试集loss： val_loss

训练集准确率： sparse_categorical_accuracy

测试集准确率： val_sparse_categorical_accuracy

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

```python
# 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.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()
```

```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(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              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,
                                                 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()
```

import

train test

models.Sequential

model.compile

model.fit

model.summary

```python
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.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()
```
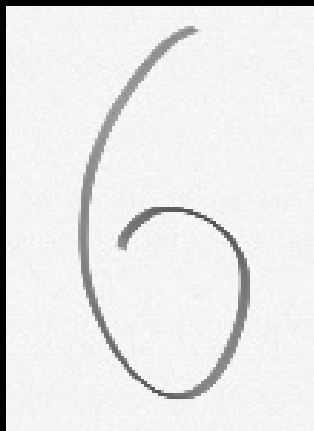
本讲目标：神经网络八股功能扩展

①自制数据集，解决本领域应用

②数据增强，扩充数据集

③断点续训，存取模型

④参数提取，把参数存入文本

⑤acc/loss可视化，查看训练效果

⑥应用程序，给图识物

给图识物

输入一张手写数字图片　　　　　　　　　　　输出识别结果

神经网络自动识别出值

前向传播执行应用

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

复现模型
（前向传播）

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

加载参数

```
model.load_weights(model_save_path)
```

预测结果

```
result = model.predict(x_predict)
```

```python
1   from PIL import Image
2   import numpy as np
3   import tensorflow as tf
4
5   model_save_path = './checkpoint/mnist.ckpt'
6
7   model = tf.keras.models.Sequential([
8       tf.keras.layers.Flatten(),
9       tf.keras.layers.Dense(128, activation='relu'),
10      tf.keras.layers.Dense(10, activation='softmax')])
11
12  model.load_weights(model_save_path)
13
14  preNum = int(input("input the number of test pictures:"))
15
```
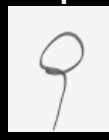
```python
16  for i in range(preNum):
17      image_path = input("the path of test picture:")
18      img = Image.open(image_path)
19      img = img.resize((28, 28), Image.ANTIALIAS)
20      img_arr = np.array(img.convert('L'))
21
22
23
24      img_arr = 255 - img_arr
25
26
27
28
29      img_arr = img_arr / 255.0
30      x_predict = img_arr[tf.newaxis, ...]
31      result = model.predict(x_predict)
32      pred = tf.argmax(result, axis=1)
33      print('\n')
34      tf.print(pred)
```

0.png 1.png 2.png 3.png 4.png

5.png 6.png 7.png 8.png 9.png

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

model_save_path = './checkpoint/mnist.ckpt'

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.
```

```
img_arr: (28, 28)
x_predict: (1, 28, 28)
```

```python
model.load_we
```

```python
preNum = int(
```

```python
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)
```

0.png 1.png 2.png 3.png 4.png

5.png 6.png 7.png 8.png 9.png