《用keras实现服装识别》

32018070103

卢文欣

1. 引言

Keras是一个由Python编写的开源人工神经网络库，可以作为Tensorflow、Microsoft-CNTK和Theano的高阶应用程序接口，进行深度学习模型的设计、调试、评估、应用和可视化 。其中可以自己建模型，十分方便好用。所以我这次打算用keras实现服装识别。

1. 材料与方法

import tensorflow as tf

from tensorflow import keras

import matplotlib.pyplot as plt

import pandas as pd

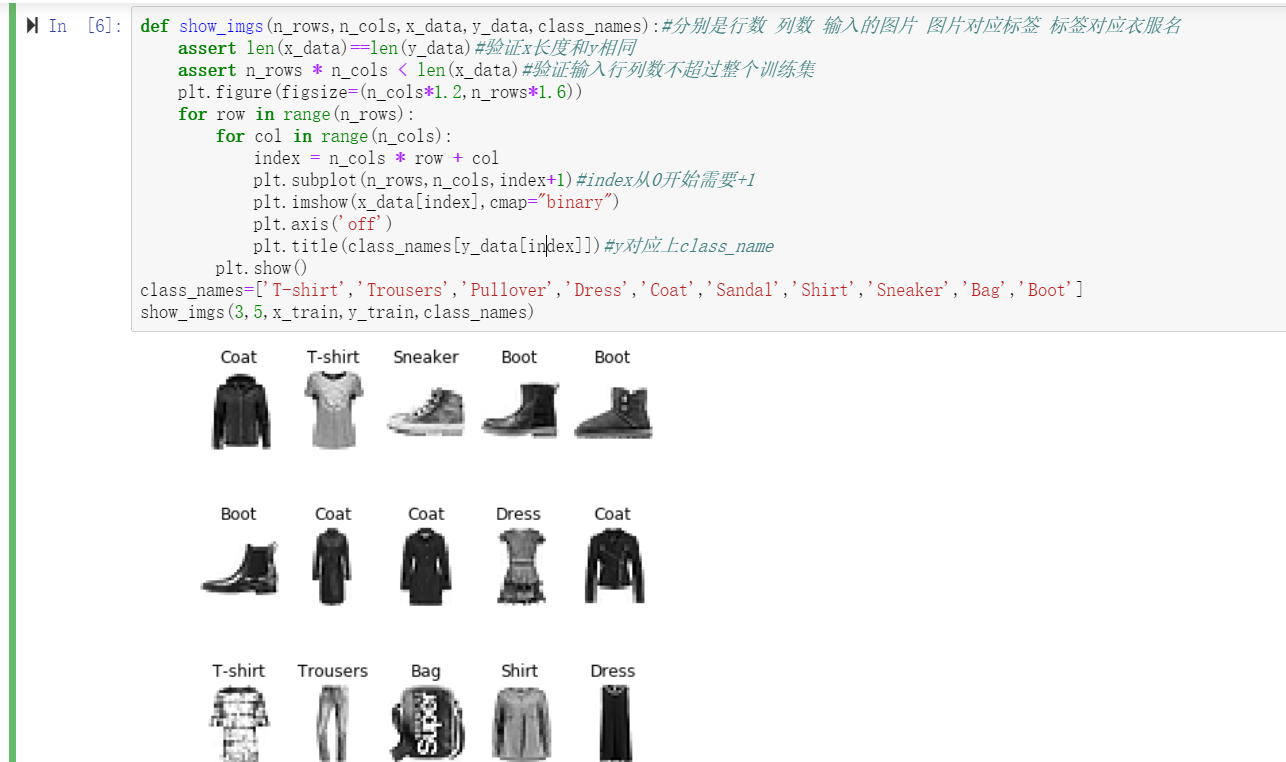
import numpy as np

**1.导入数据并划分测试集训练集验证集**

****

**2.数据查看**

****

****

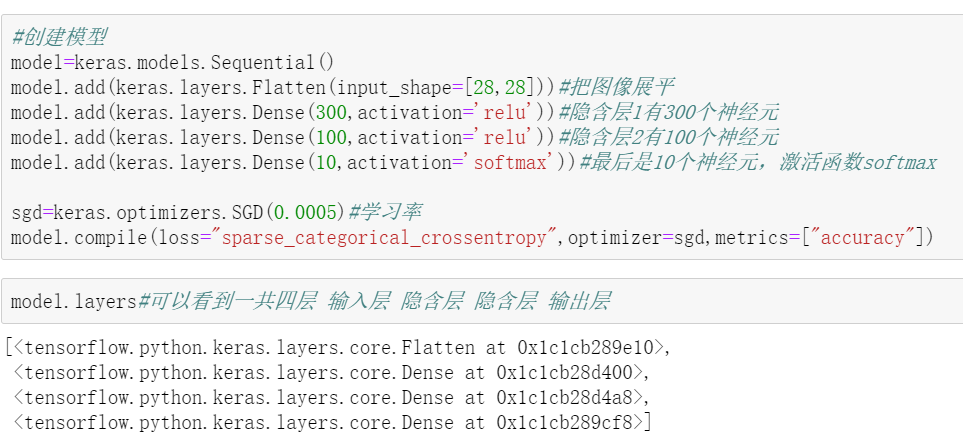
1. 实验过程

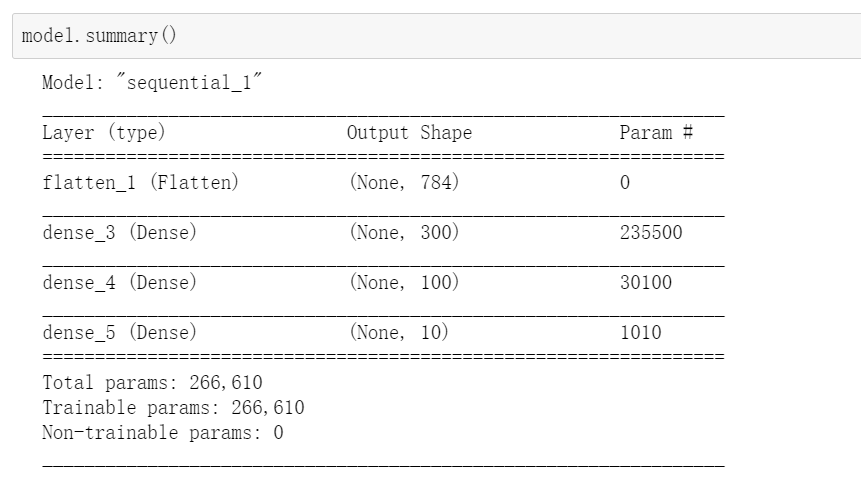
**1.数据处理**



**2.模型1**

**四层神经元**





50次之后的训练集和验证集到达89%和88%左右

Epoch 45/50

1719/1719 [==============================] - 11s 6ms/step - loss: 0.2954 - accuracy: 0.8954 - val\_loss: 0.3283 - val\_accuracy: 0.8844

Epoch 46/50

1719/1719 [==============================] - 10s 6ms/step - loss: 0.2942 - accuracy: 0.8955 - val\_loss: 0.3281 - val\_accuracy: 0.8838

Epoch 47/50

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2930 - accuracy: 0.8963 - val\_loss: 0.3268 - val\_accuracy: 0.8840

Epoch 48/50

1719/1719 [==============================] - 11s 6ms/step - loss: 0.2915 - accuracy: 0.8965 - val\_loss: 0.3271 - val\_accuracy: 0.8852

Epoch 49/50

1719/1719 [==============================] - 11s 6ms/step - loss: 0.2904 - accuracy: 0.8965 - val\_loss: 0.3264 - val\_accuracy: 0.8830

Epoch 50/50

1719/1719 [==============================] - 11s 6ms/step - loss: 0.2894 - accuracy: 0.8971 - val\_loss: 0.3263 - val\_accuracy: 0.8844

测试集86%左右

313/313 [==============================] - 1s 4ms/step - loss: 0.3659 - accuracy: 0.8697

**3.模型2**

多加20层神经元



Epoch 1/10

1719/1719 [==============================] - 18s 11ms/step - loss: 2.3018 - accuracy: 0.1436 - val\_loss: 2.3011 - val\_accuracy: 0.1764

Epoch 2/10

1719/1719 [==============================] - 18s 11ms/step - loss: 2.3002 - accuracy: 0.2369 - val\_loss: 2.2990 - val\_accuracy: 0.2618

Epoch 3/10

1719/1719 [==============================] - 20s 11ms/step - loss: 2.2969 - accuracy: 0.2839 - val\_loss: 2.2940 - val\_accuracy: 0.3312

Epoch 4/10

1719/1719 [==============================] - 20s 12ms/step - loss: 2.2890 - accuracy: 0.2994 - val\_loss: 2.2821 - val\_accuracy: 0.2750

Epoch 5/10

1719/1719 [==============================] - 20s 11ms/step - loss: 2.2700 - accuracy: 0.2684 - val\_loss: 2.2525 - val\_accuracy: 0.2598

Epoch 6/10

1719/1719 [==============================] - 20s 11ms/step - loss: 2.2154 - accuracy: 0.2421 - val\_loss: 2.1473 - val\_accuracy: 0.2310

Epoch 7/10

1719/1719 [==============================] - 19s 11ms/step - loss: 1.8359 - accuracy: 0.2641 - val\_loss: 1.4394 - val\_accuracy: 0.3514

Epoch 8/10

1719/1719 [==============================] - 16s 10ms/step - loss: 1.2390 - accuracy: 0.4613 - val\_loss: 1.0695 - val\_accuracy: 0.5614

Epoch 9/10

1719/1719 [==============================] - 18s 10ms/step - loss: 1.0074 - accuracy: 0.5949 - val\_loss: 0.8676 - val\_accuracy: 0.6846

Epoch 10/10

1719/1719 [==============================] - 19s 11ms/step - loss: 0.8718 - accuracy: 0.6655 - val\_loss: 0.7752 - val\_accuracy: 0.7078

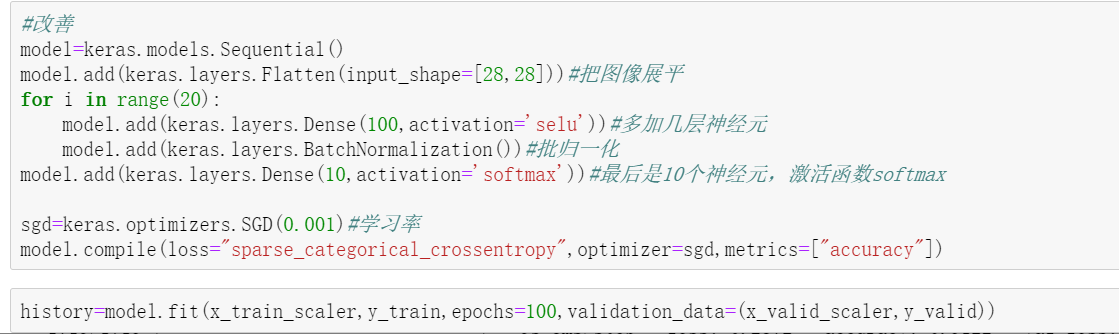
313/313 [==============================] - 2s 6ms/step - loss: 0.7939 - accuracy: 0.6946

效果一般

问题：缺少批归一化

1. **模型3**

增加BatchNormalization()并把激活函数relu改成selu（selu自带批归一化）



Epoch 95/100

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2294 - accuracy: 0.9150 - val\_loss: 0.3012 - val\_accuracy: 0.8932

Epoch 96/100

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2257 - accuracy: 0.9175 - val\_loss: 0.3052 - val\_accuracy: 0.8932

Epoch 97/100

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2257 - accuracy: 0.9171 - val\_loss: 0.2951 - val\_accuracy: 0.8950

Epoch 98/100

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2242 - accuracy: 0.9169 - val\_loss: 0.2954 - val\_accuracy: 0.8952

Epoch 99/100

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2246 - accuracy: 0.9179 - val\_loss: 0.2968 - val\_accuracy: 0.8994

Epoch 100/100

1719/1719 [==============================] - 9s 5ms/step - loss: 0.2233 - accuracy: 0.9168 - val\_loss: 0.2940 - val\_accuracy: 0.8942

测试集

313/313 [==============================] - 1s 2ms/step - loss: 0.3349 - accuracy: 0.8854

问题：37次以前一次耗时40s+，时间略长

1. **模型4**

Selu改回relu

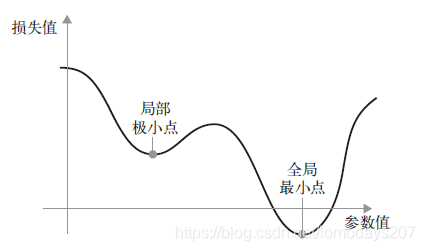
313/313 [==============================] - 1s 2ms/step - loss: 0.3469 - accuracy: 0.8777

时间变短，准确率差别不大

1. **模型5**

把optimizer=sgd换成optimizer='adam'

sgd的缺点：

  
如上图所示，在某个参数值附近，有一个局部极小点：在这个点附近，向左移动和向右移动都会导致损失值增大。如果使用小学习率的SGD 进行优化，那么优化过程可能会陷入局部极小点，导致无法找到全局最小点。

Adam的优势：每次迭代参数的学习率都有一定的范围，不会因为梯度很大而导致学习率（步长）也变得很大，参数的值相对比较稳定

Epoch 95/100

1719/1719 [==============================] - 11s 6ms/step - loss: 0.1339 - accuracy: 0.9536 - val\_loss: 0.3608 - val\_accuracy: 0.8958

Epoch 96/100

1719/1719 [==============================] - 11s 6ms/step - loss: 0.1338 - accuracy: 0.9531 - val\_loss: 0.3638 - val\_accuracy: 0.8986

Epoch 97/100

1719/1719 [==============================] - 11s 6ms/step - loss: 0.1341 - accuracy: 0.9532 - val\_loss: 0.3596 - val\_accuracy: 0.8946

Epoch 98/100

1719/1719 [==============================] - 11s 7ms/step - loss: 0.1356 - accuracy: 0.9539 - val\_loss: 0.3126 - val\_accuracy: 0.9004

Epoch 99/100

1719/1719 [==============================] - 13s 8ms/step - loss: 0.1294 - accuracy: 0.9546 - val\_loss: 0.3380 - val\_accuracy: 0.9010

Epoch 100/100

1719/1719 [==============================] - 11s 7ms/step - loss: 0.1319 - accuracy: 0.9547 - val\_loss: 0.3287 - val\_accuracy: 0.8986

313/313 [==============================] - 1s 2ms/step - loss: 0.3766 - accuracy: 0.8890

1. **模型6**

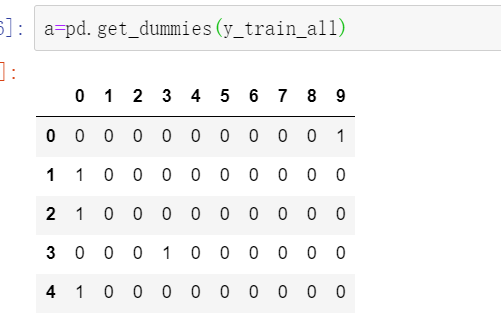
将y集改为独热编码

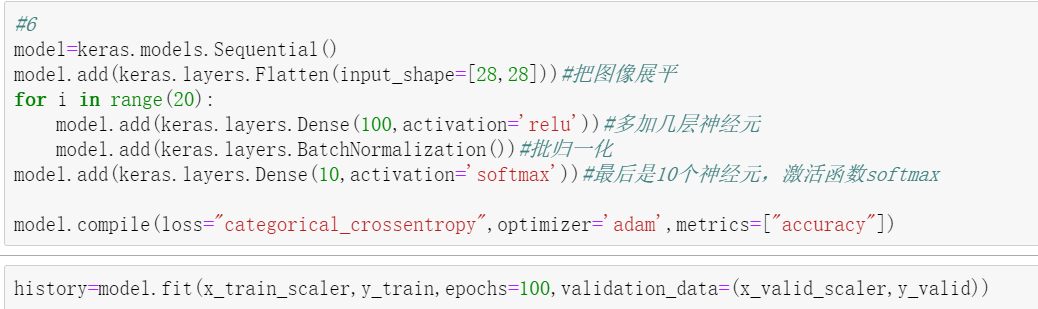
#独热编码

y\_valid=pd.get\_dummies(y\_valid)

y\_train=pd.get\_dummies(y\_train)

y\_test=pd.get\_dummies(y\_test)





Epoch 95/100

1719/1719 [==============================] - 43s 25ms/step - loss: 0.1485 - accuracy: 0.9475 - val\_loss: 0.3338 - val\_accuracy: 0.8914

Epoch 96/100

1719/1719 [==============================] - 44s 26ms/step - loss: 0.1459 - accuracy: 0.9482 - val\_loss: 0.3496 - val\_accuracy: 0.8914

Epoch 97/100

1719/1719 [==============================] - 40s 23ms/step - loss: 0.1473 - accuracy: 0.9481 - val\_loss: 0.3415 - val\_accuracy: 0.8918

Epoch 98/100

1719/1719 [==============================] - 41s 24ms/step - loss: 0.1407 - accuracy: 0.9498 - val\_loss: 0.3419 - val\_accuracy: 0.8910

Epoch 99/100

1719/1719 [==============================] - 40s 23ms/step - loss: 0.1408 - accuracy: 0.9506 - val\_loss: 0.3347 - val\_accuracy: 0.8890

Epoch 100/100

1719/1719 [==============================] - 37s 21ms/step - loss: 0.1404 - accuracy: 0.9504 - val\_loss: 0.3563 - val\_accuracy: 0.8954

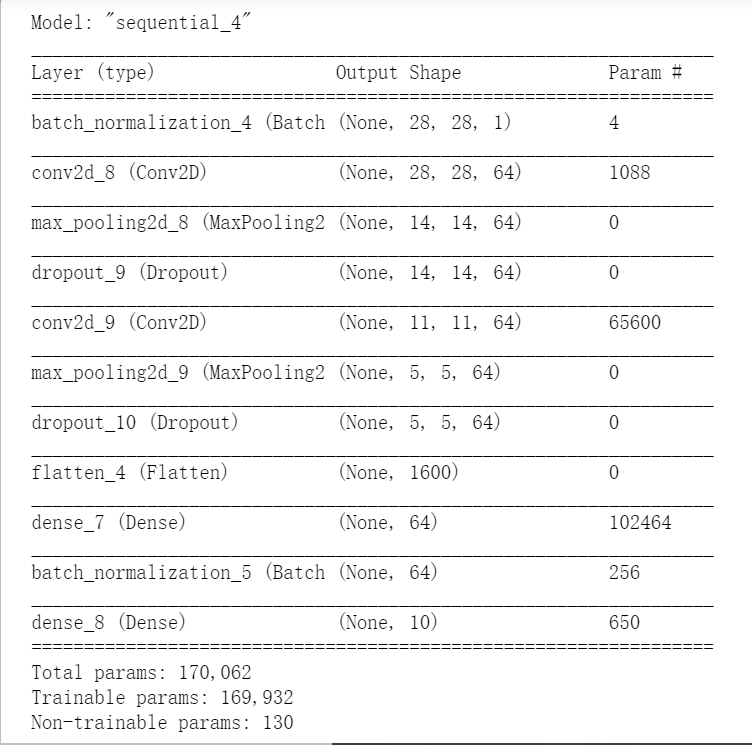
313/313 [==============================] - 2s 7ms/step - loss: 0.3895 - accuracy: 0.8872

变化不大

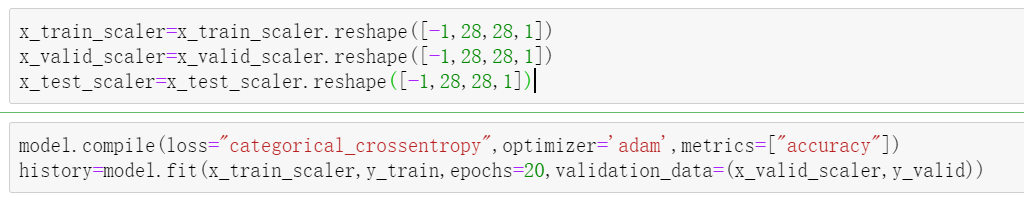
1. **模型7**

采用cnn卷积神经网络





需要对x集三维变四维



Epoch 1/20

1719/1719 [==============================] - 61s 36ms/step - loss: 0.3600 - accuracy: 0.8687 - val\_loss: 0.2903 - val\_accuracy: 0.8900

Epoch 2/20

1719/1719 [==============================] - 60s 35ms/step - loss: 0.2419 - accuracy: 0.9110 - val\_loss: 0.2473 - val\_accuracy: 0.9130

Epoch 3/20

1719/1719 [==============================] - 59s 35ms/step - loss: 0.1917 - accuracy: 0.9281 - val\_loss: 0.2326 - val\_accuracy: 0.9208

Epoch 4/20

1719/1719 [==============================] - 58s 34ms/step - loss: 0.1539 - accuracy: 0.9418 - val\_loss: 0.2611 - val\_accuracy: 0.9092

Epoch 5/20

1719/1719 [==============================] - 58s 34ms/step - loss: 0.1242 - accuracy: 0.9530 - val\_loss: 0.2806 - val\_accuracy: 0.9138

Epoch 6/20

1719/1719 [==============================] - 58s 34ms/step - loss: 0.1008 - accuracy: 0.9631 - val\_loss: 0.3188 - val\_accuracy: 0.9088

Epoch 7/20

1719/1719 [==============================] - 61s 35ms/step - loss: 0.0879 - accuracy: 0.9678 - val\_loss: 0.3192 - val\_accuracy: 0.9130

Epoch 8/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0747 - accuracy: 0.9725 - val\_loss: 0.3574 - val\_accuracy: 0.9112

Epoch 9/20

1719/1719 [==============================] - 58s 34ms/step - loss: 0.0663 - accuracy: 0.9763 - val\_loss: 0.3954 - val\_accuracy: 0.9118

Epoch 10/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0578 - accuracy: 0.9794 - val\_loss: 0.4581 - val\_accuracy: 0.9108

Epoch 11/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0540 - accuracy: 0.9811 - val\_loss: 0.5012 - val\_accuracy: 0.9084

Epoch 12/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0499 - accuracy: 0.9827 - val\_loss: 0.5428 - val\_accuracy: 0.9126

Epoch 13/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0545 - accuracy: 0.9817 - val\_loss: 0.5112 - val\_accuracy: 0.9136

Epoch 14/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0455 - accuracy: 0.9848 - val\_loss: 0.5420 - val\_accuracy: 0.9124

Epoch 15/20

1719/1719 [==============================] - 61s 35ms/step - loss: 0.0462 - accuracy: 0.9855 - val\_loss: 0.5824 - val\_accuracy: 0.9100

Epoch 16/20

1719/1719 [==============================] - 60s 35ms/step - loss: 0.0393 - accuracy: 0.9874 - val\_loss: 0.5796 - val\_accuracy: 0.9114

Epoch 17/20

1719/1719 [==============================] - 59s 34ms/step - loss: 0.0492 - accuracy: 0.9861 - val\_loss: 0.7633 - val\_accuracy: 0.9094

Epoch 18/20

1719/1719 [==============================] - 60s 35ms/step - loss: 0.0433 - accuracy: 0.9872 - val\_loss: 0.7485 - val\_accuracy: 0.9152

Epoch 19/20

1719/1719 [==============================] - 59s 35ms/step - loss: 0.0405 - accuracy: 0.9878 - val\_loss: 0.6674 - val\_accuracy: 0.9130

Epoch 20/20

1719/1719 [==============================] - 62s 36ms/step - loss: 0.0393 - accuracy: 0.9890 - val\_loss: 0.7023 - val\_accuracy: 0.9162

313/313 [==============================] - 3s 10ms/step - loss: 0.7310 - accuracy: 0.9089

准确率可以稳定在90%以上

但是训练集准确率98%验证集准确率91%，还是有点过拟合

于是打算从新开始慢慢增加次数，3\*4次之后

Epoch 1/3

1719/1719 [==============================] - 79s 46ms/step - loss: 0.1926 - accuracy: 0.9274 - val\_loss: 0.2079 - val\_accuracy: 0.9224

Epoch 2/3

1719/1719 [==============================] - 79s 46ms/step - loss: 0.1868 - accuracy: 0.9302 - val\_loss: 0.2124 - val\_accuracy: 0.9236

Epoch 3/3

1719/1719 [==============================] - 79s 46ms/step - loss: 0.1773 - accuracy: 0.9342 - val\_loss: 0.1988 - val\_accuracy: 0.9302

测试集

313/313 [==============================] - 4s 11ms/step - loss: 0.2172 - accuracy: 0.9219

1. 结论

卷积神经网络的准确率最高有92%缺点是运行时间较慢，普通model都在88%左右且调整参数变化不大。