# 初学者完整学习流程实现手写数字识别案例

文档版本：V1.0.0

更新日期：2023.10.20

# 目录

1 引言 ..... 1
2 用户需具备的基础知识 ..... 1
3 数字识别模型搭建 ..... 1
3.1 数据准备 ..... 1
3.2 搭建神经网络 ..... 2
3.3 训练网络模型 ..... 3
3.4 模型测试 ..... 3
3.5 模型格式转换 ..... 3
3.6 数字识别模型完整代码 ..... 4
4 Aidlux 上实现模型部署 ..... 4
4.1 识别图片预处理 ..... 4
4.2 tflite 模型部署 ..... 5
4.3 Aidlux 上实现模型部署完整代码 ..... 6
5 图片显示识别结果 ..... 7
5.1 调用安卓平台相机拍摄 ..... 7
5.2 屏幕中显示结果 ..... 7
5.3 图片显示识别结果完整代码 ..... 8
6 参考资料 ..... 9

# 1 引言

本文主要介绍如何使用 SC171 开发套件实现实时拍摄数字照片并识别，会带领用户一步一步，从数字识别模型的搭建，到将模型部署在 SC171 开发套件上，最后调用安卓平台的相机，实时的拍摄照片并识别出照片里的数字，完成数字识别的案例。

本文中的所有代码均分享至百度网盘:

https://pan.baidu.com/s/100clM97sv1ZhGKnRNLyEJw?pwd=1va6

# 2 用户需具备的基础知识

使用 SC171 开发套件实现 AI 工程，首先必须掌握 **python 语言**，具备一定的 python 编程基础。因为人工智能开发基本都是使用 python 来进行代码编写，在这里推荐一些课程与书籍供大家学习借鉴，课程与书籍见本文第 6 章节。

人工智能在图像处理、语音识别、自然语言处理上都有着广泛的应用，本文的案例手写数字识别属于是人工智能开发的图像处理领域，图像处理又分为图像分割、图像分类、目标检测三个方向。读者需掌握**人工智能开发的基础**，理解图像分割、图像分类、目标检测这三者的区别，然后就可以跟着本文一步一步实现手写数字识别工程部署在 SC171 开发套件上的案例了。关于人工智能开发同样推荐一些课程与书籍，见第 6 章节。

# 3 数字识别模型搭建

**以下显示黑色背景的代码均为 PC 下的代码**

## 3.1 数据准备

从 Keras 库中引入手写数字数据集 MNIST，他是一个包含 60000 个训练样本和 10000 个测试样本的数据集。使用 load_data()函数将 MNIST 数据集加载到程序中，并将数据集分为训练集和测试集，其中 train_images、train_labels 为训练集，test_images、test_labels 为测试集。

由于神经网络只接受数值型数据，所以需要将手写数字图像转换为数据张量格式。也就是将每张图像转换为 28×28 的矩阵，并进行归一化处理。



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

```
from keras.utils import to_categorical
from keras import models, layers
from keras.optimizers import RMSprop
from keras.datasets import mnist
import tensorflow as tf
# 加载数据集
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# 数据处理
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

## 3.2 搭建神经网络

模型采用了 Sequential 模型

- C1 层是卷积层，包含 6 个特征图，是由 6 个 3x3 的卷积核对输入图像卷积得到。
- S2 层是一个下采样层，包含 6 个特征图，是由 C1 层的特征图经过 2x2 的窗口进行平均池化
- C3 层是卷积层，包含 16 个特征图，是由 16 个 3x3 的卷积核对 S2 进行卷积得到
- S4 是一个下采样层，包含 16 个特征图，是由 C3 层的特征图经过 2x2 的窗口进行平均池化
- C5 是卷积层包含 120 个特征图，是由 120 个 3x3 的卷积核对 S4 进行卷积得到。
- F6 是包含 84 个神经元的全连接层，采用 relu 激活函数
- 最后添加一个分类层，使用 softmax 激活。输出 0-9 是个数字，所以单元数为 10

模型搭建完成后，进行编译模型。使用交叉熵作为损失函数，RMSprop 优化器进行训练，在训练过程中监测模型的精度。

```
# 搭建LeNet网络
def LeNet():
    network = models.Sequential()
    network.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
    network.add(layers.AveragePooling2D((2, 2)))
    network.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
    network.add(layers.AveragePooling2D((2, 2)))
    network.add(layers.Conv2D(filters=120, kernel_size=(3, 3), activation='relu'))
    network.add(layers.Flatten())
    network.add(layers.Dense(84, activation='relu'))
    network.add(layers.Dense(10, activation='softmax'))
    return network
network = LeNet()
network.compile(optimizer=RMSprop(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

## 3.3 训练网络模型

使用 fit()方法对构建好的神经网络进行训练，epochs 表示训练多少个回合，batch_size 表示每次训练给多大的数据

```
# 训练网络，用fit函数，epochs表示训练多少个回合， batch_size表示每次训练给多大的数据
network.fit(train_images, train_labels, epochs=20, batch_size=128, verbose=2)
```

## 3.4 模型测试

使用 evaluate()方法对模型进行测试，并返回测试误差和测试准确率。

```
# 在测试集上测试一下模型的性能
test_loss, test_accuracy = network.evaluate(test_images, test_labels)
print("test_loss:", test_loss, "test_accuracy:", test_accuracy)
```

下面为模型测试的结果

```
Epoch 16/20
469/469 - 6s - loss: 0.0077 - accuracy: 0.9976 - 6s/epoch - 13ms/step
Epoch 17/20
469/469 - 6s - loss: 0.0076 - accuracy: 0.9976 - 6s/epoch - 13ms/step
Epoch 18/20
469/469 - 6s - loss: 0.0070 - accuracy: 0.9976 - 6s/epoch - 13ms/step
Epoch 19/20
469/469 - 6s - loss: 0.0061 - accuracy: 0.9982 - 6s/epoch - 13ms/step
Epoch 20/20
469/469 - 7s - loss: 0.0051 - accuracy: 0.9984 - 7s/epoch - 14ms/step
313/313 [==============================] - 1s 2ms/step - loss: 0.0474 - accuracy: 0.9890
test_loss: 0.04737924039363861 test_accuracy: 0.9890000224113464
```

## 3.5 模型格式转换

Keras 支持保存 HDF5 文件，也就是.h5 文件，这个文件包含模型的体系结构，权重值和 compile()信息。因为后续我们需要将模型部署在 Linux 环境中，所以还需要将模型转换成.tflite 格式

```
# 模型保存为.h5格式
network.save('MyModel.h5')
# 模型转换为.tflite格式
keras_file = 'MyModel.h5'
tf.keras.models.save_model(network, keras_file)
model = tf.keras.models.load_model(keras_file)
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open('MyModel.tflite', 'wb') as f:
    f.write(tflite_model)
```



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com | 传真: +86 755-26520841 官网: www.fibocom.com.cn

此时，工程文件夹中会出现 MyModel.h5 和 MyModel.tflite 两个文件，其中 MyModel.tflite 是我们需要部署到 Aidlux 软件中的文件。

| | | | |
|---|---|---|---|
| MyModel.h5 | 2023/10/20 9:49 | H5 文件 | 902 KB |
| MyModel.tflite | 2023/10/20 9:49 | TFLITE 文件 | 434 KB |

## 3.6 数字识别模型完整代码

见网盘资料文件：《LeNet.py》

# 4 Aidlux 上实现模型部署

**以下显示灰色背景的代码均为 Aidlux 下的代码**

## 4.1 识别图片预处理

首先需要清楚为什么要对识别的图片做预处理，因为需要将图片保持与训练集的格式一致，模型才能精准识别出结果。

预处理过程，首先读取需要识别的图片，重新调整图片至合适的大小，并做灰度处理，然后将图片反色并做二值化处理。

```
#*对拍摄的数字图片预处理*
import cv2
from camera_android import *
import imutils

img = cv2.imread(imageFn)                    #读取图片，imageFn为图片路径
img = imutils.resize(img,width=195)
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)     #灰度图
```

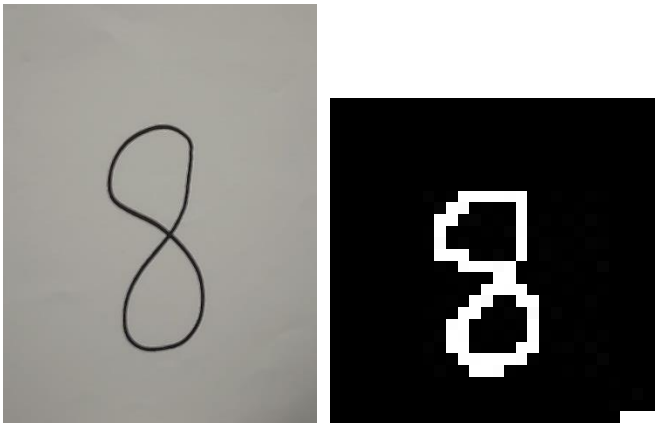
```
#创建空白数组
for row in range(260):
    for col in range(195):
        gray_img[row][col] = 255 - gray_img[row][col]#反色
# 二值化，(127,255)为阈值
retval,bit_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY)
bit_img = cv2.resize(bit_img, (28,28), interpolation= cv2.INTER_AREA)
retval,bit_img = cv2.threshold(bit_img, 20, 255, cv2.THRESH_BINARY)

#cv2.imwrite("77.jpg",bit_img)                          #保存图片
cv2.waitKey(0)
```



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

下面为预处理前和预处理后的图片对比



## 4.2 tflite 模型部署

首先将训练好并导出的 MyModel.tflite 文件存入到 Aidlux 软件下的工程文件夹中，然后新建一个 MyModel.py 文件，进行 tflite 模型部署。

- 第一步加载 tflite 文件模型，获取模型输入输出张量

```
#*训练好的模型做识别*
import numpy as np
import tensorflow as tf
import cv2
from tensorflow.keras.preprocessing import image
from picture import *

# 加载tflite文件模型
tflite_model = tf.lite.Interpreter(model_path="MyModel.tflite")
tflite_model.allocate_tensors()

# 获取输入和输出张量
input_details = tflite_model.get_input_details()
output_details = tflite_model.get_output_details()
```

- 第二步加载图片至内存中并 resize 和增加维度

```
cv2.imwrite("77.jpg",bit_img)
image_path_test = '77.jpg'#图片路径

#加载图片至内存中并resize和增加维度
img = image.load_img(image_path_test, target_size=(28, 28))
img = image.img_to_array(img)
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = np.expand_dims(img, 2)
#print(img.shape)#这里直接打印将img转换为数组后的数据维度 (28,28,1)
img = np.expand_dims(img, axis=0)#因为模型的输入是要求四维的，所以我们需要将输入图片增加一个维度
```



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

- 第三步模型预测

```
#模型预测
tflite_model.set_tensor(input_details[0]['index'], img)
tflite_model.invoke()

output_data = tflite_model.get_tensor(output_details[0]['index'])
print(output_data)
```

- 第四步读取标签文件，并根据模型的预测结果进行标签的预测，标签文件为 0~9 的 txt 文件

label.txt

```
0
1
2
3
4
5
6
7
8
9
```

标签文件

```
#读取标签文件，返回列表
def read_label_list():
    with open('label.txt', 'r',encoding="utf8") as f:
        data = f.read().splitlines()
    return data

#标签预测
#print(read_label_list())
w = np.argmax(output_data)#值最大的位置
lable_list = read_label_list()#读取标签列表
print(lable_list[w])
```

- 识别结果如下

```
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
8
```

## 4.3 Aidlux 上实现模型部署完整代码

- 识别图片预处理代码见：《picture.py》
- tflite 模型部署代码见：《MyModel.py》
- 标签文件见：《label.txt》



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

# 5 图片显示识别结果

## 5.1 调用安卓平台相机拍摄

调用安卓的相机，并拍摄照片存储至位置：/sdcard/myphoto.jpg

```
#*拍摄需要识别的数字照片*
import android    #调用安卓库
#import cv2

#调用安卓的相机，并拍摄照片存储至位置：/sdcard/myphoto.jpg
droid = android.Android()
imageFn = '/sdcard/myphoto.jpg'
droid.cameraInteractiveCapturePicture(imageFn)
#img = cv2.imread(imageFn)
```

## 5.2 屏幕中显示结果

- 读取拍摄的图片

```
#*在屏幕中显示识别的结果*
#from camera_android import *
#import cv2
#import numpy as np
from MyModel import *
from picture import *
from cvs import *
import os

#读取拍摄的图片
RGB_img=cv2.imread(imageFn)
blank_img = np.zeros(shape=(RGB_img.shape[0],RGB_img.shape[1],3), dtype=np.uint8)
font = cv2.FONT_HERSHEY_SIMPLEX
```

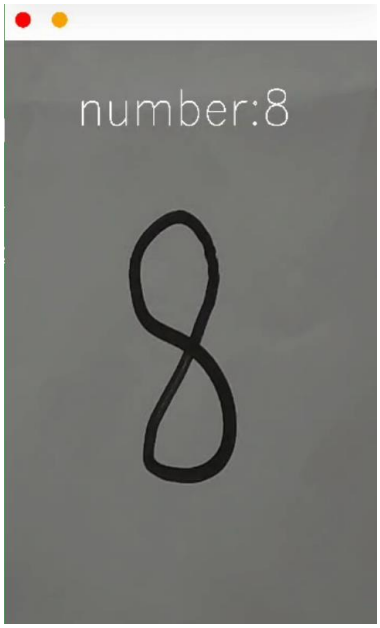


深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

- 在图片中添加文字水印，文字内容为：number：+（识别到的数字结果），org 为水印位置，fontScale 为水印字体大小，color 为字体颜色

```
# 添加水印的文字内容
text1 = "number:"+lable_list[w]
cv2.putText(blank_img,text=text1,org=(600, 600),
            fontFace=font,fontScale= 12,
            color=(255,255,255),thickness=12,lineType=cv2.LINE_4)
blended = cv2.addWeighted(src1=RGB_img, alpha=0.7,
                          src2=blank_img, beta=1, gamma = 2)
cv2.imwrite("blended1.jpg",blended)
cap = cvs.VideoCapture("blended1.jpg")
blended1 = cap.read()
cap.imshow(blended1)
os.remove("blended1.jpg")
```

- 结果如下




## 5.3 图片显示识别结果完整代码

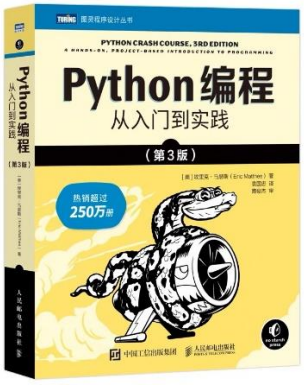
- 调用安卓平台相机拍摄代码见：《camera_android.py》
- 屏幕中显示结果代码见：《test.py》



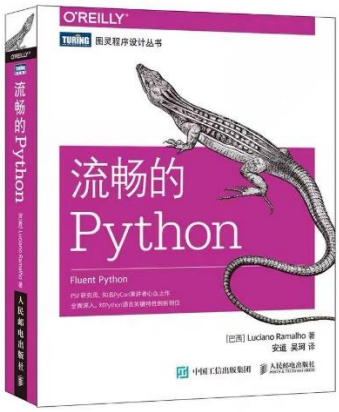
深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

# 6 参考资料

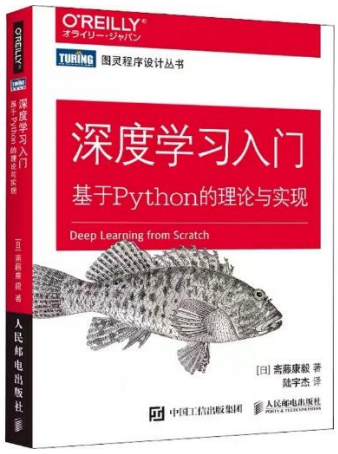
- Python 视频教程:《黑马程序员 python 教程》
  https://www.bilibili.com/video/BV1qW4y1a7fU/?spm_id_from=333.337.search-card.all.click&vd_source=0e4d9789d1a427bff36bc24af1780cd9
- Python 书籍推荐:
  《python 编程:从入门到实践》




《流畅的 python》



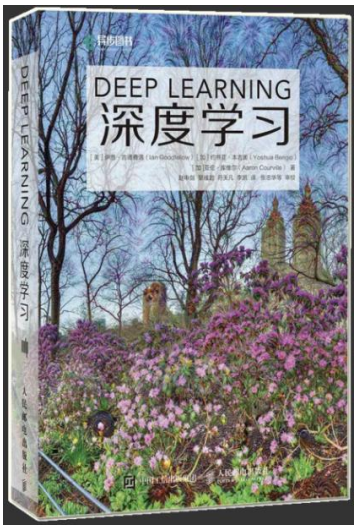

- 人工智能书籍推荐:
  《深度学习入门:基于 python 的理论与实现》



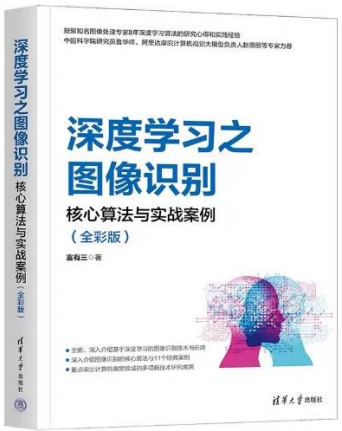



深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

《深度学习》




- 图像处理类书籍推荐:
  《深度学习之图像识别》






深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn