# 在 PyCharm 中运行 MNIST 数字图像分类模型的完整流程

## 一、项目环境搭建

### 1.1 创建新项目

打开 PyCharm，依次选择 File -> New Project。在弹出的对话框里，确定项目的存储位置，建议为项目创建独立的虚拟环境。选择 Python Interpreter 时，点击 Add Interpreter，接着选择 Virtualenv Environment，PyCharm 会自动完成虚拟环境的创建与管理。最后点击 Create 完成项目创建。

### 1.2 安装依赖库

在 PyCharm 的终端（Terminal）中执行以下命令来安装所需的依赖库：

```
pip install tensorflow flask numpy
```

## 二、代码准备

### 2.1 模型训练代码（model_dev.py）

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
# 加载 MNIST 数据集
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
# 数据预处理
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255
```

```
# 构建模型
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
# 编译模型
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# 训练模型
model.fit(train_images, train_labels, epochs=5, batch_size=64)
# 评估模型
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test accuracy: {test_acc}')
# 保存模型
model.save('mnist_model.h5')
```

## 2.2 模型服务代码（app.py）

```
from flask import Flask, request, jsonify
import tensorflow as tf
import numpy as np
app = Flask(__name__)
# 加载模型
model = tf.keras.models.load_model('mnist_model.h5')
@app.route('/predict', methods=['POST'])
```

```
def predict():
    data = request.get_json()
    if 'image' not in data:
        return jsonify({'error': 'Missing "image" key in JSON payload'}), 400
    image = np.array(data['image']).reshape((1, 28, 28, 1)).astype('float32') / 255
    try:
        prediction = model.predict(image)
        return jsonify({'prediction': prediction.tolist()})
    except Exception as e:
        return jsonify({'error': f'Error during prediction: {str(e)}'}), 500
if __name__ == '__main__':
    app.run(debug=True)
```

# 三、运行代码

## 3.1 训练模型

在 PyCharm 的项目文件列表中，右键点击 model_dev.py 文件，选择 Run 'model_dev.py'。运行过程中的输出如下：

### 数据加载

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

### 模型训练

每个 epoch 会显示训练进度、损失值和准确率。

```
Epoch 1/5
938/938 [==============================] - 4s 4ms/step - loss: 0.2543 - accuracy: 0.9236
Epoch 2/5
```

```
938/938 [==============================] - 4s 4ms/step - loss: 0.0849 - accuracy: 0.9741
Epoch 3/5
938/938 [==============================] - 4s 4ms/step - loss: 0.0578 - accuracy: 0.9819
Epoch 4/5
938/938 [==============================] - 4s 4ms/step - loss: 0.0427 - accuracy: 0.9864
Epoch 5/5
938/938 [==============================] - 4s 4ms/step - loss: 0.0329 - accuracy: 0.9896
```

### 模型评估

显示测试集的损失值和准确率。

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0337 - accuracy: 0.9903
Test accuracy: 0.9903000211715698
```

### 模型保存

运行结束后，项目目录下会生成 mnist_model.h5 文件。

## 3.2 启动模型服务

右键点击 app.py 文件，选择 Run 'app.py'。控制台会输出 Flask 应用的启动信息：

```
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

## 3.3 测试模型服务

可以使用 Python 的 requests 库发送测试请求。在 PyCharm 中创建一个新的 Python 文件（例如 test.py），并编写以下代码：

```
import requests
import numpy as np
# 生成一个随机的 MNIST 图像数据
image = np.random.randint(0, 255, size=(28, 28)).tolist()
# 发送 POST 请求
response = requests.post('http://127.0.0.1:5000/predict', json={'image': image})
# 打印响应结果
print(response.json())
```

运行 test.py 文件，根据请求是否成功，会得到不同的输出：

## 成功响应

返回模型的预测结果。

```
{'prediction': [[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.55]]}
```

## 错误响应

如果请求格式有误或预测过程中出现错误，会返回错误信息。

```
{'error': 'Missing "image" key in JSON payload'}
```

通过以上步骤，就可以在 PyCharm 中完成 MNIST 数字图像分类模型的训练、服务启动和测试。