《大数据技术与机器学习》课程实验报告

实验名称	神经网络			实验序号	实验 4	实验日期	2025/10/30
姓　名	毛梓进	院系	计算机	班　级	221042Y2	学　号	221042Y234
专　业	软件工程			指导教师	朱栩	成　绩	
评语							

一、实验目的和要求

实验名称：基于神经网络的 MNIST 手写数字识别
数据集：MNIST 手写数字数据集（内置在 Keras 中）
工具：Python 3.7+、TensorFlow 2.x、Matplotlib
神经网络结构：输入层 → 全连接隐藏层 → 输出层
实验亮点：简洁实现（<50 行代码）、可视化训练过程、实时预测展示

1. 环境安装与配置。
2. 数据加载与预处理。
3. 构建浅层神经网络。
4. 模型训练与评估
5. 模型预测与交互测试

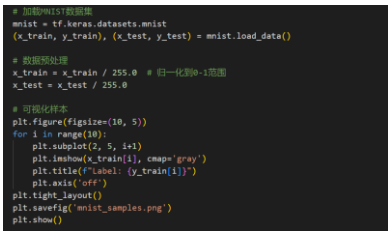
二、实验预习内容

- **神经网络结构：**输入层→隐藏层→输出层
- **MNIST 数据集：**6 万张 28×28 手写数字图片
- **全连接层原理：**每个神经元与上一层全连接
- **激活函数作用：**ReLU（隐藏层）、Softmax（输出层）
- **输入层：**接收 784 个像素值
- **隐藏层：**特征提取和学习
- **输出层：**10 个数字的概率输出

三、实验项目摘要

1. 加载数据集与模型构造

代码如下



加载数据集

```
# 创建神经网络模型
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),  # 输入层：将28x28图像展平
    tf.keras.layers.Dense(128, activation='relu'),  # 隐藏层：128个神经元
    tf.keras.layers.Dense(10, activation='softmax') # 输出层：10个数字类别
])

# 编译模型
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 打印模型结构
model.summary()
```

搭建神经网络

设置优化器、损失函数、量化指标

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100,480
dense_1 (Dense)	(None, 10)	1,290

Total params: 101,770 (397.54 KB)
Trainable params: 101,770 (397.54 KB)
Non-trainable params: 0 (0.00 B)

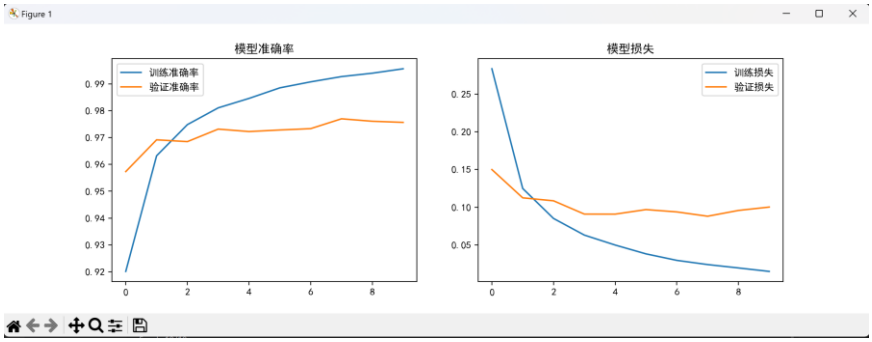
神经网络参数结构

2. 训练并验证

总体包含一个输入层一个输出层以及一个隐藏层

```python
# 训练模型
history = model.fit(
    x_train, y_train,
    epochs=10,
    batch_size=32,
    validation_split=0.2  # 使用20%训练数据作为验证集
)

# 评估模型
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"\n测试准确率: {test_acc:.4f}")
```

训练并评估



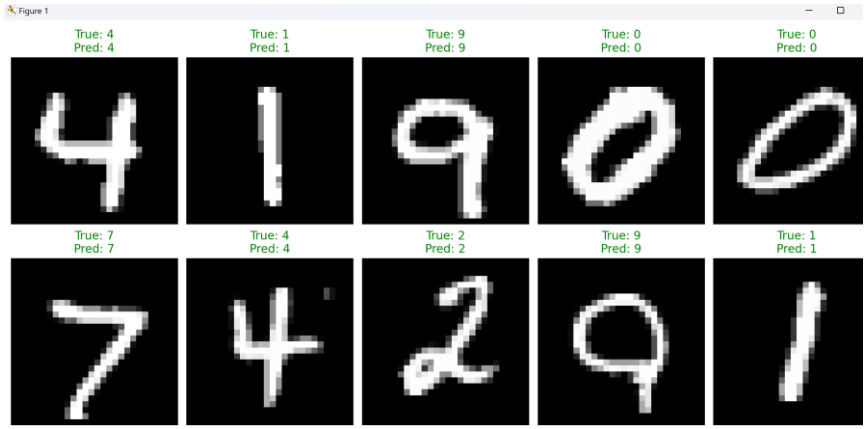
训练过程中准确率和损失的变化

3. 预测

```python
# 预测结果
predictions = model.predict(samples)
predicted_labels = np.argmax(predictions, axis=1)

# 可视化预测结果
plt.figure(figsize=(15, 6))
for i, (image, true_label, pred_label) in enumerate(zip(samples, y_test[sample_indices], predicted_labels)):
    plt.subplot(2, 5, i+1)
    plt.imshow(image, cmap='gray')
    plt.title(f"True: {true_label}\nPred: {pred_label}",
              color='green' if true_label == pred_label else 'red')
    plt.axis('off')
plt.tight_layout()
plt.savefig('predictions.png')
plt.show()
```

预测并展示

预测结果



预测结果展示

记录最终测试准确率

思考：神经网络各层的作用是什么？

答：分为输入层、输出层和隐藏层。输入层负责接受外部数据输入，输出层则是输出具体的需要求(分类等任务)，隐藏层负责对输入的特征提取。

尝试：修改隐藏层神经元数量（64 或 256）观察效果变化

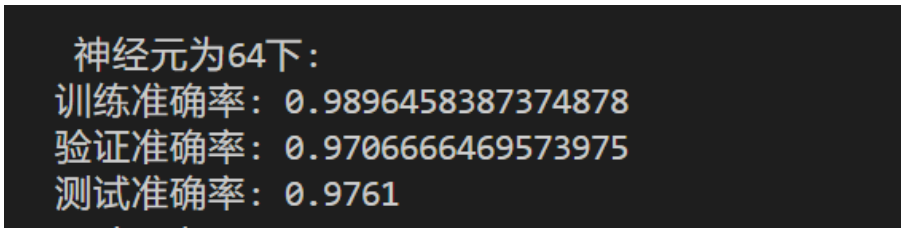
```python
for neurCount in [64, 128, 256] :
    # 创建神经网络模型
    model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),  # 输入层：将28x28图像展平
        tf.keras.layers.Dense(neurCount, activation='relu'),  # 隐藏层：128个神经元
        tf.keras.layers.Dense(10, activation='softmax') # 输出层：10个数字类别
    ])
```

对每个参数分别建立模型

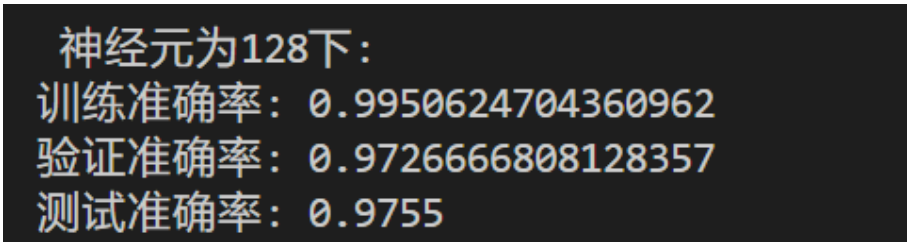
```
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow
e compiler flags.
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9047 - loss: 0.3399 - val_accuracy: 0.9457 - val_loss: 0.1930
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9505 - loss: 0.1693 - val_accuracy: 0.9552 - val_loss: 0.1491
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9646 - loss: 0.1209 - val_accuracy: 0.9628 - val_loss: 0.1233
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9726 - loss: 0.0933 - val_accuracy: 0.9663 - val_loss: 0.1086
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9769 - loss: 0.0773 - val_accuracy: 0.9688 - val_loss: 0.1018
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9810 - loss: 0.0632 - val_accuracy: 0.9691 - val_loss: 0.1011
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9833 - loss: 0.0538 - val_accuracy: 0.9716 - val_loss: 0.0959
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9865 - loss: 0.0450 - val_accuracy: 0.9712 - val_loss: 0.0951
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9878 - loss: 0.0394 - val_accuracy: 0.9669 - val_loss: 0.1042
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9896 - loss: 0.0327 - val_accuracy: 0.9707 - val_loss: 0.0945
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 923us/step - accuracy: 0.9761 - loss: 0.0854

  神经元为64下:
 训练准确率: 0.9896458387374878
 验证准确率: 0.9706666469573975
 测试准确率: 0.9761
Epoch 1/10
```

训练

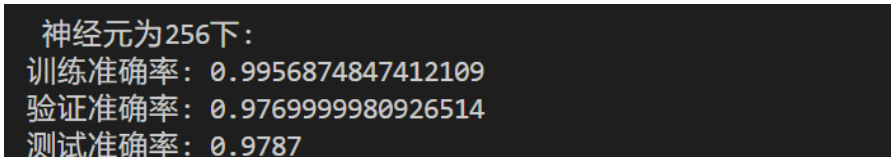
结果



神经元为 64



神经元为 128



神经元为 256

四、实验结果与分析

使用三种神经元数量如下

实验结论

1. 神经元数量增加确实提升性能:从 64→256 神经元,测试准确率从 97.61%提升到 97.87%
2. 没有严重过拟合：所有配置的训练-测试差距都在 2%以内，说明模型训练良好
3. 推荐使用 256 神经元配置：在该问题中，更大的模型容量带来了更好的泛化性能

成功实现了基于全连接神经网络的 MNIST 手写数字识别，通过对比不同神经元数量(64, 128, 256)对模型性能的影响。

📊 核心发现

1. 神经元数量与性能关系：
   - 训练准确率：256 神经元(99.57%) > 128 神经元(99.51%) > 64 神经元(98.96%)
   - 测试准确率：256 神经元(97.87%) > 64 神经元(97.61%) > 128 神经元(97.55%)
2. 过拟合控制：
   - 所有模型都表现出良好的泛化能力
   - 训练-测试准确率差距均在合理范围内(1.35%-1.96%)
3. 最佳配置：256 个神经元
   - 测试准确率最高(97.87%)
   - 泛化能力最好

注：空间不够，可以增加页码。