《人工智能与大数据管理》课程

实验报告

网络与信息安全学院

班　　级：2118021

姓　　名：盖 乐

学　　号：21009200991

提交时间：2023 年 12 月 18 日

基于神经网络的 MNIST 手写数字识别

一、实验目的

- 掌握运用神经网络模型解决有监督学习问题
- 掌握机器学习中常用的模型训练测试方法
- 了解不同训练方法的选择对测试结果的影响

二、实验内容

MNIST 数据集

本实验采用的数据集 MNIST 是一个手写数字图片数据集，共包含图像和对应的标签。数据集中所有图片都是 28x28 像素大小，且所有的图像都经过了适当的处理使得数字位于图片的中心位置。MNIST 数据集使用二进制方式存储。图片数据中每个图片为一个长度为 784（28x28x1，即长宽 28 像素的单通道灰度图）的一维向量，而标签数据中每个标签均为长度为 10 的一维向量。

分层采样方法

分层采样（或分层抽样，也叫类型抽样）方法，是将总体样本分成多个类别，再分别在每个类别中进行采样的方法。通过划分类别，采样出的样本的类型分布和总体样本相似，并且更具有代表性。在本实验中，MNIST 数据集为手写数字集，有 0~9 共 10 种数字，进行分层采样时先将数据集按数字分为 10 类，再按同样的方式分别进行采样。

神经网络模型评估方法

通常，我们可以通过实验测试来对神经网络模型的误差进行评估。为此，需要使用一个测试集来测试模型对新样本的判别能力，然后以此测试集上的测试误差作为误差的近似值。两种常见的划分训练集和测试集的方法：

留出法（hold-out）直接将数据集按比例划分为两个互斥的集合。划分时为尽可能保持数据分布的一致性，可以采用分层采样（stratified sampling）的方式，使得训练集和测试集中的类别比例尽可能相似。需要注意的是，测试集在整个数据集上的分布如果不够均匀还可能引入额外的偏差，所以单次使用留出法得到的估计结果往往不够稳定可靠。在使用留出法时，一般要采用若干次随机划分、重

复进行实验评估后取平均值作为留出法的评估结果。

k 折交叉验证法（k-fold cross validation）先将数据集划分为 k 个大小相似的互斥子集，每个子集都尽可能保持数据分布的一致性，即也采用分层采样（stratified sampling）的方法。然后，每次用 k-1 个子集的并集作为训练集，余下的那个子集作为测试集，这样就可以获得 k 组训练集和测试集，从而可以进行 k 次训练和测试。最终返回的是这 k 个测试结果的均值。显然，k 折交叉验证法的评估结果的稳定性和保真性在很大程度上取决于 k 的取值。k 最常用的取值是 10，此外常用的取值还有 5、20 等。

三、实验方法设计

介绍实验中程序的总体设计方案、关键步骤的编程方法及思路，主要包括:

1）模型构建的程序设计（伪代码或源代码截图）及说明解释 （10 分）

训练集与测试集均来自 mnist.train，并且经过打乱，验证集则来自 mnist.validation。

```
# 读取数据集
mnist = input_data.read_data_sets('./mnist_dataset', one_hot=True)
# 训练集
total_images = mnist.train.images
total_labels = mnist.train.labels
total_images, total_labels = shuffer_images_and_labels(total_images, total_labels)
# 验证集
validation_images = mnist.validation.images
validation_labels = mnist.validation.labels
validation_images, validation_labels = shuffer_images_and_labels(validation_images, validation_labels)
```

每个图片是一个长度为 784（28x28x1，即长宽 28 像素的单通道灰度图）的一维向量，而标签数据中每个标签是长度为 10 的一维向量。因此，占位符设置如下：

```
# Input layers (28*28*1)
x = tf.placeholder(tf.float32, [None, 784], name="X")
```

```
# 0-9 => 10 numbers
y = tf.placeholder(tf.float32, [None, 10], name="Y")
```

为使得模型的准确率达到 97%以上，选择了两层隐藏层，神经元分别为 256 和 64，激活函数选择的是 relu。

```
# 2 Hidden layers
h1 = fcn_layer(inputs=x,
               input_dim=784,
               output_dim=256,
               activation=tf.nn.relu)
h2 = fcn_layer(inputs=h1,
               input_dim=256,
               output_dim=64,
               activation=tf.nn.relu)
# Output layers
forward = fcn_layer(inputs=h2,
                    input_dim=64,
                    output_dim=10,
                    activation=None)
pred = tf.nn.softmax(forward)
```

为了方便，在此之前定义了一个层函数：

```
#FCN 全卷积神经网络
def fcn_layer(inputs,  # input data
              input_dim,  # Input numbers of Neurons
              output_dim,  # Output numbers of Neurons
              activation=None):  # activation function
    # Random numbers that generate data that is more than twice the standard deviation will be replaced here
    W = tf.Variable(tf.truncated_normal([input_dim, output_dim], stddev=0.1))
    b = tf.Variable(tf.zeros([output_dim]))  # init as 0

    XWb = tf.matmul(inputs, W) + b

    if activation is None:
        outputs = XWb
    else:
        outputs = activation(XWb)
```

```
16.
17.     return outputs
```

交叉熵损失函数刻画的是两个概率分布之间的距离，交叉熵越小，两个概率的分布越接近。

```
1.  loss_function = tf.reduce_mean(
2.      tf.nn.softmax_cross_entropy_with_logits(logits=forward, labels=y))
```

训练时的参数、选择的优化器以及定义的准确率如下所示：

```
1.  train_epochs = 20  # Train times
2.  batch_size = 100  # single batch train size
3.  learning_rate = 0.001  # learning rate
4.
5.  optimizer = tf.train.AdamOptimizer(learning_rate).minimize(loss_function)
6.
7.  correct_prediction = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
8.
9.  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

2）模型迭代训练的程序设计（伪代码或源代码截图）及说明解释 （10 分）

模型共迭代 20 轮，每一轮使用训练集的全部数据对模型进行一次完整训练，因为 batch_size 定义为 100，因此每批次按照 100 个样本进行训练。迭代训练模型使用的是老师提供的 batch_iter( )函数。

```
1.  # images：训练集的 feature 部分
2.  # labels：训练集的 label 部分
3.  # batch_size：  每次训练的 batch 大小
4.  # epoch_num：  训练的 epochs 数
5.  # shuffle：  是否打乱数据
6.  # 使用示例：
7.  #   for (batchImages, batchLabels) in batch_iter(images_train, labels_train,
     batch_size, epoch_num, shuffle=True):
8.  #       sess.run(feed_dict={inputLayer: batchImages, outputLabel: batchLabel
    s})
9.  def batch_iter(images,labels, batch_size, epoch_num, shuffle=True):
10.
11.     data_size = len(images)
12.
```

```
    num_batches_per_epoch = int(data_size / batch_size)  # 样本数/batch 块大小,多出来的“尾数”，不要了

    for epoch in range(epoch_num):
        # Shuffle the data at each epoch
        if shuffle:
            shuffle_indices = np.random.permutation(np.arange(data_size))

            shuffled_data_feature = images[shuffle_indices]
            shuffled_data_label   = labels[shuffle_indices]
        else:
            shuffled_data_feature = images
            shuffled_data_label = labels

        for batch_num in range(num_batches_per_epoch):   # batch_num 取值 0 到 num_batches_per_epoch-1
            start_index = batch_num * batch_size
            end_index = min((batch_num + 1) * batch_size, data_size)

            yield (shuffled_data_feature[start_index:end_index] , shuffled_data_label[start_index:end_index])
```

3）模型训练过程中周期性测试的程序设计（伪代码或源代码截图）及说明解释（周期性测试指的是每训练 n 个 step 就对模型进行一次测试，得到准确率和 loss 值）（10 分）

```
accu_test = sess.run(accuracy, feed_dict={x: images_test, y: labels_test})
accu_validation = sess.run(accuracy, feed_dict={x: images_validation, y: labels_validation})
return accu_test,accu_validation
```

4）分层采样的程序设计（伪代码或源代码截图）及说明解释 （10 分）

在 hold_out 方法中，调用了 sklearn 内的函数

在 k 折交叉验证法中实现了以下分层采样：

```
total_images = [[] for _ in range(10)]
total_labels = [[] for _ in range(10)]

for i in range(len(images)):
    index = get_label(labels[i])
```

```
        total_images[index].append(images[i])
        total_labels[index].append(labels[i])

k_total_images = []
k_total_labels = []  # 大小为 k
for i in range(10):
    for j in range(k):
        k_total_images.append(total_images[i][int(j * len(total_images[i]) / k):int((j + 1) * len(total_images[i]) / k)])  # 长度为 k*10，里面的列表长度为 len(total_images[i])/k
        k_total_labels.append(total_labels[i][int(j * len(total_images[i]) / k):int((j + 1) * len(total_images[i]) / k)])
```

5）k 折交叉验证法的程序设计（伪代码或源代码截图）及说明解释 （10 分）

```
tmp_accu_test = 0
tmp_accu_vali = 0
for idex in range(k):
    X_test_images = k_total_images[idex]  # 大小为 1
    Y_test_labels = k_total_labels[idex]
    X_train_images = k_total_images  # 大小为 k-1
    Y_train_labels = k_total_labels
    del X_train_images[idex]  # 大小为 k-1 del 删除变量
    del Y_train_labels[idex]

    f_X_train_images = []
    f_Y_train_labels = []

    for i in range(len(X_train_images)):
        for j in range(len(X_train_images[i])):
            f_X_train_images.append(X_train_images[i][j])
            f_Y_train_labels.append(Y_train_labels[i][j])

    f_X_train_images, f_Y_train_labels = np.array(f_X_train_images), np.array(f_Y_train_labels)
    X_test_images, Y_test_labels = np.array(X_test_images), np.array(Y_test_labels)

    print("[-] k = {}，当前第{}组为测试集".format(k, idex+1))
    accu_test,accu_validation = train_and_test(f_X_train_images, f_Y_train_labels, X_test_images, Y_test_labels,
                                               vali_images, vali_labels)
```

```
    print("[*] Temp accuracy of test :", accu_test)
    print("[*] Temp accuracy of validation :", accu_validation)
    tmp_accu_test += accu_test
    tmp_accu_vali += accu_validation

print("[*] Average accuracy of test :", tmp_accu_test / k)
print("[*] Average accuracy of validation :", tmp_accu_vali / k)
```

四、实验结果展示

展示程序界面设计、运行结果及相关分析等，主要包括：

1）模型在验证集下的准确率（输出结果并截图）（10 分）

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.169802591       accuracy: 0.9552
[+] 02th train: loss: 0.133499905       accuracy: 0.9622
[+] 03th train: loss: 0.107449926       accuracy: 0.9676
[+] 04th train: loss: 0.092352159       accuracy: 0.9744
[+] 05th train: loss: 0.090602949       accuracy: 0.9730
[+] 06th train: loss: 0.093158841       accuracy: 0.9754
[+] 07th train: loss: 0.098221160       accuracy: 0.9738
[+] 08th train: loss: 0.094697833       accuracy: 0.9740
[+] 09th train: loss: 0.093006872       accuracy: 0.9772
[+] 10th train: loss: 0.092334919       accuracy: 0.9768
[+] 11th train: loss: 0.095769480       accuracy: 0.9768
[+] 12th train: loss: 0.097406976       accuracy: 0.9776
[+] 13th train: loss: 0.115145423       accuracy: 0.9766
[+] 14th train: loss: 0.101600431       accuracy: 0.9784
[+] 15th train: loss: 0.115902394       accuracy: 0.9740
[+] 16th train: loss: 0.119874850       accuracy: 0.9768
[+] 17th train: loss: 0.103089087       accuracy: 0.9774
[+] 18th train: loss: 0.122543491       accuracy: 0.9736
[+] 19th train: loss: 0.132474452       accuracy: 0.9744
[+] 20th train: loss: 0.103956126       accuracy: 0.9810
[+] Train finished successfully. It takes: 10.87s
[*] Accuracy of test : 0.981
[*] Accuracy of validation : 0.978
```

2）不同模型参数（隐藏层数、隐藏层节点数）对准确率的影响和分析 （10 分）

①两层隐藏层：第一层神经元为 256，第二层神经元为 64，激活函数为 relu，交叉熵损失函数为 softmax_cross_entropy_with_logits，训练轮数为 20，批次大小为 100，学习率为 0.001，所选的优化器为 Adam。

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.169802591       accuracy: 0.9552
[+] 02th train: loss: 0.133499905       accuracy: 0.9622
[+] 03th train: loss: 0.107449926       accuracy: 0.9676
[+] 04th train: loss: 0.092352159       accuracy: 0.9744
[+] 05th train: loss: 0.090602949       accuracy: 0.9730
[+] 06th train: loss: 0.093158841       accuracy: 0.9754
[+] 07th train: loss: 0.098221160       accuracy: 0.9738
[+] 08th train: loss: 0.094697833       accuracy: 0.9740
[+] 09th train: loss: 0.093006872       accuracy: 0.9772
[+] 10th train: loss: 0.092334919       accuracy: 0.9768
[+] 11th train: loss: 0.095769480       accuracy: 0.9768
[+] 12th train: loss: 0.097406976       accuracy: 0.9776
[+] 13th train: loss: 0.115145423       accuracy: 0.9766
[+] 14th train: loss: 0.101600431       accuracy: 0.9784
[+] 15th train: loss: 0.115902394       accuracy: 0.9740
[+] 16th train: loss: 0.119874850       accuracy: 0.9768
[+] 17th train: loss: 0.103089087       accuracy: 0.9774
[+] 18th train: loss: 0.122543491       accuracy: 0.9736
[+] 19th train: loss: 0.132474452       accuracy: 0.9744
[+] 20th train: loss: 0.103956126       accuracy: 0.9810
[+] Train finished successfully. It takes: 10.87s
[*] Accuracy of test : 0.981
[*] Accuracy of validation : 0.978
```

②三层隐藏层：第一层神经元为 256，第二层神经元为 64，第三层神经元为 32。

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.178569332       accuracy: 0.9446
[+] 02th train: loss: 0.122868359       accuracy: 0.9636
[+] 03th train: loss: 0.105596341       accuracy: 0.9692
[+] 04th train: loss: 0.098489381       accuracy: 0.9716
[+] 05th train: loss: 0.098945297       accuracy: 0.9710
[+] 06th train: loss: 0.082749471       accuracy: 0.9750
[+] 07th train: loss: 0.112535208       accuracy: 0.9718
```

```
[+] 08th train: loss: 0.082941025       accuracy: 0.9770
[+] 09th train: loss: 0.092324398       accuracy: 0.9756
[+] 10th train: loss: 0.096432678       accuracy: 0.9784
[+] 11th train: loss: 0.111147106       accuracy: 0.9760
[+] 12th train: loss: 0.109806009       accuracy: 0.9768
[+] 13th train: loss: 0.110407412       accuracy: 0.9764
[+] 14th train: loss: 0.112699881       accuracy: 0.9756
[+] 15th train: loss: 0.110865794       accuracy: 0.9754
[+] 16th train: loss: 0.112595461       accuracy: 0.9772
[+] 17th train: loss: 0.134697363       accuracy: 0.9720
[+] 18th train: loss: 0.143008068       accuracy: 0.9746
[+] 19th train: loss: 0.117918320       accuracy: 0.9768
[+] 20th train: loss: 0.139959618       accuracy: 0.9772
[+] Train finished successfully. It takes: 11.34s
[*] Accuracy of test : 0.9772
[*] Accuracy of validation : 0.9782
```

多层隐藏层其实是对输入特征多层次的抽象，最终的目的就是为了更好的线性划分不同类型的数据，但是层数越多，参数也会爆炸式地增长。所以，达到一定的层数后再加深隐藏层，准确率提升会越来越不明显。

3）不同训练参数（batch size、epoch num、学习率）对准确率的影响和分析（10 分）

①改变 batch size

Batch size = 50

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.147734404       accuracy: 0.9568
[+] 02th train: loss: 0.096720994       accuracy: 0.9698
[+] 03th train: loss: 0.093080029       accuracy: 0.9698
[+] 04th train: loss: 0.086363815       accuracy: 0.9730
[+] 05th train: loss: 0.080658332       accuracy: 0.9766
[+] 06th train: loss: 0.088234328       accuracy: 0.9758
[+] 07th train: loss: 0.080055803       accuracy: 0.9782
[+] 08th train: loss: 0.091958039       accuracy: 0.9772
[+] 09th train: loss: 0.091786154       accuracy: 0.9774
[+] 10th train: loss: 0.091970243       accuracy: 0.9778
[+] 11th train: loss: 0.090219527       accuracy: 0.9792
[+] 12th train: loss: 0.119705282       accuracy: 0.9742
[+] 13th train: loss: 0.102605715       accuracy: 0.9790
```

```
[+] 14th train: loss: 0.099555887       accuracy: 0.9800
[+] 15th train: loss: 0.108803369       accuracy: 0.9794
[+] 16th train: loss: 0.111768253       accuracy: 0.9774
[+] 17th train: loss: 0.111067675       accuracy: 0.9788
[+] 18th train: loss: 0.156365022       accuracy: 0.9730
[+] 19th train: loss: 0.114212058       accuracy: 0.9818
[+] 20th train: loss: 0.120656751       accuracy: 0.9790
[+] Train finished successfully. It takes: 17.05s
[*] Accuracy of test : 0.979
[*] Accuracy of validation : 0.9802
```

Batch size = 100

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.183157325       accuracy: 0.9450
[+] 02th train: loss: 0.111994974       accuracy: 0.9682
[+] 03th train: loss: 0.095591098       accuracy: 0.9708
[+] 04th train: loss: 0.106001489       accuracy: 0.9678
[+] 05th train: loss: 0.088448085       accuracy: 0.9724
[+] 06th train: loss: 0.084663555       accuracy: 0.9750
[+] 07th train: loss: 0.086517282       accuracy: 0.9758
[+] 08th train: loss: 0.093787283       accuracy: 0.9760
[+] 09th train: loss: 0.106040932       accuracy: 0.9728
[+] 10th train: loss: 0.124335743       accuracy: 0.9710
[+] 11th train: loss: 0.096565083       accuracy: 0.9758
[+] 12th train: loss: 0.103044972       accuracy: 0.9762
[+] 13th train: loss: 0.110727511       accuracy: 0.9738
[+] 14th train: loss: 0.102714039       accuracy: 0.9784
[+] 15th train: loss: 0.135215625       accuracy: 0.9704
[+] 16th train: loss: 0.112099901       accuracy: 0.9790
[+] 17th train: loss: 0.108759128       accuracy: 0.9778
[+] 18th train: loss: 0.114044607       accuracy: 0.9798
[+] 19th train: loss: 0.116589747       accuracy: 0.9756
[+] 20th train: loss: 0.114664920       accuracy: 0.9790
[+] Train finished successfully. It takes: 11.47s
[*] Accuracy of test : 0.979
[*] Accuracy of validation : 0.9804
```

增大 batch size，所花费的时间会减少

②改变 epoch num

Epoch num = 20

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.183157325       accuracy: 0.9450
[+] 02th train: loss: 0.111994974       accuracy: 0.9682
[+] 03th train: loss: 0.095591098       accuracy: 0.9708
[+] 04th train: loss: 0.106001489       accuracy: 0.9678
[+] 05th train: loss: 0.088448085       accuracy: 0.9724
[+] 06th train: loss: 0.084663555       accuracy: 0.9750
[+] 07th train: loss: 0.086517282       accuracy: 0.9758
[+] 08th train: loss: 0.093787283       accuracy: 0.9760
[+] 09th train: loss: 0.106040932       accuracy: 0.9728
[+] 10th train: loss: 0.124335743       accuracy: 0.9710
[+] 11th train: loss: 0.096565083       accuracy: 0.9758
[+] 12th train: loss: 0.103044972       accuracy: 0.9762
[+] 13th train: loss: 0.110727511       accuracy: 0.9738
[+] 14th train: loss: 0.102714039       accuracy: 0.9784
[+] 15th train: loss: 0.135215625       accuracy: 0.9704
[+] 16th train: loss: 0.112099901       accuracy: 0.9790
[+] 17th train: loss: 0.108759128       accuracy: 0.9778
[+] 18th train: loss: 0.114044607       accuracy: 0.9798
[+] 19th train: loss: 0.116589747       accuracy: 0.9756
[+] 20th train: loss: 0.114664920       accuracy: 0.9790
[+] Train finished successfully. It takes: 11.47s
[*] Accuracy of test : 0.979
[*] Accuracy of validation : 0.9804
```

Epoch num = 40

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.162154600       accuracy: 0.9546
[+] 02th train: loss: 0.105730198       accuracy: 0.9702
[+] 03th train: loss: 0.100496486       accuracy: 0.9704
[+] 04th train: loss: 0.080202639       accuracy: 0.9746
[+] 05th train: loss: 0.077481717       accuracy: 0.9774
[+] 06th train: loss: 0.080998473       accuracy: 0.9768
[+] 07th train: loss: 0.082480319       accuracy: 0.9780
[+] 08th train: loss: 0.094995715       accuracy: 0.9736
[+] 09th train: loss: 0.086363889       accuracy: 0.9774
[+] 10th train: loss: 0.090298057       accuracy: 0.9780
[+] 11th train: loss: 0.095298961       accuracy: 0.9752
[+] 12th train: loss: 0.104132816       accuracy: 0.9762
[+] 13th train: loss: 0.097881615       accuracy: 0.9784
[+] 14th train: loss: 0.097069502       accuracy: 0.9770
```

```
[+] 15th train: loss: 0.089636825       accuracy: 0.9784
[+] 16th train: loss: 0.102563694       accuracy: 0.9778
[+] 17th train: loss: 0.124738842       accuracy: 0.9752
[+] 18th train: loss: 0.106141478       accuracy: 0.9778
[+] 19th train: loss: 0.111032069       accuracy: 0.9782
[+] 20th train: loss: 0.110552527       accuracy: 0.9770
[+] 21th train: loss: 0.125871599       accuracy: 0.9788
[+] 22th train: loss: 0.116042383       accuracy: 0.9794
[+] 23th train: loss: 0.118711546       accuracy: 0.9754
[+] 24th train: loss: 0.118426345       accuracy: 0.9786
[+] 25th train: loss: 0.130635515       accuracy: 0.9776
[+] 26th train: loss: 0.147374541       accuracy: 0.9740
[+] 27th train: loss: 0.113732800       accuracy: 0.9804
[+] 28th train: loss: 0.118071474       accuracy: 0.9778
[+] 29th train: loss: 0.114254557       accuracy: 0.9802
[+] 30th train: loss: 0.115076609       accuracy: 0.9792
[+] 31th train: loss: 0.152130559       accuracy: 0.9746
[+] 32th train: loss: 0.133291498       accuracy: 0.9780
[+] 33th train: loss: 0.110589914       accuracy: 0.9812
[+] 34th train: loss: 0.124287888       accuracy: 0.9794
[+] 35th train: loss: 0.141428679       accuracy: 0.9788
[+] 36th train: loss: 0.144603118       accuracy: 0.9764
[+] 37th train: loss: 0.130253091       accuracy: 0.9786
[+] 38th train: loss: 0.120986648       accuracy: 0.9806
[+] 39th train: loss: 0.118629612       accuracy: 0.9812
[+] 40th train: loss: 0.117276527       accuracy: 0.9820
[+] Train finished successfully. It takes: 23.10s
[*] Accuracy of test : 0.982
[*] Accuracy of validation : 0.9814
```

花费的时间更长了，测试集的准确率看起来提高了。

③改变 learning rate

Learning rate = 0.01

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.139196724       accuracy: 0.9612
[+] 02th train: loss: 0.127321258       accuracy: 0.9650
[+] 03th train: loss: 0.133640423       accuracy: 0.9668
[+] 04th train: loss: 0.127516404       accuracy: 0.9678
[+] 05th train: loss: 0.152415067       accuracy: 0.9634
[+] 06th train: loss: 0.203116789       accuracy: 0.9568
```

```
[+] 07th train: loss: 0.137514606       accuracy: 0.9722
[+] 08th train: loss: 0.159833863       accuracy: 0.9670
[+] 09th train: loss: 0.160907790       accuracy: 0.9712
[+] 10th train: loss: 0.200503886       accuracy: 0.9638
[+] 11th train: loss: 0.154320657       accuracy: 0.9750
[+] 12th train: loss: 0.164581984       accuracy: 0.9754
[+] 13th train: loss: 0.208559737       accuracy: 0.9638
[+] 14th train: loss: 0.196656182       accuracy: 0.9700
[+] 15th train: loss: 0.199008808       accuracy: 0.9720
[+] 16th train: loss: 0.222129688       accuracy: 0.9684
[+] 17th train: loss: 0.231966749       accuracy: 0.9698
[+] 18th train: loss: 0.221545219       accuracy: 0.9684
[+] 19th train: loss: 0.203758523       accuracy: 0.9744
[+] 20th train: loss: 0.247628435       accuracy: 0.9702
[+] Train finished successfully. It takes: 11.93s
[*] Accuracy of test : 0.9702
[*] Accuracy of validation : 0.9694
```

Learning rate = 0.001

```
简单划分前 50000 个为训练集，后 5000 个为测试集，对其进行训练，并使用验证集评估模型
[+] 01th train: loss: 0.136891678       accuracy: 0.9602
[+] 02th train: loss: 0.105703019       accuracy: 0.9682
[+] 03th train: loss: 0.083157286       accuracy: 0.9748
[+] 04th train: loss: 0.075365245       accuracy: 0.9776
[+] 05th train: loss: 0.081722938       accuracy: 0.9766
[+] 06th train: loss: 0.071631521       accuracy: 0.9806
[+] 07th train: loss: 0.068727762       accuracy: 0.9792
[+] 08th train: loss: 0.082863927       accuracy: 0.9770
[+] 09th train: loss: 0.080244780       accuracy: 0.9792
[+] 10th train: loss: 0.071700573       accuracy: 0.9808
[+] 11th train: loss: 0.083797976       accuracy: 0.9776
[+] 12th train: loss: 0.093615651       accuracy: 0.9766
[+] 13th train: loss: 0.106170505       accuracy: 0.9756
[+] 14th train: loss: 0.093926676       accuracy: 0.9772
[+] 15th train: loss: 0.096539006       accuracy: 0.9806
[+] 16th train: loss: 0.105295949       accuracy: 0.9788
[+] 17th train: loss: 0.131663233       accuracy: 0.9724
[+] 18th train: loss: 0.098230995       accuracy: 0.9804
[+] 19th train: loss: 0.099405102       accuracy: 0.9828
[+] 20th train: loss: 0.118864186       accuracy: 0.9794
[+] Train finished successfully. It takes: 11.39s
```

```
[*] Accuracy of test : 0.9794
[*] Accuracy of validation : 0.9796
```

减小 learning rate，准确率将有所提高

但是若学习率太大，易损失函数爆炸，易震荡；若学习率太小，容易过拟合，收敛速度减慢。

4）留出法不同比例对结果的影响和分析（10 分）

train_percentage=0.8 时

```
使用分层采样的留出法训练、测试模型，并使用验证集评估模型
划分比例为 80.0%
[+] 01th train: loss: 0.194731444       accuracy: 0.9427
[+] 02th train: loss: 0.134377033       accuracy: 0.9585
[+] 03th train: loss: 0.108549081       accuracy: 0.9666
[+] 04th train: loss: 0.100686878       accuracy: 0.9673
[+] 05th train: loss: 0.104537018       accuracy: 0.9688
[+] 06th train: loss: 0.101801746       accuracy: 0.9693
[+] 07th train: loss: 0.097825304       accuracy: 0.9728
[+] 08th train: loss: 0.096836671       accuracy: 0.9731
[+] 09th train: loss: 0.114498354       accuracy: 0.9702
[+] 10th train: loss: 0.107484631       accuracy: 0.9745
[+] 11th train: loss: 0.117018022       accuracy: 0.9732
[+] 12th train: loss: 0.125605717       accuracy: 0.9711
[+] 13th train: loss: 0.117100142       accuracy: 0.9734
[+] 14th train: loss: 0.119585291       accuracy: 0.9728
[+] 15th train: loss: 0.117334031       accuracy: 0.9763
[+] 16th train: loss: 0.137848094       accuracy: 0.9708
[+] 17th train: loss: 0.137966946       accuracy: 0.9718
[+] 18th train: loss: 0.125369906       accuracy: 0.9735
[+] 19th train: loss: 0.137740120       accuracy: 0.9745
[+] 20th train: loss: 0.135814607       accuracy: 0.9738
[+] Train finished successfully. It takes: 10.62s
[*] Accuracy of test : 0.9738182
[*] Accuracy of validation : 0.9794
```

train_percentage=0.5 时

```
使用分层采样的留出法训练、测试模型，并使用验证集评估模型
划分比例为 50.0%
[+] 01th train: loss: 0.217452988       accuracy: 0.9375
```

```
[+] 02th train: loss: 0.153062180       accuracy: 0.9567
[+] 03th train: loss: 0.130004302       accuracy: 0.9603
[+] 04th train: loss: 0.132151902       accuracy: 0.9590
[+] 05th train: loss: 0.102736525       accuracy: 0.9699
[+] 06th train: loss: 0.097638853       accuracy: 0.9714
[+] 07th train: loss: 0.111962341       accuracy: 0.9671
[+] 08th train: loss: 0.102141723       accuracy: 0.9715
[+] 09th train: loss: 0.102494285       accuracy: 0.9724
[+] 10th train: loss: 0.103992559       accuracy: 0.9731
[+] 11th train: loss: 0.113913514       accuracy: 0.9716
[+] 12th train: loss: 0.107623227       accuracy: 0.9729
[+] 13th train: loss: 0.117835373       accuracy: 0.9721
[+] 14th train: loss: 0.114233121       accuracy: 0.9728
[+] 15th train: loss: 0.117222831       accuracy: 0.9731
[+] 16th train: loss: 0.129818663       accuracy: 0.9706
[+] 17th train: loss: 0.133694217       accuracy: 0.9696
[+] 18th train: loss: 0.119176872       accuracy: 0.9729
[+] 19th train: loss: 0.130604088       accuracy: 0.9728
[+] 20th train: loss: 0.115728803       accuracy: 0.9756
[+] Train finished successfully. It takes: 7.18s
[*] Accuracy of test : 0.97563636
[*] Accuracy of validation : 0.9764
```

train_percentage=0.1 时

```
使用分层采样的留出法训练、测试模型，并使用验证集评估模型
划分比例为 10.0%
[+] 01th train: loss: 0.411667228       accuracy: 0.8846
[+] 02th train: loss: 0.321537048       accuracy: 0.9050
[+] 03th train: loss: 0.269970328       accuracy: 0.9202
[+] 04th train: loss: 0.244475141       accuracy: 0.9277
[+] 05th train: loss: 0.247159362       accuracy: 0.9253
[+] 06th train: loss: 0.219042018       accuracy: 0.9353
[+] 07th train: loss: 0.219790086       accuracy: 0.9359
[+] 08th train: loss: 0.212351233       accuracy: 0.9392
[+] 09th train: loss: 0.202130243       accuracy: 0.9425
[+] 10th train: loss: 0.208860680       accuracy: 0.9411
[+] 11th train: loss: 0.201098636       accuracy: 0.9433
[+] 12th train: loss: 0.205682650       accuracy: 0.9431
[+] 13th train: loss: 0.206574097       accuracy: 0.9442
[+] 14th train: loss: 0.210223392       accuracy: 0.9445
[+] 15th train: loss: 0.215024814       accuracy: 0.9441
```

```
[+] 16th train: loss: 0.213510126       accuracy: 0.9458
[+] 17th train: loss: 0.215828881       accuracy: 0.9455
[+] 18th train: loss: 0.216512725       accuracy: 0.9462
[+] 19th train: loss: 0.223397180       accuracy: 0.9454
[+] 20th train: loss: 0.219470903       accuracy: 0.9464
[+] Train finished successfully. It takes: 2.91s
[*] Accuracy of test : 0.94638383
[*] Accuracy of validation : 0.952
```

减小 train_percentage，训练集会变小，准确率下降了，训练出的参数对于验证集会不太适应。

6）k 折交叉验证法不同 k 值对结果的影响和分析（10 分）

k=5 时：

```
[+] 18th train: loss: 0.022649713       accuracy: 0.9976
[+] 19th train: loss: 0.025593059       accuracy: 0.9960
[+] 20th train: loss: 0.036319405       accuracy: 0.9935
[+] Train finished successfully. It takes: 10.48s
[*] Temp accuracy of test : 0.99352753
[*] Temp accuracy of validation : 0.9794
[-] k = 5，当前第 5 组为测试集
[+] 01th train: loss: 0.112664305       accuracy: 0.9717
[+] 02th train: loss: 0.103968032       accuracy: 0.9717
[+] 03th train: loss: 0.066095226       accuracy: 0.9814
[+] 04th train: loss: 0.038035966       accuracy: 0.9862
[+] 05th train: loss: 0.042218033       accuracy: 0.9879
[+] 06th train: loss: 0.046070859       accuracy: 0.9871
[+] 07th train: loss: 0.068847828       accuracy: 0.9846
[+] 08th train: loss: 0.060095858       accuracy: 0.9871
[+] 09th train: loss: 0.035809826       accuracy: 0.9927
[+] 10th train: loss: 0.066647008       accuracy: 0.9846
[+] 11th train: loss: 0.104547471       accuracy: 0.9733
[+] 12th train: loss: 0.062989727       accuracy: 0.9838
[+] 13th train: loss: 0.114801154       accuracy: 0.9717
[+] 14th train: loss: 0.063874193       accuracy: 0.9838
[+] 15th train: loss: 0.060858037       accuracy: 0.9871
[+] 16th train: loss: 0.084816165       accuracy: 0.9830
[+] 17th train: loss: 0.089202777       accuracy: 0.9773
[+] 18th train: loss: 0.075645283       accuracy: 0.9830
[+] 19th train: loss: 0.089022890       accuracy: 0.9822
[+] 20th train: loss: 0.091392644       accuracy: 0.9838
```

```
[+] Train finished successfully. It takes: 10.30s
[*] Temp accuracy of test : 0.98381877
[*] Temp accuracy of validation : 0.9754
[*] Average accuracy of test : 0.9857335329055786
[*] Average accuracy of validation : 0.9792399883270264
```

k=10 时：

```
[+] 18th train: loss: 0.023518218       accuracy: 0.9951
[+] 19th train: loss: 0.026792917       accuracy: 0.9903
[+] 20th train: loss: 0.017160028       accuracy: 0.9951
[+] Train finished successfully. It takes: 10.55s
[*] Temp accuracy of test : 0.9951456
[*] Temp accuracy of validation : 0.9778
[-] k = 10，当前第 10 组为测试集
[+] 01th train: loss: 0.085972734       accuracy: 0.9838
[+] 02th train: loss: 0.062644206       accuracy: 0.9871
[+] 03th train: loss: 0.109430790       accuracy: 0.9725
[+] 04th train: loss: 0.087105028       accuracy: 0.9773
[+] 05th train: loss: 0.068091646       accuracy: 0.9822
[+] 06th train: loss: 0.075597949       accuracy: 0.9806
[+] 07th train: loss: 0.035453826       accuracy: 0.9871
[+] 08th train: loss: 0.033834849       accuracy: 0.9903
[+] 09th train: loss: 0.056794282       accuracy: 0.9838
[+] 10th train: loss: 0.032748748       accuracy: 0.9919
[+] 11th train: loss: 0.018768238       accuracy: 0.9935
[+] 12th train: loss: 0.033625629       accuracy: 0.9935
[+] 13th train: loss: 0.041266926       accuracy: 0.9887
[+] 14th train: loss: 0.056842551       accuracy: 0.9854
[+] 15th train: loss: 0.048712485       accuracy: 0.9887
[+] 16th train: loss: 0.077967301       accuracy: 0.9806
[+] 17th train: loss: 0.039274741       accuracy: 0.9887
[+] 18th train: loss: 0.098558523       accuracy: 0.9773
[+] 19th train: loss: 0.043576218       accuracy: 0.9887
[+] 20th train: loss: 0.016460277       accuracy: 0.9935
[+] Train finished successfully. It takes: 10.41s
[*] Temp accuracy of test : 0.99352753
[*] Temp accuracy of validation : 0.9798
[*] Average accuracy of test : 0.9866730213165283
[*] Average accuracy of validation : 0.97985999584198
```

k=20 时：

```
[+] 18th train: loss: 0.036207333       accuracy: 0.9968
[+] 19th train: loss: 0.037735254       accuracy: 0.9935
[+] 20th train: loss: 0.063786730       accuracy: 0.9935
[+] Train finished successfully. It takes: 10.62s
[*] Temp accuracy of test : 0.99352753
[*] Temp accuracy of validation : 0.981
[-] k = 20，当前第 20 组为测试集
[+] 01th train: loss: 0.208185658       accuracy: 0.9450
[+] 02th train: loss: 0.164062470       accuracy: 0.9612
[+] 03th train: loss: 0.078316592       accuracy: 0.9773
[+] 04th train: loss: 0.106838055       accuracy: 0.9676
[+] 05th train: loss: 0.106989630       accuracy: 0.9612
[+] 06th train: loss: 0.067302033       accuracy: 0.9709
[+] 07th train: loss: 0.149658576       accuracy: 0.9644
[+] 08th train: loss: 0.173581362       accuracy: 0.9612
[+] 09th train: loss: 0.062553443       accuracy: 0.9838
[+] 10th train: loss: 0.081178918       accuracy: 0.9676
[+] 11th train: loss: 0.171117410       accuracy: 0.9579
[+] 12th train: loss: 0.110403672       accuracy: 0.9709
[+] 13th train: loss: 0.182118550       accuracy: 0.9612
[+] 14th train: loss: 0.140303746       accuracy: 0.9773
[+] 15th train: loss: 0.082665317       accuracy: 0.9838
[+] 16th train: loss: 0.118665032       accuracy: 0.9741
[+] 17th train: loss: 0.171076789       accuracy: 0.9676
[+] 18th train: loss: 0.128182411       accuracy: 0.9676
[+] 19th train: loss: 0.120584756       accuracy: 0.9773
[+] 20th train: loss: 0.167755201       accuracy: 0.9644
[+] Train finished successfully. It takes: 10.51s
[*] Temp accuracy of test : 0.9644013
[*] Temp accuracy of validation : 0.9804
[*] Average accuracy of test : 0.9861039310693741
[*] Average accuracy of validation : 0.9804799973964691
```

k 值越大，准确率上升，训练出的模型会更加准确、稳定，但耗时也会更久。

五、实验总结及心得

1. 本次实验学习到了机器学习中模型的训练和应用，掌握了分层采样的留出法和 k 折交叉验证法的编写，通过编写代码以及代码调试，提高了自己写代

码的能力。

2. 通过本次实验，学习到了不同的超参数对于模型训练和应用，对于不是超参数的参数，只能通过训练优化。为了让模型的准确率达到我们的要求，需要不断地调整超参数的设置来达到我们要求的准确度。

3. 代码：https://github.com/XDUgaile/Mechine_Learning

4. 在配置本次试验环境时由于有些python包不存在，代码运行时报错等问题，参考了网上的一些资料：

https://blog.csdn.net/qq_43060552/article/details/103189040

https://stackoverflow.com/questions/37383812/tensorflow-module-object-has-no-attribute-placeholder