人工智能实验报告

钱兴宇
人工智能 2101 班 8207211912

2023 年 4 月

摘要

本学期的人工智能实验采用了 ModelArts 和 MindSpore 开发框架，主要关注机器学习、神经网络以及深度学习。该实验报告分为三部分，分别是回归分析、手写数字识别和乳腺癌识别。

在回归分析部分，本文对新冠肺炎全国和地区病例数据进行了分析，并建立了合适的回归模型来预测疾病的发展趋势。此外，也对前列腺癌数据进行了回归分析，并对指定测试集进行了测试，得到了 R2，MSE 指标。

在手写数字识别部分，本文使用了 ModelArts 上的 Minist 手写数字集，采用 AI 开发框架 Mindspore 进行训练和测试，并对平方差损失和交叉熵损失、Mini-batch 和 no batch、Relu 和 Sigmoid、有 dropout 和无 dropout 等进行了对比实验。此外，本文将该模型稍加修改，迁移到美国邮政编码手写数字集的数字识别上，并进行了改进。还试图使用前期遗传算法生成的手写数字图像进行测试，并观察识别效果如何。

在乳腺癌识别部分，本文对不同级别的任务进行了不同的评估，包括图像级别的分类任务、肿瘤检测任务以及肿瘤分割任务。评价指标包括每类数据的 F1 值、分类的 F1 和检测框的 mIOU 以及分类的 F1 和 MASK 的 mIOU。

通过实验的进行，笔者不仅熟练掌握了 ModelArts 和 MindSpore 开发框架的使用方法，也学习到了如何选择适合的算法和模型以及如何针对实际问题进行改进。在未来的实践中，可以进一步针对不同领域的数据集进行更深入的实验，并探索更多机器学习和深度学习的应用场景。

目录

Chapter 1

实验环境

华为云是华为推出的云服务平台，提供了计算、存储、网络、安全等各种云服务，包括弹性云服务器、云存储、云数据库、人工智能服务等。华为云致力于成为全球领先的数字化基础设施服务商，为客户提供全球一体化的云服务。

MindSpore 是华为公司开源的全场景深度学习框架，支持 CPU、GPU 和🄵腾 AI 处理器算力，提供了易用、高效的开发体验，支持端到端的自动微分。该框架可以应用于计算机视觉、自然语言处理、推荐系统等多个领域，为数据科学家和算法工程师提供了一个强大的工具。通过与华为🄵腾系列 AI 处理器的结合使用，MindSpore 可以实现更加高效的深度学习处理，为人工智能技术在各个行业中的广泛应用提供了支持。

本次实验基于华为云平台，使用 ModelArts 和 MindSpore 开发框架，使用 python 进行实验开发。具体信息见表 1.1、表 1.2。

表 1.1: 软件、硬件信息

硬件	规格
GPU	1 * T4 (16GB)
CPU	8 vCPUs, 32GB
软件	**版本**
MindSpore	1.7.0
CUDA	10.1
Python	3.7
Operating System	Ubuntu 18.04

表 1.2: 主要依赖包及版本

Package	Version
APScheduler	3.8.1
arrow	1.2.3
ipython	7.34.0
matplotlib	3.5.2
matplotlib-inlie	0.1.6
mock	4.0.3
modelarts	1.4.118
numpy	1.21.6
opencv-python-headless	4.5.60
pandas	1.3.5
pickleshare	0.7.5
Pillow	9.3.0
pyparser	2.21
scipy	1.7.3
scikit-learn	1.2.2
zipp	3.15.0

Chapter 2

回归分析

2.1 新冠肺炎全国病例回归分析

2.1.1 实验内容

分析新冠肺炎全国和地区病例数据，利用 ModelArts 平台上开发环境中的 Notebook 自编代码，建立合适的回归模型进行回归分析，建立疾病发展情况的趋势模型，并对指定日期之后的数据进行预测。本实验决定利用所给数据对全国新冠肺炎病例进行回归分析。

2.1.2 数据处理

本实验选择 DXYArea.csv 进行实验，该文件有 87392 行，11 列。

首先选取有效特征，经过查阅文献，结合实验指导书，本实验初步选择了 countryName,provinceName,province_comfirmedCount province_curedCount, province_deadCount,updateTime 列。考虑到 updateTime 跨度比较大，精确到时分秒是不必要的，可以将其转换为年月日。

```
# 筛选所需的列
df = df[['countryName', 'provinceName', 'province_confirmedCount',
'province_suspectedCount', 'province_curedCount',
'province_deadCount', 'updateTime']]

```

```
time_format = '%Y-%m-%d %H:%M:%S'  # 定义原始时间格式
new_format = '%Y-%m-%d'  # 定义新的时间格式
new_time_col = []  # 新的时间列

for t_str in df['updateTime']:
    # 将字符串转换成 datetime 对象
    t = datetime.datetime.strptime(t_str, time_format)
    new_time_str = datetime.datetime.strftime(t, new_format)
    new_time_col.append(new_time_str)

df['new_updateTime'] = new_time_col
```

本实验对中国数据分析，因此保留 countryName 为中国的数据，保留后国名列可以去掉。观察原数据发现 provuince_suspectedCount 绝大多数均为 0，可以去掉。还需要去掉时间和省份相同的重复列。观察到有一个 provinceName 为中国的数据，应该是同一天内全国的数据，在这里显然应该去掉。最后把每个省份的数据按照时间求和，得到全国一天的数据，保存在最终版本的 DXYArea.csv 中。

```
# 保留中国的数据
df = df[df['countryName'] == '中国']
# 去掉国名、详细更新时间的列
df.drop("countryName", axis=1, inplace=True)
df.drop("updateTime", axis=1, inplace=True)
df.drop("province_suspectedCount", axis=1, inplace=True)
# 删除重复数据
df.drop_duplicates(subset=['new_updateTime', 'provinceName'])
# 去掉省名为“中国”的行数据
df = df[df['provinceName'] != '中国']
df.drop("provinceName", axis=1,
inplace=True)
# 按 new_updateTime 列分组
grouped = df.groupby("new_updateTime")
# 将每个分组进行求和操作
new_data = grouped.sum()
# 保存新的 CSV 文件
new_data.to_csv(" 最终版本的 DXYArea.csv")
```

最终得到的数据表格形状为 60x4。前三列为特征，第四列为时间。表 2.1 为实验所用部分数据。图 2.1 为前三列的小提琴图。

表 2.1: 实验部分数据

nupdateTime	confirmedCount	curedCount	deadCount
2020/1/22	544	28	17
2020/1/23	181	2	0
2020/1/24	901	36	26

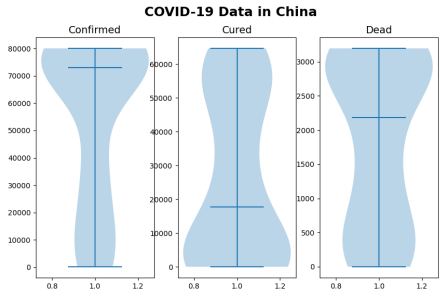

图 2.1: 新冠肺炎病例数据小提琴图

通过数据散点图可以直观了解数据的分布情况，从而判断采用何种拟合方法进行拟合。下面绘制确诊人数、治愈人数、死亡人数散点图。

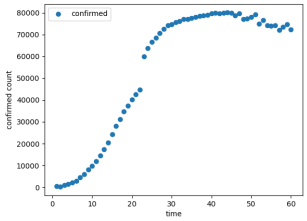

图 2.2: 确诊人数

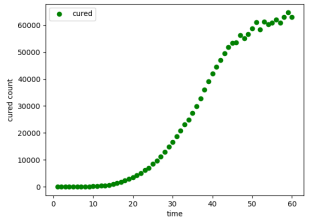

图 2.3: 治愈人数

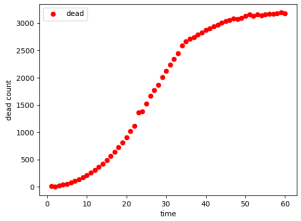

图 2.4: 死亡人数

2.1.3 模型介绍

通过观察上述数据散点图，本实验最终选择使用多项式函数来拟合数据，并给出后十天的预测值。

多项式阶数是可以选择的超参数，本实验考虑了阶数 3、4、5 三种情况。选用 3 阶多项式拟合时拟合效果不佳，选用 5 阶多项式拟合时在边界点预测会出现急剧震荡，因此本实验最终选择 4 阶多项式对实验数据进行拟合。

```
# 定义多项式函数及其阶数

poly_degree = 4

def poly_func(x, *coef):
    result = 0
    for i, c in enumerate(coef):
        result += c * x ** i
    return result
```

2.1.4 实验结果

本实验利用 sklearn 库，用一个四次多项式对上述数据进行拟合，并且对接下来十天进行预测。最后计算 MSE 和 R^2. 具体拟合指标见表 2.2，拟合图像见图 2.5、图 2.6、图 2.7.

表 2.2: 拟合指标

拟合数据	MSE	R^2
confirmedCount	71928.02	0.99
curedCount	21581.31	1.00
deadCount	3959.71	1.00

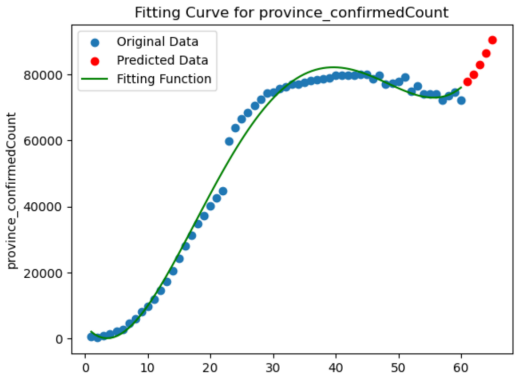

图 2.5: 确诊数据拟合结果

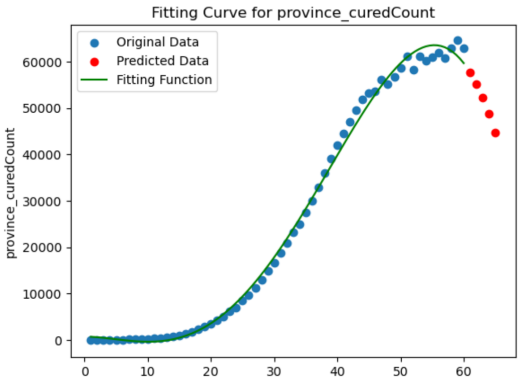

图 2.6: 治疗数据拟合结果

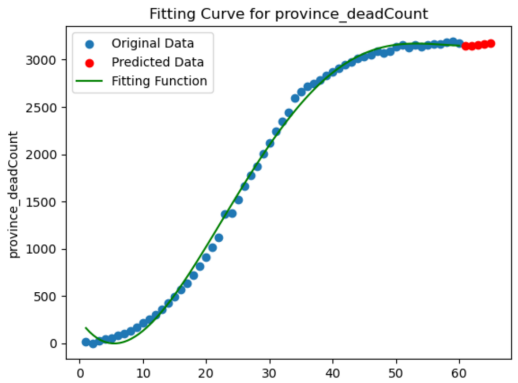

图 2.7: 死亡数据拟合结果

2.1.5 实验改进

从实验结果预测数据可以观察到多项式拟合确诊人数、治疗人数在边界处出现了震荡现象，显然不符合实际情况。在多项式回归的基础上，再加上 Ridge 回归（岭回归）。Ridge 回归数学模型如公式 2.1 所示。

$$\text{minimize} \|Y - X\beta\|_2^2 + \alpha\|\beta\|_2^2, \tag{2.1}$$

通过对回归系数引入 L2 正则化来避免过拟合问题，在普通最小二乘法的基础上加入了正则化项。可以通过调节 alpha 的值来调整正则化项对模型影响，alpha 值越小，则干预越小，模型就更复杂，alpha 值越大，则干预大，模型更简单。

观察图 2.8、图 2.9、图 2.10，可以看到引入岭回归对确诊数据影响最为明显，对治疗数据、死亡数据影响较小，这可能是三者曲线在区域边界趋势不同导致。对比表 2.2、表 2.3，可以看到引入岭回归后 MSE 和 R^2 两个指标表现有所下降。因此在选择 alpha 时需要综合考虑原有数据、预测数据拟合效果，选择一个较好的平衡点。

```
# 进行多项式函数回归并加入岭回归正则化

alpha = 50

model = make_pipeline(PolynomialFeatures(poly_degree,
 include_bias=False), Ridge(alpha=alpha))
model.fit(x.reshape(-1,1), y)
```

表 2.3: 拟合指标

拟合数据	MSE	R^2
confirmedCount	132102.21	0.98
curedCount	59336.37	0.99
deadCount	3959.71	1.00

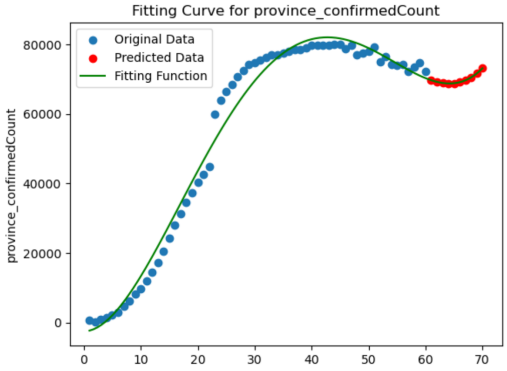

图 2.8: 岭回归确诊数据拟合结果

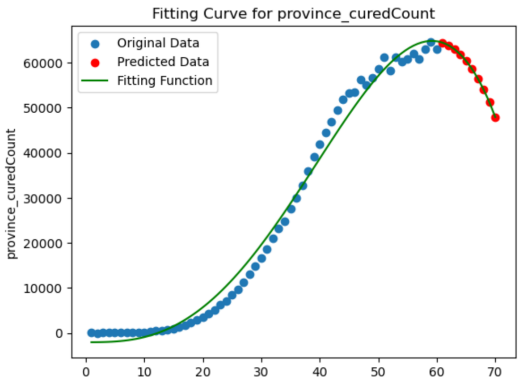

图 2.9: 岭回归治疗数据拟合结果

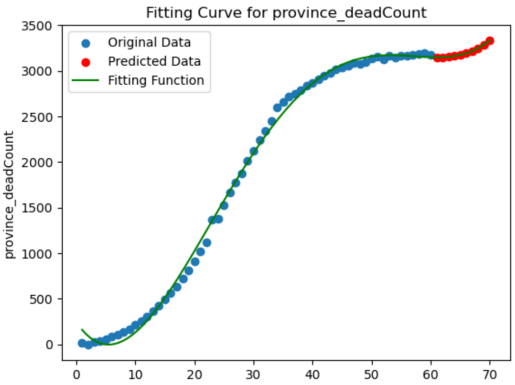

图 2.10: 岭回归死亡数据拟合结果

2.2 前列腺癌回归分析

2.2.1 实验内容

前列腺癌预测，利用 ModelArts 平台上开发环境中的 Notebook 自编代码，对前列腺癌数据进行回归分析，该数据集已分出训练和测试样例，请按统一的测试集进行测试。

2.2.2 数据处理

本实验选择 prostate.data 进行实验，一共有 97 行,10 列，分别为 lcavol，lweight，age,lbph,svi,lcp,gleason,pgg45,lpsa,train. 前 8 列为回归分析的特征列，第 9 列为回归分析的目标值列，第 10 列用来区分是否为测试集。

先把文件以 csv 文件形式储存，然后提取所需要的特征列。由于不同特征之间的数据量纲不同，数据范围也存在较大差距，直接进行回归分析可能会影响回归系数的解释和分析。因此需要对不同特征进行标准化，如式 2.2 所示。

$$Z = \frac{X - \mu}{\sigma} \tag{2.2}$$

```
# 复制并删除名为 'column_name' 的列
data = data.drop('Unnamed: 0', axis=1)
# 提取需要缩放的特征列
features = ['lcavol', 'lweight', 'age', 'lbph', 'svi',
    'lcp', 'gleason', 'pgg45']
x = data[features].values

# 计算特征的均值和标准差
x_mean = x.mean(axis=0)
x_std = x.std(axis=0)
# 对特征进行缩放
xp = (x - x_mean) / x_std

# 将缩放后的特征保存到 DataFrame 中
data[features] = xp
```

表 2.4 给出了数据集中前三列数据，图 2.11 给出了整个数据特征列的小提琴图，可以帮助我们进一步掌握数据的基本情况。

表 2.4: 实验部分数据

lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
-1.65	-2.02	-1.87	-1.03	-0.53	-0.87	-1.05	-0.87	-0.43
-2.00	-0.73	-0.79	-1.03	-0.53	-0.87	-1.05	-0.87	-0.16
-1.59	-2.20	1.37	-1.03	-0.53	-0.87	0.34	-0.16	-0.16

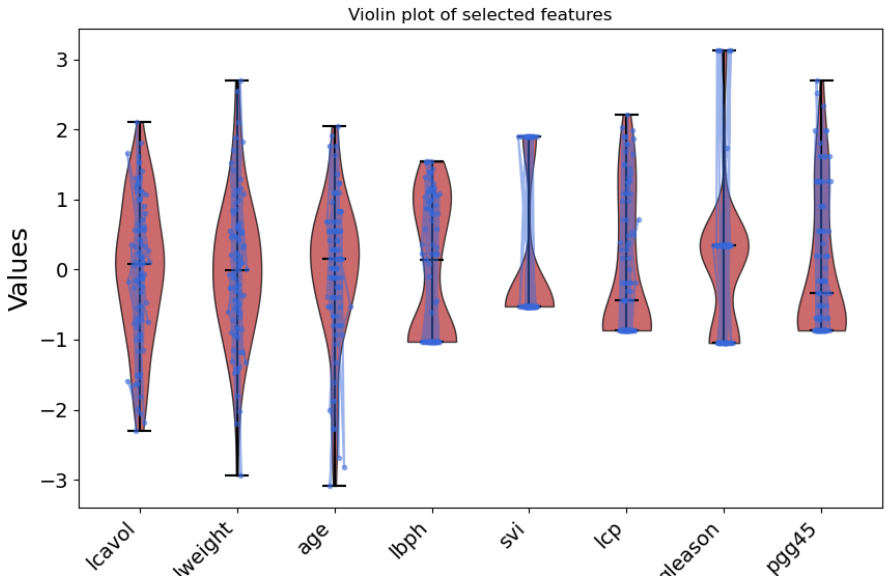

图 2.11: 前列腺癌数据小提琴图

最后我们需要根据 train 列的值划分训练集、测试集，若值为 T 则为训练集，否则为测试集。

```
# 提取 train 列中值为 T 的行
train_data = data[data['train'] == 'T']
# 提取 train 列中值为 F 的行
train_data = data[data['train'] == 'F']
```

2.2.3 模型介绍

本实验选择多元线性回归模型来拟合训练数据，见式 2.3。

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \tag{2.3}$$

本实验选择 sklearn 库，使用线性回归模型对测试集数据进行拟合。

```
# 提取特征和目标变量
x_train = train_data[['lcavol', 'lweight',
    'lbph', 'svi', 'lcp', 'gleason', 'pgg45']].values
y_train = train_data['lpsa'].values
x_test = test_data[['lcavol', 'lweight',
    'lbph', 'svi', 'lcp', 'gleason', 'pgg45']].values
y_test = test_data['lpsa'].values

# 创建线性回归模型并拟合训练数据
reg = LinearRegression().fit(x_train, y_train)

# 对测试集进行预测
y_pred = reg.predict(x_test)
```

2.2.4 实验结果

本实验利用 sklearn 库线性回归模型对测试数据集拟合后对测试集进行预测。最后计算 MSE 和 R^2.

表 2.5: 实验指标

MSE	R^2
0.4079	0.6114

2.2.5 实验改进

从 R^2 和 MSE 两个回归分析指标来看，直接使用线性回归模型对前列腺癌数据进行回归拟合效果并不是很好。本实验尝试了使用梯度下降进行回归拟合、主成分分析两种方法来获得更好的回归表现。

随机梯度下降（Stochastic Gradient Descent，SGD）通过随机抽取一个训练样本计算参数梯度，根据梯度更新模型参数，直到一定迭代次数。随机梯度下降相较于传统的梯度下降算法能够更快的收敛，但也可能会出现震荡的现象，因此需要对超参数进行多次调整。

```
# 提取训练集中的自变量 X 和因变量 y

X_train = train_data[['lcavol', 'lweight', 'lbph',
    'svi', 'lcp', 'gleason', 'pgg45']].values
y_train = train_data['lpsa'].values.reshape(-1, 1)

# 初始化梯度下降回归模型并拟合数据

sgd = SGDRegressor(learning_rate='constant', eta0=0.05,
    max_iter=132000, penalty=None)
sgd.fit(X_train, y_train.ravel())

# 提取测试集中的自变量 X 和因变量 y

X_test = test_data[['lcavol', 'lweight',
    'lbph', 'svi', 'lcp', 'gleason', 'pgg45']].values
y_test = test_data['lpsa'].values.reshape(-1, 1)

# 在测试集上评估模型性能

y_pred = sgd.predict(X_test)
```

本实验对随机梯度下降重复实验 100 次，分别计算 MSE 和 R^2 并求均值，最终结果见表 2.6。

表 2.6: 实验指标

MSE	R^2
0.4079	0.6114

在进行主成分分析之前本实验先对特征量进行了相关性分析，使用皮

尔逊相关系数（式 2.4）来描述变量之间得线性关系强度。

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.4}$$

通过相关性分析我们可以判断前列腺癌特征量中是否存在一些相关性较强的特征变量，本实验相关性分析结果见图 2.12.

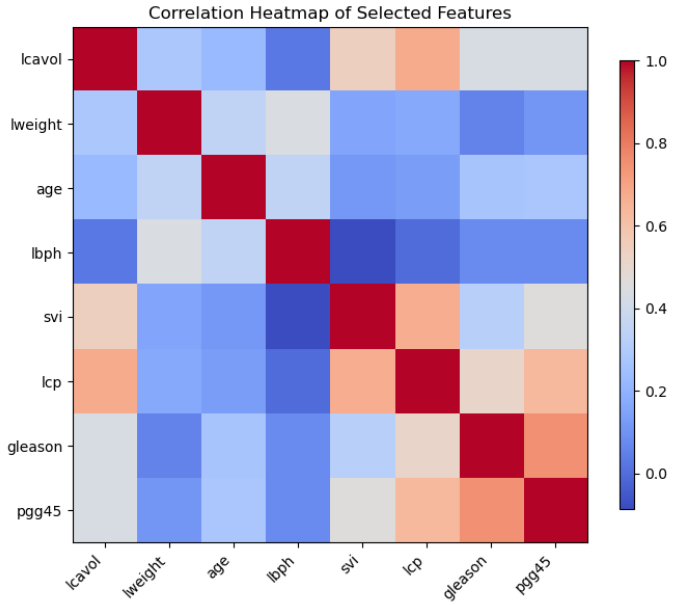

图 2.12: 前列腺癌特征热力图

主成分分析（Principal Component Analysis，PCA）是一种经典的降维算法，常用于高维数据的特征提取和可视化。其基本思想是通过线性变换将原始数据映射到低维空间中，从而实现降维。尤其是在多维回归分析时数据可能存在冗余或者噪声，会影响模型的准确性。主成分分析在保留原始数据的主要结构信息的前提下，可以有效减少数据冗余、噪声，降低了过拟合程度，提升了模型的泛化性能。

```python
# 选择需要进行降维的特征
features = ['lcavol', 'lweight', 'age', 'lbph', 'svi', 'lcp',
    'gleason', 'pgg45']
x = data.loc[:, features].values

# 进行标准化处理
from sklearn.preprocessing import StandardScaler
x = StandardScaler().fit_transform(x)

# 创建 PCA 模型并拟合数据
pca = PCA(n_components=8)
x_pca = pca.fit_transform(x)

# 输出主成分的方差贡献率
print('Explained variance ratio:', pca.explained_variance_ratio_)

# 将降维后的数据转换为 DataFrame 并加入 data
df_pca = pd.DataFrame(data=x_pca, columns=['PC1', 'PC2','PC3',
    'PC4','PC5'])
```

一般而言，主成分分析保留的主成分方差占原数据方差至少应在 80% 以上，但是保留过多的主成分又会使得主成分分析法的效果下降。本实验中，保留到第 4、5、6 主成分的方差占原数据的方差分别为：82.19%、88.83%、94.35%.

因此考虑依次选择保留前 4、5、6 个主成分进行 100 次重复实验，最终得到的 MSE、R^2 如表 2.7。

表 2.7: 实验指标

保留前 n 个主成分	MSE	R^2
4	0.3833	0.6348
5	0.3340	0.6818
6	0.3509	0.6657

上述实验结果验证了使用主成分分析法确实能够有效去除数据噪声、冗余。

Chapter 3

手写数字识别

3.1 Mnist 数据集识别

3.1.1 实验内容

利用 ModelArts 上的 Minist 手写数字集，采用 AI 开发框架 Mindspore 实现训练和测试手写数字识别。对比平方差损失和交叉熵损失，Mini-batch 和 no batch，Relu 和 Sigmoid，有 dropout 和无 dropout 的实验结果（详见课件要求），同样请统一训练集与测试集。

3.1.2 数据处理

MNIST（Modified National Institute of Standards and Technology）数据集是一个常用的手写数字识别数据集，其中包含了大量的手写数字图像及其对应的标签。该数据集由美国国家标准与技术研究所（NIST）的员工和美国邮政服务（USPS）的员工共同收集而成。

MNIST 数据集包含 60000 张训练图像和 10000 张测试图像，每张图像大小为 28x28 像素，像素值在 0 到 255 之间，并被保存为灰度图像。每个图像的标签（Label）为 0-9 之间的一个数字。

Mindspore 可以非常简便地读取 Mnist 数据集，并将其转换成一个键为 image、label 的数据字典。本实验中每个 image 为形状为 (28,28,1) 的张量，label 为 0-9 之间的数字。

在进行训练之前，本实验通过图像增强技术对手写数字数据集图片进行了一系列操作，可以使得图像更好地适应深度学习模型对输入的要求，从而提高识别准确率。首先对图像进行归一化处理，将像素值减去均值除以方差，使得数据满足标准正态分布，可以帮助模型训练时更好的收敛，降低梯度消失或梯度爆炸的可能性。归一化处理数学表达式见式 3.1，其中 x 代表原始像素值，μ 代表图像像素均值，σ 代表图像像素标准差。

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{3.1}$$

调整图像的大小、张量形状，使其符合模型的输入，也可以一定程度上减少计算量。

```
# According to the parameters, generate the data enhancement method
 # Resize images to (32, 32) by bilinear interpolation
resize_op = CV.Resize((resize_height, resize_width))
rescale_nml_op = CV.Rescale(rescale_nml, shift_nml) # normalize images
rescale_op = CV.Rescale(rescale, shift) # rescale images
#from (height, width, channel) to (channel, height, width) to fit network.
hwc2chw_op = CV.HWC2CHW()
# change data type of label to int32 to fit network
type_cast_op = C.TypeCast(mstype.int32)
```

手写数字集部分图片见图 3.1.

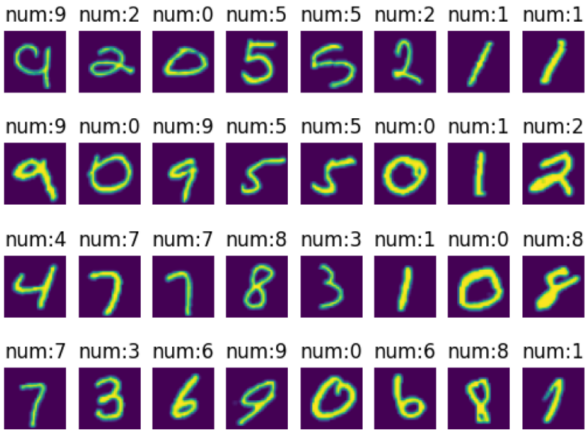

图 3.1: 手写数字集示例图

3.1.3 模型介绍

本实验采用经典卷积神经 LeNet 模型对 Mnist 手写数据集进行识别。总体来看，LeNet 分为两个部分：

- 卷积编码器：由两个卷积层组成。
- 全连接层密集块：由三个全连接层组成。

每个卷积块中的基本单元是一个卷积层、一个 sigmoid 激活函数和平均汇聚层。每个卷积层使用 5×5 卷积核和一个 sigmoid 激活函数。这些层将输入映射到多个二维特征输出，通常同时增加通道的数量。第一卷积层有 6 个输出通道，而第二个卷积层有 16 个输出通道。每个 2×2 池操作（步幅 2）通过空间下采样将维数减少 4 倍。卷积的输出形状由批量大小、通道数、高度、宽度决定。

该神经网络架构见图 3.2.

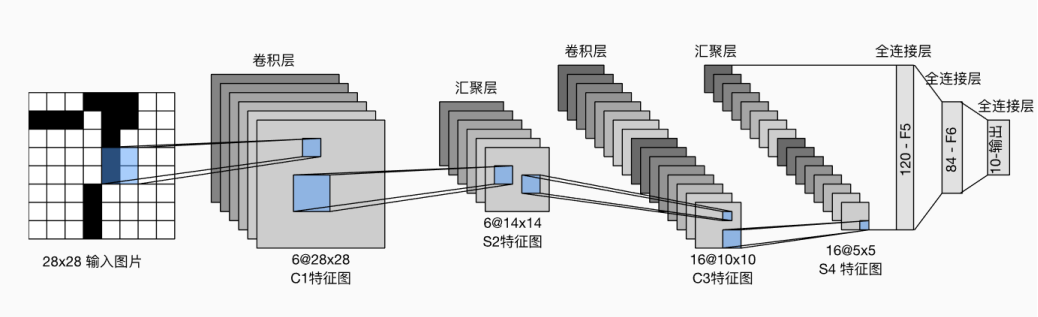

图 3.2: LeNet 结构图

```
class LeNet5(nn.Cell):
    def __init__(self):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, stride=1, pad_mode='valid')
        self.conv2 = nn.Conv2d(6, 16, 5, stride=1, pad_mode='valid')
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(400, 120)
```

```
        self.fc2 = nn.Dense(120, 84)
        self.fc3 = nn.Dense(84, 10)
        self.dropout = nn.Dropout(keep_prob=0.5)
    def construct(self, x):
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        return x
```

3.1.4 实验结果

根据实验要求，对于相同的测试集，分别在损失函数为平方差损失、交叉熵损失，Mini-batch、no-batch，激活函数为 ReLU、Sigmoid，有 dropout、无 dropout 几种情况下进行训练，并进行测试。

这里需要说明的是本实验采用单一变量法。首先使用损失函数为交叉熵、Mini-batch、激活函数为 ReLU、无 Dropout 层的神经网络作为对照模型，每次实验时仅改变一个变量，其余变量保持不变，在同一测试、训练集上进行训练、测试，得到结果。

对照模型训练过程见图 3.3

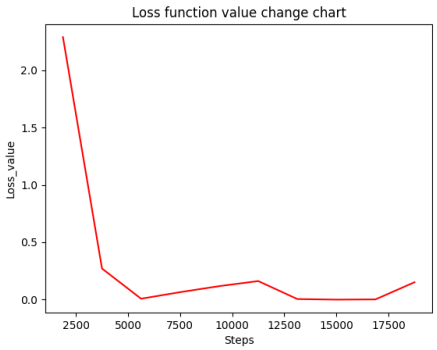

图 3.3: 对照模型 loss 曲线图

下面给出对照模型的混淆矩阵。

$$\begin{pmatrix} 965 & 0 & 2 & 0 & 0 & 1 & 3 & 2 & 0 & 6 \\ 0 & 1130 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1026 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 998 & 0 & 7 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 974 & 0 & 2 & 0 & 0 & 5 \\ 0 & 0 & 0 & 4 & 0 & 884 & 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 & 3 & 4 & 946 & 0 & 0 & 0 \\ 1 & 3 & 17 & 0 & 1 & 0 & 0 & 1003 & 1 & 1 \\ 4 & 1 & 1 & 2 & 0 & 3 & 5 & 0 & 954 & 2 \\ 1 & 0 & 0 & 0 & 8 & 8 & 0 & 5 & 2 & 981 \end{pmatrix}$$

图 3.8 给出了四次实验模型训练过程中 loss 曲线。

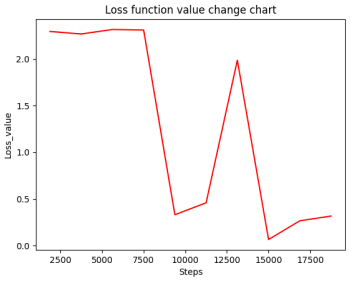

图 3.4: MSE 损失函数 loss 曲线

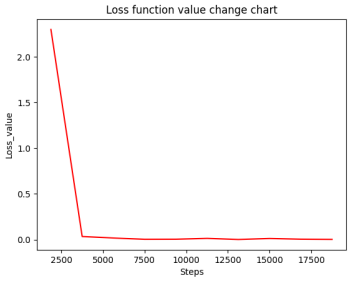

图 3.5: no-batch loss 曲线

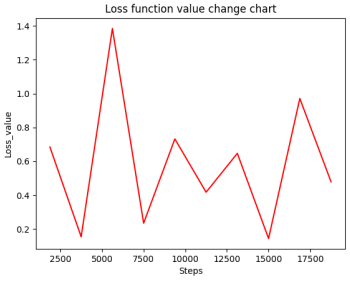

图 3.6: Sigmoid 激活函数 loss 曲线

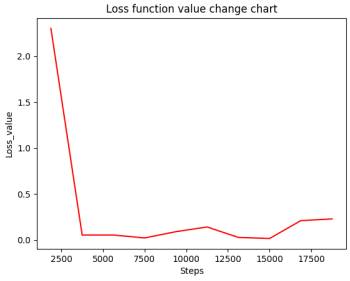

图 3.7: 有 dropout loss 曲线

图 3.8: 四种措施训练过程 loss 曲线

由四种措施训练过程中的 loss 曲线可以看出，对于分类问题而言与交叉熵相比 MSE 作为损失函数可能会导致模型训练过程中出现震荡，不利于快速收敛。采用 no-batch 策略，一次性把所有手写数字图像作为测试集进行训练，loss 很快收敛，并且不再波动，说明每一次学习整个测试集会很快学习所有特征，导致过拟合程度上升。使用 Sigmoid 作为激活函数，在极端

取值情况下很容易出现梯度消失的问题，loss 曲线会不断震荡，难以收敛。在对照模型中加入 dropout 层可以有效降低模型的过拟合程度，在训练的后阶段 loss 曲线的震荡幅度有所减小。

对照模型与四组实验模型在测试集上表现见表 3.1.

表 3.1: 模型测试结果

模型	指标			
	Accuracy	Precision	Recall	F1 Score
对照模型	98.768 %	98.756 %	98.760 %	98.747 %
MSE 损失函数	93.920 %	94.316 %	93.815 %	93.939 %
nobatch 策略	98.528 %	98.540 %	98.529 %	98.528 %
sigmoid 激活函数	93.520 %	93.599 %	93.498 %	93.481 %
有 dropout 层	98.377 %	98.408 %	98.370 %	98.384 %

四次实验的混淆矩阵结果见图 3.13

$$\begin{pmatrix} 918 & 3 & 16 & 1 & 0 & 10 & 4 & 2 & 8 & 18 \\ 0 & 1117 & 11 & 0 & 0 & 1 & 0 & 3 & 0 & 0 \\ 0 & 2 & 1003 & 0 & 0 & 2 & 1 & 8 & 10 & 4 \\ 0 & 0 & 11 & 905 & 5 & 17 & 0 & 18 & 10 & 39 \\ 2 & 0 & 18 & 0 & 886 & 0 & 1 & 5 & 1 & 67 \\ 0 & 6 & 4 & 10 & 4 & 818 & 1 & 2 & 5 & 40 \\ 6 & 4 & 23 & 0 & 24 & 7 & 889 & 0 & 0 & 5 \\ 0 & 18 & 24 & 1 & 4 & 1 & 0 & 955 & 2 & 23 \\ 4 & 2 & 15 & 0 & 5 & 9 & 3 & 1 & 912 & 22 \\ 2 & 1 & 5 & 0 & 19 & 0 & 0 & 7 & 0 & 974 \end{pmatrix}$$

图 3.9: MSE 混淆矩阵

$$\begin{pmatrix} 968 & 0 & 0 & 0 & 1 & 0 & 7 & 1 & 2 & 0 \\ 3 & 1115 & 3 & 3 & 1 & 1 & 4 & 0 & 2 & 0 \\ 1 & 0 & 1018 & 0 & 1 & 0 & 0 & 7 & 1 & 1 \\ 0 & 0 & 1 & 1005 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 0 & 960 & 0 & 3 & 0 & 0 & 17 \\ 0 & 0 & 0 & 15 & 0 & 875 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 2 & 950 & 0 & 3 & 0 \\ 1 & 4 & 4 & 3 & 1 & 1 & 0 & 993 & 1 & 19 \\ 1 & 0 & 2 & 0 & 1 & 4 & 1 & 0 & 956 & 6 \\ 0 & 0 & 0 & 7 & 2 & 1 & 0 & 0 & 0 & 997 \end{pmatrix}$$

图 3.10: no-batch 混淆矩阵

$$\begin{pmatrix} 939 & 0 & 4 & 0 & 1 & 3 & 16 & 1 & 13 & 3 \\ 0 & 1108 & 2 & 0 & 2 & 2 & 17 & 1 & 1 & 0 \\ 7 & 6 & 960 & 0 & 1 & 0 & 2 & 32 & 20 & 3 \\ 0 & 1 & 20 & 897 & 0 & 26 & 0 & 46 & 6 & 13 \\ 1 & 2 & 0 & 0 & 936 & 0 & 13 & 3 & 4 & 19 \\ 1 & 4 & 0 & 3 & 0 & 856 & 3 & 1 & 4 & 18 \\ 6 & 3 & 0 & 0 & 6 & 39 & 901 & 0 & 3 & 0 \\ 4 & 6 & 27 & 1 & 4 & 0 & 0 & 974 & 7 & 5 \\ 8 & 4 & 11 & 0 & 22 & 15 & 24 & 8 & 875 & 6 \\ 8 & 9 & 0 & 0 & 31 & 4 & 0 & 57 & 4 & 891 \end{pmatrix}$$

图 3.11: Sigmoid 混淆矩阵

$$\begin{pmatrix} 971 & 1 & 0 & 1 & 0 & 0 & 2 & 0 & 3 & 0 \\ 0 & 1132 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 12 & 995 & 2 & 1 & 0 & 2 & 9 & 2 & 1 \\ 1 & 1 & 2 & 999 & 0 & 2 & 0 & 3 & 1 & 0 \\ 1 & 2 & 2 & 0 & 962 & 0 & 1 & 0 & 2 & 8 \\ 1 & 1 & 0 & 8 & 0 & 877 & 4 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 & 1 & 0 & 948 & 0 & 1 & 0 \\ 1 & 5 & 4 & 5 & 1 & 0 & 0 & 1008 & 1 & 3 \\ 6 & 1 & 1 & 1 & 1 & 0 & 1 & 2 & 956 & 2 \\ 3 & 3 & 0 & 2 & 17 & 4 & 0 & 3 & 2 & 974 \end{pmatrix}$$

图 3.12: 有 dropout 混淆矩阵

图 3.13: 四种措施混淆矩阵

3.1.5 实验改进

对于本实验而言可以进一步优化 LeNet 的网络结构，主要考虑从以下几个方面：

- 增加卷积层深度：适当增加卷积层的深度可以提高模型的特征提取能力，从而提高分类效果。
- 使用更复杂的卷积核：可以尝试使用更大的卷积核或者多种不同大小的卷积核来捕获更多的特征。
- 使用批量归一化技术：批量归一化可以使模型更加稳定，加速模型收敛，以及提高模型的泛化能力。
- 增加 Dropout：Dropout 是一种常用的正则化方法，可以有效降低模型的过拟合程度。
- 调整超参数：超参数包括学习率、batch size、优化算法等等，可以通过网格搜索等方法寻找最优的超参数组合，以提高模型性能。

优化后的 LeNet 结构见下面代码。

```
        class LeNetImproved(nn.Cell):
def __init__(self, num_class=10):
    super(LeNetImproved, self).__init__()

    self.conv1 = nn.Conv2d(1, 32, kernel_size=5, pad_mode='valid')
    self.bn1 = nn.BatchNorm2d(num_features=32)
    self.relu1 = nn.ReLU()
    self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2)

    self.conv2 = nn.Conv2d(32, 64, kernel_size=3, pad_mode='valid')
    self.bn2 = nn.BatchNorm2d(num_features=64)
    self.relu2 = nn.ReLU()
    self.maxpool2 = nn.MaxPool2d(kernel_size=2, stride=2)

    self.flatten = nn.Flatten()
    self.fc1 = nn.Dense(64 * 4 * 4, 256, weight_init=TruncatedNormal(0.02))
    self.bn3 = nn.BatchNorm1d(num_features=256)
```

```
        self.relu3 = nn.ReLU()
        self.dropout = nn.Dropout(keep_prob=0.5)
        self.fc2 = nn.Dense(256, 128, weight_init=TruncatedNormal(0.02))
        self.bn4 = nn.BatchNorm1d(num_features=128)
        self.relu4 = nn.ReLU()
        self.fc3 = nn.Dense(128, num_class, weight_init=TruncatedNormal(0.02))

    def construct(self, x):
        x = self.maxpool1(self.relu1(self.bn1(self.conv1(x))))
        x = self.maxpool2(self.relu2(self.bn2(self.conv2(x))))
        x = self.flatten(x)
        x = self.relu3(self.bn3(self.fc1(x)))
        x = self.dropout(x)
        x = self.relu4(self.bn4(self.fc2(x)))
        x = self.fc3(x)
        return x
```

改进后的 LeNet 测试结果如表：

评价指标	值
Accuracy	99.398%
Precision	99.394%
Recall	99.385%
F1 Score	99.389%

表 3.2: 改进后 LeNet 实验结果

3.2 美国邮编数据集识别

3.2.1 实验内容

对 Mnist 手写数据集模型稍加修改，迁移到美国邮政编码手写数字集的数字识别上，并计算 F1、混淆矩阵、准确率。

3.2.2 数据介绍

美国邮编手写数字集是一个手写数字识别数据集，包含了 USPS 部门自动扫描信件获得的数字图像，图像大小为 16x16 像素，共有 9298 个手写数字图像。这个数据集可以用于机器学习领域的分类任务，也可以用于迁移学习和连续学习等研究。

本实验的数据处理部分与 Mnist 手写数据集类似，在此不做赘述。

美国邮编代码手写数字集部分图片见图 3.14.

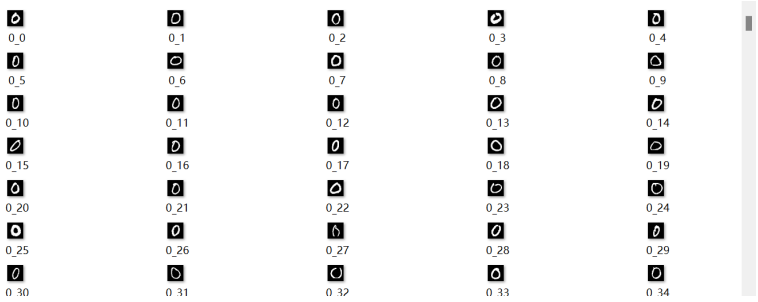

图 3.14: 美国邮编代码手写数据集示例图

3.2.3 模型介绍

本实验采用的模型是由 Mnist 数据集使用的改进后 LeNet 模型。

3.2.4 实验结果

在上述数据集上测试得到的实验结果如表 3.2

评价指标	值
Accuracy	99.228%
Precision	99.221%
Recall	99.222%
F1 Score	99.221%

表 3.3: 美国邮政代码手写数字集实验结果

3.3 遗传算法生成数字识别

3.3.1 实验内容

使用 Mnist 手写数据集手写模型对遗传算法生成的数字进行识别，并计算识别准确率。

3.3.2 数据介绍

遗传算法可以用来解决最优化问题，具体过程为：首先随机生成一组可行解，然后根据问题的特定要求设计适应度函数对这些解进行评估，对于适应度高的解进行遗传操作（如交叉、变异等），获得新一代解。重复执行遗传操作，直到满足某一终止条件。

在产生数字方面，可以将数字的每一位视为一个基因，对数字进行编码，然后使用遗传算法进行进化，得到最优解即可。例如，可以将需要产生的数字转化为二进制数，将每一位作为一个基因，然后使用遗传算法进行进化，直到找到适应度最高的数字。

本次实验利用遗传算法对数字 0-9 每个类别生成了 100 张图片，一共 1000 张图片。该数据集的部分图片示例见图 3.15

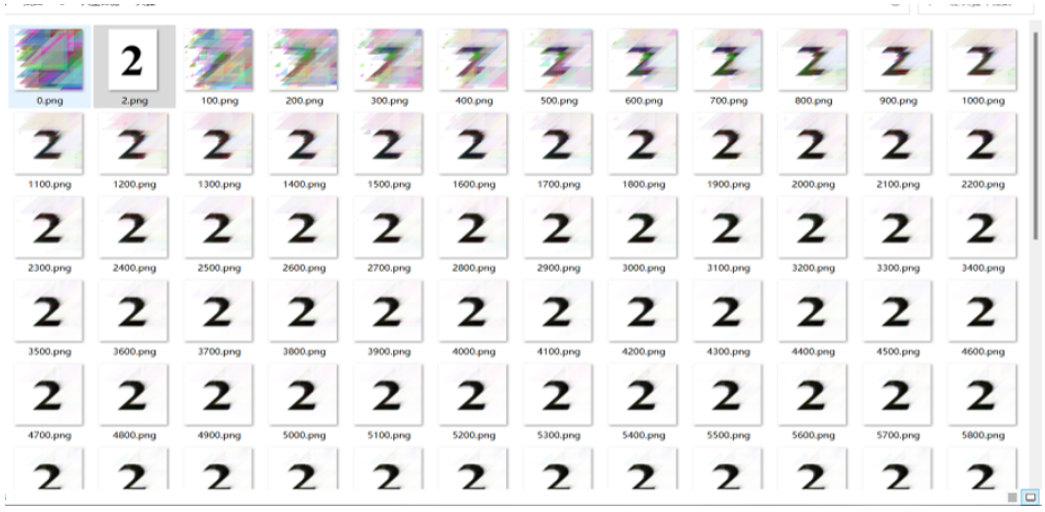

图 3.15: 遗传算法生成数字数据集示例图

3.3.3 模型介绍

本实验采用的模型是由 Mnist 数据集使用的改进后 LeNet 模型。

3.3.4 实验结果

在上述数据集上测试得到的实验结果如表 3.3

评价指标	值
Accuracy	99.278%
Precision	99.275%
Recall	99.269%
F1 Score	99.272%

表 3.4: 遗传算法生成数字实验结果

Chapter 4

乳腺癌识别

4.1 肿瘤分类

4.1.1 实验内容

完成图像级别的分类任务，用 ModelArts 的标签工具对图像进行分类标注，训练模型完成良性、恶性和正常 B 超影像的分类识别任务。评价指标为每类数据的 F1 值。

4.1.2 数据介绍

Dataset_BUSI_with_GT 是一个用于乳腺超声成像(Breast Ultrasound Imaging）的数据集。本实验使用的数据根据乳腺癌的不同病理程度分为: normal(正常)、benign(良性)、malignant(恶性), 分别有 133 张、437 张、210 张。对于每一张图片还会有一张 mask 图片。在计算机视觉、机器学习等领域中，通常使用 mask （掩码）来表示图像中某些特定区域的信息。掩码是一种二值化的图像，其中 0 表示该像素不属于感兴趣的区域，1 表示该像素属于感兴趣的区域。

本实验所用的部分原始数据集见图 4.1。

在本任务中,为每一张图片都标上了两个标签,分别是 normal/benign/-malignant 和 photo/mask。这样不仅区分开了图片和 mask 图，更可以实现通过图片或者 mask 图都能判断病症轻重的功能。部分标注过程见图 4.2

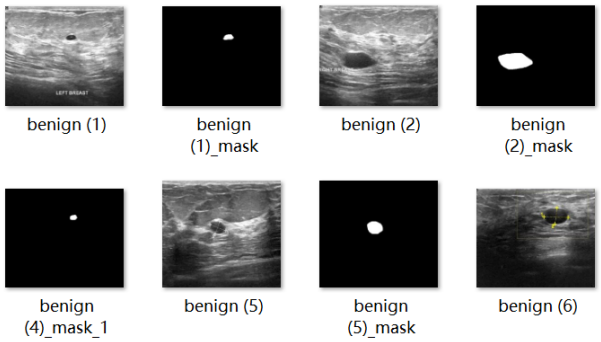

图 4.1: 乳腺癌数据集示例图

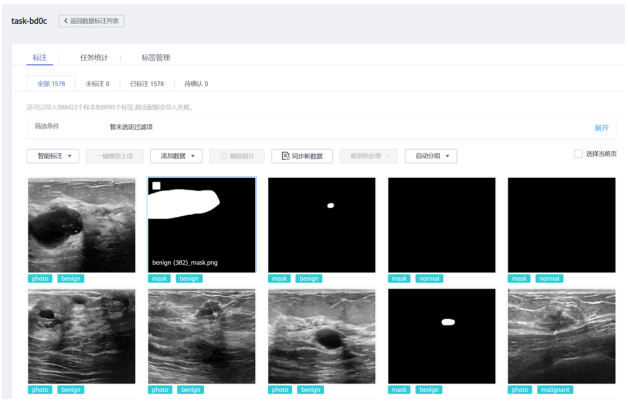

图 4.2: 肿瘤分类标注示例图

各类标签数量信息见表 4.1

标签	benign	malignant	mask	normal	photo
数量	891	421	798	266	780

表 4.1: 肿瘤分类标注信息

4.1.3 模型介绍

本任务选择了 ModelArts 中的 ResNet_v1_50, 该模型为 ResNet 系列，深度为 50 层，此模型基于 Deep Residual Learning for Image Recognition

中提出的模型结构实现。可以用图像分类任务，比如猫狗分类、花卉分类等等。用户提供一系列带有标注的数据集，该算法会载入在 ImageNet-1000 上的预训练模型，在用户数据集上做迁移学习。训练后生成的模型可直接在 ModelArts 平台部署成在线服务，同时支持使用 CPU 和 GPU 规格进行推理。模型训练的基本信息如图 4.3。

图 4.3: 肿瘤分类模型信息

4.1.4 实验结果

实验结果见表 4.2.

评价指标	值
Accuracy	99.219%
Precision	98.214%
Recall	96.887%
F1 Score	97.504%

表 4.2: 肿瘤分类实验结果

4.2 肿瘤检测

4.2.1 实验内容

完成肿瘤检测任务，用 ModelArts 的标签工具对图像进行检测框标注，不仅要完成良性、恶性等的识别，还要求用检测框定位出肿瘤位置。评价指标为分类的 F1 和检测框的 mIOU（均交并比，良性、恶性两类检测框的 IOU 均值）。

4.2.2 数据介绍

本任务使用的原始数据与任务 C 完全一致，只是数据标注有所不同。这一次只设定两个标签 benign/malignant，但需要将病理位置仔细框出。部分标注过程见图 4.4.

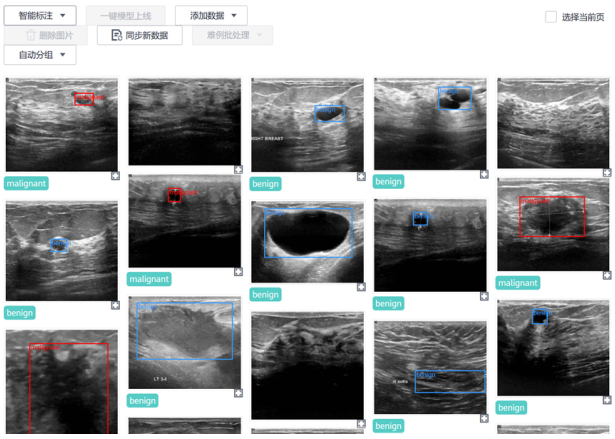

图 4.4: 肿瘤检测标注示例图

各类标签数量信息见表 4.3。

标签	benign	malignant
数量	457	215

表 4.3: 肿瘤检测标注信息

4.2.3 模型介绍

本任务选择了 ModelArts 中的 YOLOv3_ResNet18。针对提供带有物体框标注的数据集，该算法会载入在 ImageNet-1k 上的 ResNet18 预训练模型，训练后生成的模型可直接在 ModelArts 平台部署成在线服务，同时支持使用 CPU 和 GPU 规格进行推理。用户还可以将模型转换成 Ascend 类型，并在 Hilens 芯片上部署推理。

模型训练的基本信息如图 4.3。

图 4.5: 肿瘤检测模型信息

4.2.4 实验结果

实验结果见表 4.2.

评价指标	值
Accuracy	89.018%
Precision	90.112%
Recall	87.126%
F1 Score	89.565%

表 4.4: 肿瘤检测实验结果

4.3 肿瘤分割

4.3.1 实验内容

完成肿瘤分割任务，利用数据集提供的 MASK 图像提取分割标签，不仅要完成良性、恶性等的识别，还需要分割出肿瘤轮廓。评价指标为分类的 F1 和 MASK 的 mIOU。

4.3.2 方法介绍

本任务需要在前面检测任务的基础上将检测框转换为 mask 形式，再与原数据集中的 mask 图进行比较，得到 mIOU 值。

在检测任务中能够得到检测框的坐标信息，为 txt 文件。手动标注的图片检测框坐标为 xml 文件。

通过下述代码可以计算每张图片的 IOU 值。

```
def iou(box1, box2):
    x1, y1, x2, y2 = box1
    x3, y3, x4, y4 = box2
    intersection_left = max(x1, x3)
    intersection_top = max(y1, y3)
    intersection_right = min(x2, x4)
    intersection_bottom = min(y2, y4)
    if intersection_left >= intersection_right or intersection_top
        >= intersection_bottom:
        return 0
    intersection_area = (intersection_right - intersection_left) * \
                        (intersection_bottom - intersection_top)
    box1_area = (x2 - x1) * (y2 - y1)
    box2_area = (x4 - x3) * (y4 - y3)
    iou_value = intersection_area /
        float(box1_area + box2_area - intersection_area)
    return iou_value

for i in range(100):
    pred_file = 'shuru/malignant ({0}).xml'.format(i+1)
    try:
```

```
        with open(pred_file,"r") as f:
            lines = f.readlines()
    except FileNotFoundError:
        # 如果文件不存在，跳过循环
        continue
    linexm = lines[22]
    match = re.search(r'\d+', linexm)
    linexm = match.group(0)
    lineym = lines[23]
    match = re.search(r'\d+', lineym)
    lineym = match.group(0)
    linexa = lines[24]
    match = re.search(r'\d+', linexa)
    linexa = match.group(0)
    lineya = lines[25]
    match = re.search(r'\d+', lineya)
    lineya = match.group(0)
    box1 = [lineym, linexm, lineya, linexa]
    print(box1)

    pred_file = 'result/malignant ({0}).png result.txt'.format(i+1)
    try:
        with open(pred_file,"r") as f:
            lines = f.readlines()
    except FileNotFoundError:
        # 如果文件不存在，跳过循环
        continue
    my_dict = json.loads(lines[0])
    box2 = my_dict.get('detection boxes')
    box2 = [elem for row in box2 for elem in row]
```

4.3.3 实验结果

最终计算得 mIOU 值为：0.904859.

部分标注图片的 IOU 值如图 4.6.

```
0.8749344956810315
['141', '205', '269', '383']
['143.83011', '205.40675', '272.29898', '386.79828']
205.40675
0.93174109914565
['189', '173', '271', '293']
['188.39447', '173.88867', '269.6362', '292.5695']
173.88867
0.9655066260749671
['76', '27', '177', '290']
['71.61499', '36.627872', '177.88974', '293.55804']
36.627872
0.9049771163957055
['46', '18', '214', '208']
['48.78233', '17.76774', '214.52016', '206.36705']
18
0.9708355282549066
['97', '35', '263', '260']
['94.12261', '28.87837', '265.6804', '261.48355']
35
0.9359675264761683
['82', '3', '289', '412']
['87.07373', '7.0935097', '297.32864', '427.89835']
7.0935097
0.8947584050689147
['99', '223', '286', '505']
['89.23156', '221.76703', '296.70917', '525.43726']
223
0.8369842002426081
```

图 4.6: 部分图片 IOU 值

Chapter 5

实验总结与感受

本学期的人工智能实验主要关注机器学习、神经网络以及深度学习，其中包括了回归分析、手写数字识别和乳腺癌识别等多个部分。通过本次实验，我不仅深入理解了人工智能技术的应用和原理，还掌握了开发框架 ModelArts 和 MindSpore 的使用方法和技巧。

在回归分析部分，我对新冠肺炎全国和地区病例数据进行了分析，并建立了合适的回归模型来预测疾病的发展趋势。通过对测试集的验证，我得到了 R2，MSE 指标，并对模型进行了优化和改进，提高了预测精度。

在手写数字识别部分，我使用了 ModelArts 上的 Minist 手写数字集，采用 AI 开发框架 Mindspore 进行训练和测试，并对平方差损失和交叉熵损失、Mini-batch 和 no batch、Relu 和 Sigmoid、有 dropout 和无 dropout 等进行了对比实验。通过实验结果的分析，我得出了最佳的模型，并将该模型稍加修改，迁移到美国邮政编码手写数字集的数字识别上，并进行了改进。尝试使用前期遗传算法生成的手写数字图像进行测试，观察识别效果如何。这个实验不仅让我更加深入地了解了神经网络的应用和训练方法，还增强了我对开发框架 Mindspore 的认识和掌握。

在乳腺癌识别部分，我对不同级别的任务进行了不同的评估，并评价了每个任务的 F1 值、检测框的 mIOU 以及 MASK 的 mIOU 等指标。通过对比实验，我得出了不同模型之间的差异，同时也发现了一些优化空间，为未来的研究提供了参考。

在本次实验的过程中，我遇到了很多课堂上没有提及的问题，通过查

阅资料、咨询老师、请教同学等方法最终克服了一个个困难，最终完成了本次人工智能实验。也让我对于平时所学的理论知识有了更多的思考与理解。

总的来说，本学期的人工智能实验让我更加全面、深入地了解和掌握了人工智能技术的应用和发展，不仅提升了我的理论能力，还增强了我的实践能力，为我未来的研究和工作打下了坚实的基础。

最后，非常感谢老师一直以来的指导！