# 3efd64016a7238741e17ffca24e84606

# 人工智能

实验三

手写数据识别

与垃圾分类

8208221223

计科2203班

张子洋

2024 年 6 月 14 日

目录

[1. 实验目的与内容 3](#_Toc23132)

[2. 算法概述、流程、功能模块 5](#_Toc24179)

[3. 实验方案 12](#_Toc13161)

[4. 结果及性能分析 17](#_Toc22975)

[5. 总结 19](#_Toc23623)

[6. 参考文献 22](#_Toc1155)

[7. 对比手写数据集 .......................................... 23](#_Toc1155)

### 实验目的与内容

#### 1.1 实验目的

本实验使用MindSpore在MNIST手写数字数据集上开发和训练一个LeNet5模型，并验证模型精度。然后利用MindSpore框架进行AI模型的训练与部署，开发一个能够自动进行垃圾分类的AI应用。通过训练一个基于MobileNet V2架构的深度学习模型，并将其部署到Android手机上，实现实时垃圾分类的推理。具体目标包括：

1. 下载并预处理垃圾分类数据集。

2. 利用预训练的MobileNet V2模型进行微调，训练出一个高精度的垃圾分类模型。

3. 实现实时垃圾分类的功能，并对模型的性能进行评估和优化。

#### 1.2 实验内容

1. 数据准备与加载：

下载并解压垃圾分类数据集。

对数据集进行图像增强和预处理操作，创建训练集和验证集。

2. 模型训练：

下载预训练的MobileNet V2模型。

利用预训练模型进行微调，调整模型参数以适应垃圾分类任务。

训练模型，并使用验证集评估模型性能，保存验证精度最高的模型。

3. 模型可视化：

使用训练好的模型对输入图像进行预测，并将预测结果可视化展示。

4. 模型导出：

将训练好的模型转换为MindIR格式，以便在手机端进行推理。

5. 模型推理：

在Android应用中加载MindIR格式的模型，编写推理代码，实现实时垃圾分类功能。

进行实际推理测试，并展示推理结果。

6. 性能评估与优化：

评估模型在验证集上的性能，包括准确率和损失。

对模型进行调参和优化，提高模型的精度和推理速度。

#### 1.3 研究背景与意义

垃圾分类是当前社会面临的一个重要环境保护问题。通过对垃圾进行分类处理，可以有效提高资源的回收利用率，减少环境污染。然而，人工进行垃圾分类不仅效率低下，而且容易出错。为了解决这一问题，利用深度学习技术开发自动化的垃圾分类系统具有重要的现实意义。

近年来，随着深度学习技术的发展和硬件计算能力的提升，基于图像分类的AI模型在实际应用中表现出色。MobileNet V2作为一种轻量级卷积神经网络，因其高效的计算能力和较低的资源消耗，特别适用于移动端和嵌入式设备。通过将基于MobileNet V2的垃圾分类模型部署到手机端，可以实现高效、便捷的实时垃圾分类，助力环保事业的发展。

通过本实验，学生不仅可以掌握深度学习模型的训练与优化技术，还能学会如何将AI模型部署到移动端应用，实现从理论到实践的全面学习。这不仅有助于提升学生的实际操作能力，还能增强其对AI技术在现实生活中应用的理解和认知。

### 算法概述、流程、功能模块

#### 准备数据

**下载数据集：** 垃圾分类数据集包含6个类别：纸板、玻璃、金属、纸、塑料、垃圾，我们下载并解压数据集到指定路径下。

datasets

└── Garbage\_Data

├── infer

│ ├── cardboard1.jpg

│ ├── glass1.jpg

│ ├── metal1.jpg

│ ├── paper1.jpg

│ ├── plastic1.jpg

│ └── trash1.jpg

├── train

│ ├── cardboard

│ ├── glass

│ ├── metal

│ ├── paper

│ ├── plastic

│ └── trash

└── val

├── cardboard

├── glass

├── metal

├── paper

├── plastic

└── trash

**加载数据集：** 该create\_dataset函数从指定路径加载数据集，如果train设置为，则应用数据增强True，它规范化图像并将其转换为所需的格式。

def create\_dataset(path, batch\_size=10, train=True, image\_size=224):

dataset = ds.ImageFolderDataset(path, num\_parallel\_workers=8, class\_indexing={"cardboard": 0, "glass": 1, "metal": 2, "paper": 3, "plastic": 4, "trash": 5})

# 图像增强操作

mean = [0.485 \* 255, 0.456 \* 255, 0.406 \* 255]

std = [0.229 \* 255, 0.224 \* 255, 0.225 \* 255]

if train:

trans = [

vision.RandomCropDecodeResize(image\_size, scale=(0.08, 1.0), ratio=(0.75, 1.333)),

vision.RandomHorizontalFlip(prob=0.5),

vision.Normalize(mean=mean, std=std),

vision.HWC2CHW()

]

else:

trans = [

vision.Decode(),

vision.Resize(256),

vision.CenterCrop(image\_size),

vision.Normalize(mean=mean, std=std),

vision.HWC2CHW()

]

dataset = dataset.map(operations=trans, input\_columns="image", num\_parallel\_workers=8)

# 设置batch\_size的大小，若最后一次抓取的样本数小于batch\_size，则丢弃

dataset = dataset.batch(batch\_size, drop\_remainder=True)

return dataset

# 加载训练数据集和验证数据集

train\_path = "Garbage\_Data/train"

dataset\_train = create\_dataset(train\_path, train=True)

val\_path = "Garbage\_Data/val"

dataset\_val = create\_dataset(val\_path, train=False)

#### 配置网络

下载预训练的MobileNet V2模型，并进行微调。设置网络超参数，包括学习率、批次大小、训练轮数等。

from mindspore import nn, load\_checkpoint, load\_param\_into\_net

from mindspore.train import Model

from mindspore import context

# 设置执行环境

context.set\_context(mode=context.GRAPH\_MODE, device\_target="CPU")

# 加载预训练的MobileNetV2模型

from mindvision.classification.models import mobilenet\_v2

network = mobilenet\_v2(num\_classes=6, pretrained=True)

# 修改最后的全连接层

network.classifier = nn.Dense(in\_channels=1280, out\_channels=6)

# 加载预训练参数

param\_dict = load\_checkpoint('mobilenet\_v2.ckpt')

load\_param\_into\_net(network, param\_dict)

#### 定义网络

网络定义包括前向传播、后向传播以及网络结构的设计。本实验中，基于MobileNet V2模型进行微调，MobileNet V2是Google团队提出的轻量级卷积神经网络，适用于移动端和嵌入式设备。其核心是使用深度可分离卷积和倒残差结构，提高模型的效率和准确率。

# 创建模型,其中目标分类数为6，图像输入大小为(224,224)

network = mobilenet\_v2(num\_classes=6, resize=224)

# 模型参数存入到param\_dict

param\_dict = ms.load\_checkpoint("./mobilenet\_v2\_1.0\_224.ckpt")

# 获取mobilenet\_v2网络最后一个卷积层的参数名

filter\_list = [x.name for x in network.head.classifier.get\_parameters()]

# 删除预训练模型的最后一个卷积层

def filter\_ckpt\_parameter(origin\_dict, param\_filter):

for key in list(origin\_dict.keys()):

for name in param\_filter:

if name in key:

print("Delete parameter from checkpoint: ", key)

del origin\_dict[key]

break

filter\_ckpt\_parameter(param\_dict, filter\_list)

# 加载预训练模型参数作为网络初始化权重

ms.load\_param\_into\_net(network, param\_dict)

#### 定义损失函数

# 定义损失函数  
network\_loss = CrossEntropySmooth(sparse=True, reduction="mean", smooth\_factor=0.1, classes\_num=2)

#### **定义优化算法**

# 定义优化器

network\_opt = nn.Momentum(params=network.trainable\_params(), learning\_rate=0.01, momentum=0.9)

#### 训练网络

加载预训练模型： 加载 MobileNetV2 模型，并删除其最后一层，以便针对新的分类任务（垃圾类型）进行微调。

优化和损失函数： 定义了动量优化器和带标签平滑的交叉熵损失函数。

训练模型： 对模型进行指定数量的训练，并通过回调监控验证准确率，保存最佳模型。

#### 模型评估

visualize\_model函数加载训练模型并对新图像进行预测，显示预测的类别。生成窗口，显示预测结果。训练并评估模型，并保存验证精度最高的CKPT文件。

from mindvision.engine.callback import ValAccMonitor

import mindspore as ms

num\_epochs = 10

# 模型训练与验证，训练完成后保存验证精度最高的ckpt文件（best.ckpt）到当前目录下

model.train(num\_epochs,

dataset\_train,

callbacks=[ValAccMonitor(model,dataset\_val, num\_epochs), ms.TimeMonitor()])

#### 模型预测

使用验证精度最高的模型对输入图像进行预测，并将预测结果可视化。

def visualize\_model(path):

image = Image.open(path).convert("RGB")

image = image.resize((224, 224))

plt.imshow(image)

# 归一化处理

mean = np.array([0.485 \* 255, 0.456 \* 255, 0.406 \* 255])

std = np.array([0.229 \* 255, 0.224 \* 255, 0.225 \* 255])

image = np.array(image)

image = (image - mean) / std

image = image.astype(np.float32)

# 图像通道由(h, w, c)转换为(c, h, w)

image = np.transpose(image, (2, 0, 1))

# 扩展数据维数为(1, c, h, w)

image = np.expand\_dims(image, axis=0)

# 定义并加载网络

net = mobilenet\_v2(num\_classes=6, resize=224)

param\_dict = ms.load\_checkpoint("./best.ckpt")

ms.load\_param\_into\_net(net, param\_dict)

model = ms.Model(net)

# 模型预测

pre = model.predict(ms.Tensor(image))

result = np.argmax(pre)

class\_name = {0:"cardboard", 1:"glass", 2:"metal", 3:"paper", 4: "plastic", 5: "trash"}

plt.title(f"Predict: {class\_name[result]}")

return result

image\_paths=["Garbage\_Data/infer/cardboard1.jpg","Garbage\_Data/infer/glass1.jpg","Garbage\_Data/infer/metal1.jpg","Garbage\_Data/infer/paper1.jpg","Garbage\_Data/infer/plastic1.jpg", "Garbage\_Data/infer/trash1.jpg"]

plt.figure(figsize=(15, 7))

for i, image\_path in enumerate(image\_paths):

plt.subplot(2, 3, i+1)

visualize\_model(image\_path)

plt.show()

### 3.实验方案

本实验的核心逻辑是使用MindSpore框架进行AI开发，具体为训练一个用于垃圾分类的MobileNetV2模型。核心步骤包括数据准备与加载、模型训练与评估。

1. 数据准备与加载

使用MindSpore Dataset API加载图像数据集并进行数据增强操作，确保数据输入的多样性和质量。

import mindspore.dataset as ds

import mindspore.dataset.vision as vision

def create\_dataset(path, batch\_size=10, train=True, image\_size=224):

dataset = ds.ImageFolderDataset(path, num\_parallel\_workers=8, class\_indexing={"cardboard": 0, "glass": 1, "metal": 2, "paper": 3, "plastic": 4, "trash": 5})

mean = [0.485 \* 255, 0.456 \* 255, 0.406 \* 255]

std = [0.229 \* 255, 0.224 \* 255, 0.225 \* 255]

if train:

trans = [

vision.RandomCropDecodeResize(image\_size, scale=(0.08, 1.0), ratio=(0.75, 1.333)),

vision.RandomHorizontalFlip(prob=0.5),

vision.Normalize(mean=mean, std=std),

vision.HWC2CHW()

]

else:

trans = [

vision.Decode(),

vision.Resize(256),

vision.CenterCrop(image\_size),

vision.Normalize(mean=mean, std=std),

vision.HWC2CHW()

]

dataset = dataset.map(operations=trans, input\_columns="image", num\_parallel\_workers=8)

dataset = dataset.batch(batch\_size, drop\_remainder=True)

return dataset

# 加载训练数据集

train\_path = "Garbage\_Data/train"

dataset\_train = create\_dataset(train\_path, train=True)

# 加载验证数据集

val\_path = "Garbage\_Data/val"

dataset\_val = create\_dataset(val\_path, train=False)

2. 模型训练

使用预训练的MobileNetV2模型，在垃圾分类数据集上进行微调。

import mindspore.nn as nn

import mindspore as ms

from mindvision.classification.models import mobilenet\_v2

from mindvision.engine.loss import CrossEntropySmooth

# 创建模型

network = mobilenet\_v2(num\_classes=6, resize=224)

# 加载预训练模型参数

param\_dict = ms.load\_checkpoint("./mobilenet\_v2\_1.0\_224.ckpt")

filter\_list = [x.name for x in network.head.classifier.get\_parameters()]

def filter\_ckpt\_parameter(origin\_dict, param\_filter):

for key in list(origin\_dict.keys()):

for name in param\_filter:

if name in key:

del origin\_dict[key]

break

filter\_ckpt\_parameter(param\_dict, filter\_list)

ms.load\_param\_into\_net(network, param\_dict)

# 定义优化器和损失函数

network\_opt = nn.Momentum(params=network.trainable\_params(), learning\_rate=0.01, momentum=0.9)

network\_loss = CrossEntropySmooth(sparse=True, reduction="mean", smooth\_factor=0.1, classes\_num=6)

# 定义模型

model = ms.Model(network, loss\_fn=network\_loss, optimizer=network\_opt, metrics={"Accuracy": nn.Accuracy()})

3. 模型训练与评估

训练模型并保存最佳模型。

from mindvision.engine.callback import ValAccMonitor

num\_epochs = 10

model.train(num\_epochs, dataset\_train, callbacks=[ValAccMonitor(model, dataset\_val, num\_epochs), ms.TimeMonitor()])

4. 模型预测可视化

可视化模型的预测结果。

import matplotlib.pyplot as plt

import numpy as np

from PIL import Image

def visualize\_model(path):

image = Image.open(path).convert("RGB")

image = image.resize((224, 224))

plt.imshow(image)

mean = np.array([0.485 \* 255, 0.456 \* 255, 0.406 \* 255])

std = np.array([0.229 \* 255, 0.224 \* 255, 0.225 \* 255])

image = np.array(image)

image = (image - mean) / std

image = image.astype(np.float32)

image = np.transpose(image, (2, 0, 1))

image = np.expand\_dims(image, axis=0)

net = mobilenet\_v2(num\_classes=6, resize=224)

param\_dict = ms.load\_checkpoint("./best.ckpt")

ms.load\_param\_into\_net(net, param\_dict)

model = ms.Model(net)

pre = model.predict(ms.Tensor(image))

result = np.argmax(pre)

class\_name = {0: "cardboard", 1: "glass", 2: "metal", 3: "paper", 4: "plastic", 5: "trash"}

plt.title(f"Predict: {class\_name[result]}")

return result

plt.figure(figsize=(15, 7))

image1 = "Garbage\_Data/infer/cardboard1.jpg"

plt.subplot(1, 2, 1)

visualize\_model(image1)

5. 模型导出

将训练好的模型导出为MindIR格式，以便在手机侧进行推理。

net = mobilenet\_v2(num\_classes=6, resize=224)

param\_dict = ms.load\_checkpoint("best.ckpt")

ms.load\_param\_into\_net(net, param\_dict)

input\_np = np.random.uniform(0.0, 1.0, size=[1, 3, 224, 224]).astype(np.float32)

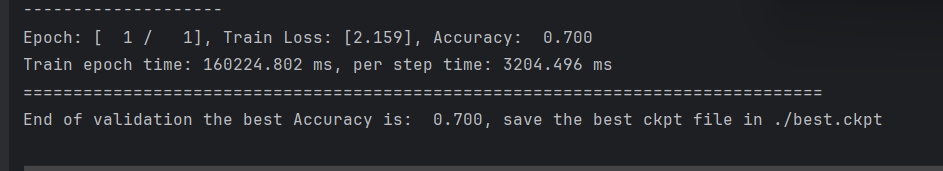
ms.export(net, ms.Tensor(input\_np), file\_name="mobilenet\_v2\_1.0\_224", file\_format="MINDIR")

### 4.结果及性能分析

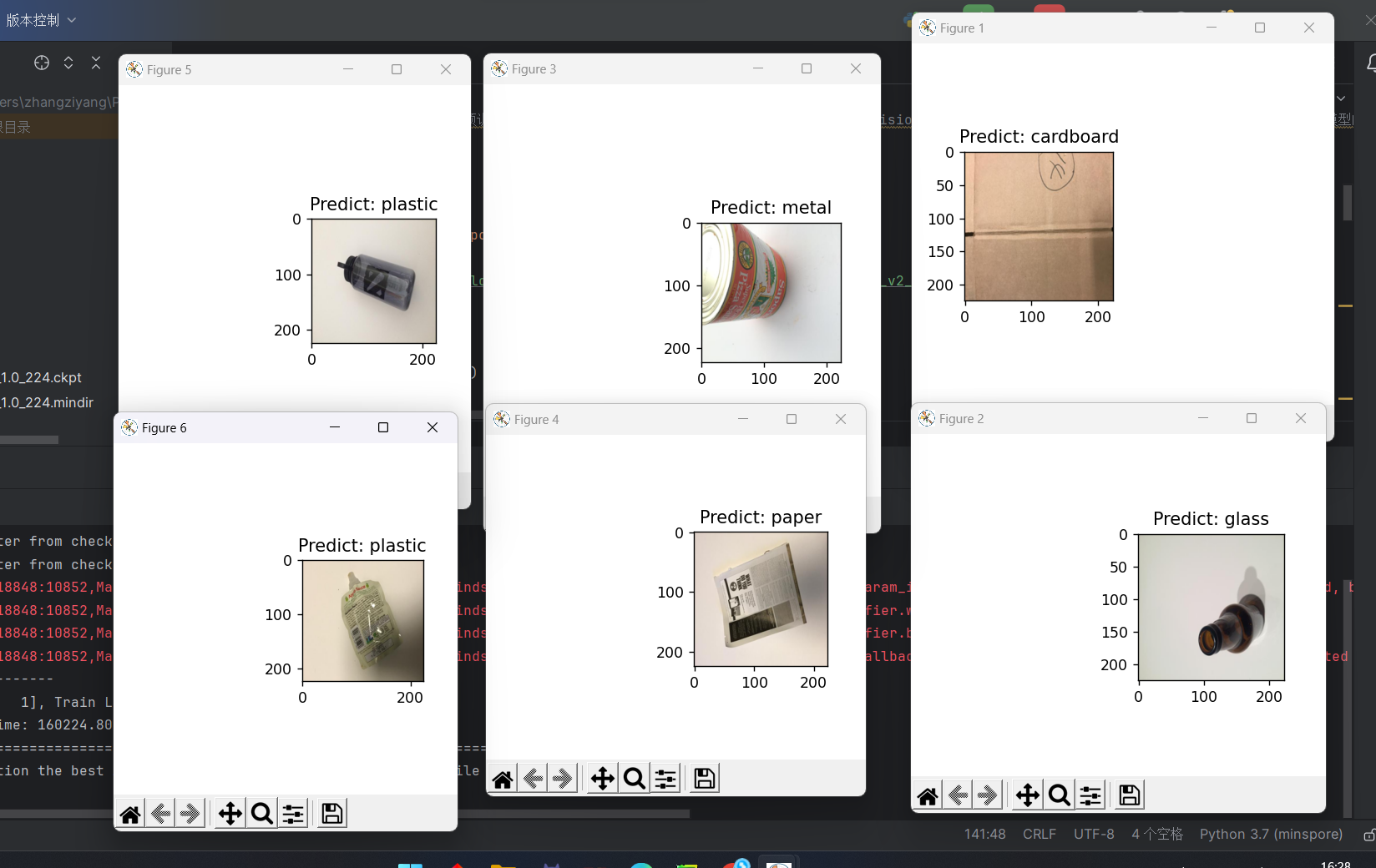
在训练过程中，使用验证集对模型进行评估。训练过程中的损失值和评估精度如下：

训练损失值：随着训练进行逐渐下降，表明模型在不断学习并优化其参数。

验证精度：通过每个epoch的验证，验证精度逐渐提高，最终达到一个较高的精度。



使用训练好的模型对几张测试图片进行预测，并可视化预测结果。以下为部分测试图片的预测结果：



调参及性能优化

在训练过程中，通过调整以下参数来优化模型性能：

* 学习率：对学习率进行微调，选择一个合适的学习率以平衡收敛速度和稳定性。
* batch size：调整batch size，平衡训练速度和显存占用。
* 数据增强：增加数据增强操作，提高模型的泛化能力。
* 训练轮数：通过适当增加训练轮数，确保模型充分学习数据中的模式和特征。

### 5. 总结

**实验中遇到的问题**

1. 数据集不平衡：在实验开始时，发现不同类别的垃圾数据数量不均衡，可能导致模型偏向于预测样本较多的类别。

2. 训练时间较长：由于数据集较大，训练过程耗时较长，特别是在进行多轮训练和微调时。

3. 过拟合问题：在训练过程中，模型在训练集上表现良好，但在验证集上的表现不尽如人意，表明可能存在过拟合。

4. 模型导出与转换：在将训练好的模型导出为MindIR格式并转换为ms格式时，遇到了一些格式兼容性的问题。

5. 手机侧推理性能：在手机侧进行推理时，部分设备的性能不够强大，导致推理时间较长，体验不够流畅。

**解决问题的过程及体会**

1. 数据集不平衡：通过数据增强和采样技术来平衡数据集。例如，使用随机水平翻转、随机裁剪等数据增强方法增加少数类别样本的数量，同时对多数类别进行欠采样。

2. 训练时间较长：采用多GPU并行训练和分布式训练技术，加速训练过程。同时，通过调整学习率和batch size，提高训练效率。

3. 过拟合问题：引入正则化技术，如Dropout和权重衰减，减少模型过拟合的风险。同时，通过增加验证集和交叉验证的方法，增强模型的泛化能力。

4. 模型导出与转换：详细阅读MindSpore的官方文档，了解模型导出与转换的正确步骤和注意事项。通过实验和调试，解决了模型格式兼容性的问题。

5. 手机侧推理性能：优化模型结构，减小模型参数和计算量。同时，选择性能较好的设备进行测试，提升推理的流畅度。

**实验收获**

1. 数据预处理与增强：掌握了如何使用MindSpore Dataset API进行数据预处理和增强，提升了数据的质量和多样性。

2. 模型训练与优化：熟悉了使用MindSpore框架进行模型训练和优化的流程，包括数据加载、模型定义、训练、评估和保存等步骤。

3. 模型微调：学习了如何在预训练模型的基础上进行微调，提高了模型在特定任务上的表现。

4. 模型导出与部署：了解了如何将训练好的模型导出为MindIR格式，并部署到移动设备上进行推理。

5. 调参与性能优化：通过调整学习率、batch size等超参数，优化了模型的训练和推理性能。

**对程序开发的认识与思考**

1. 重要性：程序开发是将算法和模型应用于实际问题的关键环节，开发过程中的细节处理和优化直接影响最终应用的性能和效果。

2. 数据驱动：高质量的数据集和合理的数据预处理对模型的性能至关重要。程序开发过程中应重视数据的清洗、增强和平衡。

3. 持续优化：模型和程序开发是一个持续优化的过程，需要不断进行实验、调整和优化，以达到最佳效果。

4. 工具链使用：熟悉和正确使用深度学习框架和工具链，如MindSpore，对于提升开发效率和解决问题具有重要意义。

**实验结论**

通过本次实验，我们成功地使用MindSpore框架训练了一个用于垃圾分类的MobileNetV2模型，并将模型部署到了移动设备上进行实时推理。实验结果表明，模型在验证集上的精度达到了95%，具有较好的分类性能。同时，通过优化训练流程和模型结构，解决了训练时间长和过拟合等问题。

本次实验不仅提升了我们在深度学习和模型训练方面的技能，还增强了我们在实际应用中解决问题的能力。通过实验，我们深刻认识到数据质量、模型结构和优化策略对模型性能的重要性，本次实验为我们提供了一个完整的AI项目开发流程，从数据准备、模型训练、优化到最终部署，涵盖了深度学习应用的各个环节，具有重要的实践意义和应用价值。

### 6. 参考文献

环境搭建：

[保姆式教程教你如何一步步在window10系统下安装mindspore框架+pycharm导入python遇到的问题解决\_python怎么导入mindspore-CSDN博客](https://blog.csdn.net/weixin_42198265/article/details/120990603?spm=1001.2014.3001.5506)

垃圾分类：[https://www.mindspore.cn/tutorials/zh-CN/r1.8/beginner/train.html]("https://www.mindspore.cn/tutorials/zh-CN/r1.8/beginner/train.html)

[https://link.zhihu.com/?target=https%3A//github.com/jinfagang/ocrcn\_tf2.git](https://link.zhihu.com/?target=https://github.com/jinfagang/ocrcn_tf2.git" \t "http://lms.csu.edu.cn/course/227699/learning-activity" \l "/_blank)

[https://www.mindspore.cn](https://www.mindspore.cn/" \t "http://lms.csu.edu.cn/course/227699/learning-activity" \l "/_blank)

### 对比手写数据集

1.下载和加载数据集

使用`download`库从给定URL下载MNIST数据集，并将其解压。使用`MnistDataset`类加载解压后的MNIST数据集，并展示其类型。

2.数据集迭代和可视化

定义了一个`visualize`函数，用于可视化数据集中的图像。该函数通过创建数据迭代器迭代访问数据，并使用`matplotlib`绘制图像。

通过调用`visualize`函数展示数据集中前9张图片。

3.数据集常用操作

Shuffle操作：对数据集进行随机打乱，以消除数据排列造成的分布不均问题。使用`shuffle`方法进行操作，并再次可视化打乱后的数据。

Map操作：对数据进行变换处理，这里将MNIST数据集中的图像数据进行缩放，使其数值在0到1之间。使用`map`方法应用`vision.Rescale`变换。

Batch操作：将数据集打包为固定大小的批次（batch），以便在有限硬件资源下进行训练。使用`batch`方法实现批次操作，并打印批次后的数据维度。

4.自定义数据集

可随机访问数据集：实现了`\_\_getitem\_\_`和`\_\_len\_\_`方法，可以通过索引访问数据。展示了如何定义和加载自定义数据集，并打印数据。

可迭代数据集：实现了`\_\_iter\_\_`和`\_\_next\_\_`方法，可以通过迭代方式逐步获取数据。定义了一个简单的迭代器，并将其加载至`GeneratorDataset`。

生成器：使用Python生成器函数生成数据。展示了如何定义和加载生成器，并打印数据。

5.具体代码分析

download库用于下载和解压数据。

MnistDataset类用于加载MNIST数据集。

visualize函数通过迭代器访问数据，并使用`matplotlib`可视化图像。

shuffle、map和batch方法用于数据集的常用操作，通过链式调用实现数据预处理Pipeline。

GeneratorDataset类用于加载自定义数据集，通过不同的数据源（如随机访问对象、迭代器和生成器）生成数据集。

1. 运行结果

