**电子科技大学**

**一年级新生课外创新实践项目计划**

**中期检查报告**

**项目名称：** 图片分类设计与开发 **小组组长：** 张璟哲 **学号：** 2024300905011

**联系电话：** 15388447015 **电子邮箱：** 3127305493@qq.com

**院系及专业：** 格拉斯哥海南学院 电子信息

**指导教师：** 邓建 **职称：** 讲师

**联系电话：** 13980082109 **电子邮箱： dengjian@uestc.edu.cn**

**院系及专业：** 计算机科学与工程 学院

**电子科技大学教务处**

**填表日期：** **2025 年** **5 月** **19 日**

**一、项目团队成员**（包括小组组长、按顺序）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 性别 | 所在学院 | 学号 | 联系电话 | 本人签字 |
| 张璟哲 | 男 | 格拉斯哥海南学院 | 2024300095011 | 15388447015 |  |
| 曹城玮 | 男 | 格拉斯哥海南学院 | 2024300901016 | 137 0803 2802 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**二、指导教师意见**

|  |
| --- |
| 签 名：  年 月 日 |

**三、专家组评审意见**（学生不填此栏）

|  |  |  |
| --- | --- | --- |
| 1．是否达到中期目标（在□内打 √ ) :  □达到中期目标 □基本达到中期目标 □未达到中期目标  2．成绩评定（在□内打 √ ) :  □优秀 □合格 | | |
| □不合格，项目实施意见： |  | □改进后可继续执行  □提出警告、观察后再定继续执行或终止  □终止实施 |
| 3．其它意见和建议：  专家组长签名： （ 盖 章 ）  年 月 日 | | |

**四、中期报告**（字数在 2000 字左右）

（一）项目背景（研究现状、趋势、研究意义等）

（二）项目研究内容及实施方案

（三）项目实施的进展情况及初步取得的创新成果(包括遇到的困难、下一步工作计划等)

（四）结题预期目标（结题验收时考核的依据）

**五、格式**

题目（二号黑体，居中，不超 20 字）

Word 文档页面设置为 A4 纸型，页边距各 2 ，行距 16 磅，正文用小四号宋体，其中阿拉 伯数字、英文用 Times New Roman 体。

参考其他文献，包括引用原文或参考、综述、评论他人观点，要在文中加引注标记，采 用顺序编码制，符号按出现的先后顺序为[1][2]…… ，用上角标，与文后所列参考文献序号一 致。参考文献只列出已经公开出版且在文中加注的文献，著录格式另附。文中图、表应有自 明性，且随文出现，须注明图名、表名，按顺序标明序号如表 1 、表 2……、图 1 、图 2……， 图名、表名及内容、参考文献均为五号字。请在稿件首页地脚处给出作者简介信息。

**基于MindSpore** **的图像分类实践与探索**

**（一）项目背景（研究现状、趋势、研究意义等）**

当前，图像分类技术的研究现状主要集中在开发更深、更复杂的卷积神经网络（CNN）以追 求更高的识别精度，例如 ResNet 、Vision Transformer 等模型在各大公开数据集上取得了显 著成果。同时，一个重要的发展趋势是将这些强大的AI模型部署到资源受限的移动端或边缘 设备上，以实现实时、高效的智能应用 。这就对模型的轻量化和高效率提出了更高的要求， 催生了如 MobileNet 系列、ShuffleNet 系列等轻量级网络架构的研发 。这些网络在保持可 接受精度的同时，显著减少了模型参数量和计算复杂度，为 AI在端侧设备上的普及奠定了基 础。

本项目以图像分类为切入点，旨在帮助我们一年级新生了解深度学习的基本概念和一般流程， 初步掌握不同神经网络架构（如 MobileNetV2 、ShuffleNetV1）的设计思想 。通过使用华为 自主研发的 MindSpore 全场景 AI 框架进行模型训练，并利用 MindSpore Lite 将模型部署到 移动端，我们可以获得从理论学习到实际操作的完整体验 。这对于培养我们的实践能力、创 新思维，以及未来在人工智能领域进一步学习和探索具有重要的研究意义和实践价值。

**（二）项目研究内容及实施方案**

本项目的核心研究内容是学习并实践图像分类模型的完整开发与部署流程，具体包括以下几 个方面：

1. **深度学习基础与图像分类原理理解**：学习卷积神经网络（CNN）的基本构成（卷积层、 池化层、全连接层等），了解 MobileNetV2 和 ShuffleNetV1 这类轻量级网络的设计特 点，如深度可分离卷积、分组卷积和通道重排等技术。

2. **MindSpore** **深度学习框架应用**：学习使用 MindSpore 框架进行数据集的准备与加载、 网络模型的构建、模型训练（包括损失函数、优化器的选择与配置，以及微调等技术）、 模型保存与导出等操作 。

3. **模型轻量化与端侧部署**：学习使用 MindSpore Lite 工具将训练好的模型转换为适用于 移动端的 .ms 格式，并将其部署到安卓应用程序中，实现端侧的图像分类推理 。

4. **不同网络模型的实践与对比**：通过先后实现 MobileNetV2 和 ShuffleNetV1 两个模 型，初步感受不同网络结构在实现和部署上的异同。

**实施方案**：

本项目的实施主要分为两个阶段性任务，并依托项目指导书提供的代码框架和数据集进行：

**任务一：基于** **MindSpore** **Lite** **的猫狗分类（MobileNetV2）** 1. **环境搭建**：配置 MindSpore (CPU 版) 及 Python 开发环境。 2. **数据准备与预处理**：下载猫狗图片数据集 ，使用项目 提供的 preprocessing\_dataset.py 脚本对数据进行清洗（去除损坏图片、统一格式）和划分（训 练集、评估集） 。 3. **模型构建与训练**： 以 MobileNetV2 作为骨干网络 ，加载其在

ImageNet 上的预训练权重。 冻结骨干网络参数，替换并训练顶部分类头（适应猫狗二分类 任务） 。 使用 train.py 脚本，配置合适的损失函数、优化器和学习率策略进行模型微调训 练 。4. **模型转换与导出**：将训练得到的最佳模型权重导出为 MindSpore 的 .mindir 格式 ， 再使用 converter\_lite 工具将其转换为 .ms 格式 。 5. **端侧部署与验证**：将 .ms 模型文 件传输到安卓手机，并使用项目提供的 APP 加载模型，对猫狗图片进行分类预测，验证模型 效果 。

**任务二：ShuffleNet** **模型的设计与开发** 1. **模型学习与获取**：学习 ShuffleNetV1 的网络结 构特点。尝试从 MindSpore 的模型库或社区资源（如 mindcv 库）获取 ShuffleNetV1 的

MindSpore 实现。 2. **代码集成与修改**：将获取到的 ShuffleNetV1 模型定义集成到现有项目 框架中，替换原有的 MobileNetV2 部分。这主要涉及修改 src/models.py 中的网络定义部分。

3. **模型训练**：利用 mindcv 库加载 ShuffleNetV1 在 ImageNet 上的预训练权重，并采用与 任务一类似的微调策略，在猫狗数据集上训练和优化 ShuffleNetV1 模型。 4. **模型转换与部** **署**：与任务一类似，将训练好的 ShuffleNetV1 模型导出为 .mindir 和 .ms 格式，并尝试在 安卓 APP 上进行部署和验证。

**（三）项目实施的进展情况及初步取得的创新成果(包括遇到的困难、下一步工作计划等)**

**1.** **项目实施进展情况**

目前，我们已经顺利完成了项目的第一阶段任务，并已开始第二阶段任务的初步工作。

**任务一：基于** MindSpore Lite **的猫狗分类（**MobileNetV2**）已全面完成**

o 成功搭建了 MindSpore CPU 运行环境。

o 完成了猫狗数据集的下载、清洗和按 9:1 比例划分为训练集与评估集。主要通 过 preprocessing\_dataset.py 脚本实现，确保了训练数据的质量。

o 深入学习并实践了 train.py 脚本中的模型训练流程：

理解了如何定义和使用 MobileNetV2 作为骨干网络，并加载预训练权 重。

掌握了通过冻结骨干网络、仅训练分类头（head\_net）的方式进行模型微 调。

学习了损失函数（如 SoftmaxCrossEntropyWithLogits ）、优化器（如 Momentum ）和学习率动态调整策略的配置与作用。

训练结果显示，微调后的模型在猫狗分类任务上的准确率相较于预训练 模型直接预测有了显著提升。

o 掌握了模型导出流程：将训练好的模型从 MindSpore 的检查点（checkpoint） 格式导出为 .mindir 标准格式。

o 掌握 了模 型转 换流程 ： 使用 MindSpore Lite 提供 的 converter\_lite 工 具 将 .mindir 文件成功转换为适用于端侧部署的 .ms 文件。

o 完成了端侧部署验证：将转换后的 pet.ms 文件部署到项目提供的安卓 APP 中，成功实现了对相册图片及摄像头捕获图片的猫狗分类功能，验证了整个流 程的通畅性。

**任务二：**ShuffleNet **模型的设计与开发（初步进行中）**

o 对 ShuffleNetV1 的网络结构和设计理念进行了学习。

o 为了更便捷地获取模型和预训练权重，我们调研并决定采用 mindcv 这个 MindSpore 计算机视觉库。

o 已成功在项目中安装了 mindcv 库。

o 通过 mindcv.list\_models 功能，确认了适用于我们需求的 ShuffleNetV1 模型名 称为 shufflenet\_v1\_g3\_20。

o 已 修 改 src/models.py 文 件 中 的 define\_net 函 数 ， 使 其 调 用 mindcv.create\_model("shufflenet\_v1\_g3\_20", pretrained=True, num\_classes=2) 来 创建 ShuffleNetV1 模型，并利用其 pretrained=True 参数自动加载预训练权重。

o 相应地，已修改 train.py 文件，移除了原先手动调用 load\_ckpt 函数为骨干网 络加载预训练权重的代码行。

**2.** **初步取得的创新成果**

作为一年级新生，本项目对我们而言本身就是一个充满探索和创新的过程。初步成果主要体 现在：

. **完整流程的实践与验证**：从零开始，完整地走通了“数据准备-模型训练-模型转换- 端侧部署 ”的深度学习图像分类项目全流程，这对于我们理解理论知识、建立工程概 念具有重要意义。

o MindSpore **生态初探**：通过本项目，我们对华为 MindSpore 深度学习框架及其 端侧推理引擎 MindSpore Lite 有了初步的认识和实际操作经验。

. **模型替换与库使用能力的初步培养**：在第二阶段任务中，我们学习了如何查找和使用 现有的模型库（mindcv）来替换和集成新的网络模型，这锻炼了我们利用开源资源解 决问题的能力。

. **问题解决能力的提升**：在项目进展中，通过查阅文档、分析代码、讨论交流，解决了 一些实际操作中遇到的问题，提升了自主学习和解决实际工程问题的能力。

**3.** **遇到的困难**

. **初始代码理解难度**：项目初期，面对指导书中提供的完整 Python 训练脚本和部分 C++/Java 推理侧代码，作为新生，理解其内部逻辑、函数调用关系以及各个参数的含 义存在一定的困难。

. **预训练权重文件的获取**：在尝试手动为 ShuffleNetV1 寻找和配置 .ckpt 预训练权重文 件时，遇到了难以直接找到适用下载链接的问题，这促使我们转向使用 mindcv 库。

. **超参数理解与调整**：虽然项目提供了默认配置，但要深入理解不同损失函数、优化器、 学习率等超参数的意义及其对模型训练效果的影响，仍需要更多的学习和实践。

. **CPU** **训练性能瓶颈（预期）**：由于我们目前使用 CPU 进行模型训练，可以预见在训练 ShuffleNetV1 这种相对复杂的网络时，训练时间会非常长，这将是后续工作的一个挑 战。

**4.** **下一步工作计划**

目前项目已暂停，准备期中汇报以及期末。期末完成后，下一步工作将围绕完成 ShuffleNetV1 的训练和部署展开：

1. **适配优化器设置**： 当前最关键的一步是修改 train.py 中优化器的参数来源 。 由于 mindcv 的 create\_model 直接返回了整个网络 net，我们需要确保优化器能够正确获 取 net 中所有可训练的参数（例如，使用 net.trainable\_params()），而不是像之前那样 仅优化 head\_net 的参数。

2. **数据预处理适配**：仔细核对 shufflenet\_v1\_g3\_20 模型所需的输入图像尺寸和归一化参 数，如有必要，修改 src/dataset.py 中的图像预处理逻辑，确保与模型要求一致。

3. **超参数审阅与调整**：重新审视 src/config.py 中的学习率、权重衰减等超参数，判断其 是否适合作为 ShuffleNetV1 微调的初始值，如有必要则进行调整。

4. **模型训练与监控**：在 CPU 环境下运行 python train.py 开始训练 ShuffleNetV1 模型。 密切监控训练过程中的损失值和准确率变化，如果训练不稳定或效果不佳，则返回调 整超参数。

5. **模型导出与转换**：将训练得到的最佳 ShuffleNetV1 模型权重导出为 shufflenet.mindir 文件，并使用 converter\_lite 转换为 shufflenet.ms 文件。

6. **端侧推理验证**：将 shufflenet.ms 模型部署到安卓 APP ，测试其在猫狗分类任务上的 实际表现。

7. **报告撰写与总结**：完成期中报告和后续的结题报告，总结项目经验和学习成果。

**（四）结题预期目标**

1. **功能完整性**：

o 成功在猫狗分类数据集上完成 MobileNetV2 模型的训练、转换和在安卓端的部 署与验证。 (已完成)

o 成功在猫狗分类数据集上完成 ShuffleNetV1 模型的训练（基于 mindcv 和预训 练权重进行微调）、转换和在安卓端的部署与验证。

2. **知识掌握与理解**：

o 能够清晰阐述图像分类的基本流程和关键技术点。

o 能够说明 MobileNetV2 和 ShuffleNetV1 的基本网络结构特点和设计思想。

o 能够解释 MindSpore 框架在模型训练中的主要步骤和 MindSpore Lite 在模型 部署中的作用。

o 能够总结在项目过程中遇到的主要问题及解决方法。

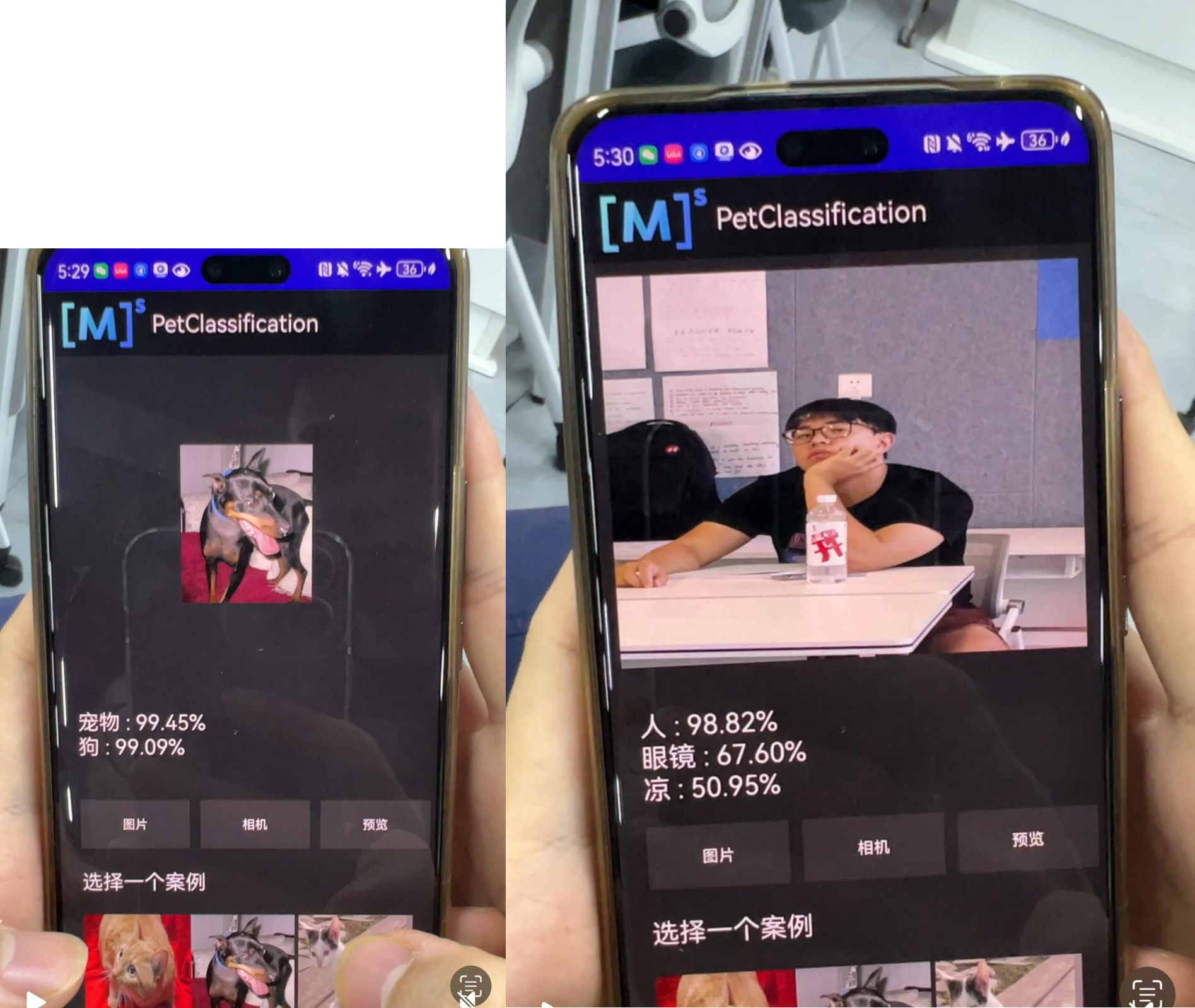
3. **实践能力**：

o 能够熟练配置 MindSpore 开发环境并运行所提供的项目代码。

o 能够根据项目需求，对现有代码（如模型定义、训练参数配置）进行修改和调 整。

o 能够独立完成模型的导出、转换和简单的端侧部署验证。

附录：附录：程序运行效果与代码展示：



新⽣项⽬

这些代码⽚段是基于项⽬指导书以及我们逐步的讨论。⼀个完整的可运⾏项⽬通常还 包含⼀些未在这⾥完整列出的辅助脚本和配置⽂件（例如 src/utils.py , src/config.py ,

 src/args.py , src/lr\_generator.py 等）

项⽬ ⼀ ：基于 **MindSpore Lite** 的猫狗分类 **(MobileNetV2)** 这部分代码主要来⾃您提供的项⽬指导书。

1. 数据清洗及数据集划分 (preprocessing\_dataset.py) (源⾃项⽬指导书 2.4.3 节代码解析)

**Python**

# 导⼊项⽬所需模块 import os

import sys

import zipfiIe

from PIL import Image import shutiI

# 设定数据集⽂件夹名称及需要的图⽚格式

\_DATASET\_NAME 〓 "PetImages" # [cite: 10]

IMG\_EXTENSIONS 〓 [I.jpg I, I.jpeg I, I.png I, I.ppmI, I.bmpI, I.tifI, I.tiffI] # [cite: 1 0]

# 判断该⽂件是否为图⽚

def is\_image\_fiIe(fiIename): # [cite: 10]

"""Judge whether it is a picture."""

return any(fiIename.Iower().endswith(extension) for extension in IMG\_EX TENSIONS) # [cite: 10]

# 删除图⽚或⽂件夹

def deIete\_fiIe(img\_path): # [cite: 10]

if os.path.isdir(img\_path): # [cite: 10]

os.rmdir(img\_path) # [cite: 10]

eIse:

os.remove(img\_path) # [cite: 10]

print(f"Invalid image file, delete {img\_path}") # [cite: 10]

# 判断是否为JPG格式图⽚

def is\_jpg(img\_path): # [cite: 10]

try:

i=Image.open(img\_path) # [cite: 10]

return i.format =='JPEG' # [cite: 10] except IOError: # [cite: 11]

return False # [cite: 11]

# 过滤数据集⽂件夹下的所有图⽚ ，如果是不需要的格式 ，则删除 def filter\_dataset(dataset\_path): # [cite: 11]

for sub\_dir in os.listdir(dataset\_path): # [cite: 11]

for file\_name in os.listdir(os.path.join(dataset\_path, sub\_dir)): # [cite: 1 1]

img\_path = os.path.join(os.path.join(dataset\_path, sub\_dir, file\_nam

e)) # [cite: 11]

if not (is\_image\_file(file\_name) and is\_jpg(img\_path)): # [cite: 11] delete\_file(img\_path) # [cite: 11]

continue # [cite: 11]

# 分割数据集 ，将上⼀步已经清洗⼲净的数据集进⾏分割 ，分割⽐例为训练集 ：

测试集9:1

def split\_dataset(dataset\_path, eval\_split=0.1): # [cite: 11]

os.makedirs(os.path.join(dataset\_path, "train"), exist\_ok=True) # [cite: 1 2]

os.makedirs(os.path.join(dataset\_path, "eval"), exist\_ok=True) # [cite: 12] for sub\_dir in os.listdir(dataset\_path): # [cite: 12]

if sub\_dir in ["train", "eval"]: # [cite: 12]

continue # [cite: 12]

cls\_list = os.listdir(os.path.join(dataset\_path, sub\_dir)) # [cite: 12] train\_size = int(len(cls\_list) \* (1 - eval\_split)) # [cite: 12]

os.makedirs(os.path.join(dataset\_path, "train", sub\_dir), exist\_ok=True) # [cite: 12]

os.makedirs(os.path.join(dataset\_path, "eval", sub\_dir), exist\_ok=True) # [cite: 12]

for i, file\_name in enumerate(os.listdir(os.path.join(dataset\_path, sub\_d ir))): # [cite: 12]

source\_file = os.path.join(dataset\_path, sub\_dir, file\_name) # [cite: 1 3]

if i < train\_size: # Adjusted index from <= to < for typical split

target\_file = os.path.join(dataset\_path, "train", sub\_dir, file\_name)

# [cite: 13]

else:

target\_file = os.path.join(dataset\_path, "eval", sub\_dir, file\_name) # [cite: 13]

shutil.move(source\_file, target\_file) # [cite: 13]

delete\_file(os.path.join(dataset\_path, sub\_dir)) # [cite: 14]

# 解压数据集 ，将zip⽂件解压

def extract\_dataset(zip\_file, save\_dir): # [cite: 14]

if not os.path.exists(zip\_file): # [cite: 14]

raise ValueError(f"{zip\_file} is not a valid zip file!") # [cite: 14] # Chang

ed to raise error try:

print("extract dataset") # [cite: 14]

zip\_f = zipfile.ZipFile(zip\_file) # Changed variable name from zip\_file to zip\_f

for names in zip\_f.namelist(): # [cite: 14]

zip\_f.extract(names, save\_dir) # [cite: 14]

zip\_f.close() # [cite: 14]

print(f"extract dataset at {os.path.join(save\_dir, \_DATASET\_NAME)}") # [cite: 14]

except Exception as e: # Changed to catch specific Exception

raise ValueError(f"Error extracting {zip\_file}: {e}") # [cite: 15] # Change d to raise error with more info

# main函数执⾏时 ，⾸先解压数据集 ，然后遍历解压后⽂件夹获取所有样本 ，进 ⾏数据清洗 ，然后分割为训练集和测试集 ，分别保存在对应的⽂件夹中

if \_\_name\_\_ == '\_\_main\_\_': # [cite: 15]

save\_dir = os.path.abspath("./dataset") # [cite: 15] if not os.path.isdir(save\_dir): # [cite: 15]

os.makedirs(save\_dir) # [cite: 15]

zip\_file\_path = sys.argv[1] # [cite: 15] # Renamed for clarity extract\_dataset(zip\_file\_path, save\_dir) # [cite: 15]

print("filter invaild images!") # [cite: 15]

dataset\_path = os.path.join(save\_dir, \_DATASET\_NAME) # [cite: 15] filter\_dataset(dataset\_path) # [cite: 15]

print("filter invaild images done, then split dataset to train and eval") # [ci te: 15]

split\_dataset(dataset\_path, eval\_split=0.1) # [cite: 15]

print(f"final dataset at {dataset\_path}") # [cite: 15]

2. ⽹络模型搭建及训练 (train.py 主要逻辑⽚段)

(源⾃项⽬指导书 2.4.4 节代码解析)

**Python**

# --- 步骤 1: 训练环境配置 (部分) --- import time

from mindspore import Tensor, nn # [cite: 20]

from mindspore.nn.optim.momentum import Momentum # [cite: 20]

from mindspore.nn.loss import SoftmaxCrossEntropyWithLogits # [cite: 20] from mindspore.common import set\_seed # [cite: 20]

# from src.dataset import extract\_features # [cite: 20] (Assuming this is in s rc/dataset.py)

# from src.lr\_generator import get\_lr # [cite: 21] (Assuming this is in src/lr\_g enerator.py)

# from src.config import set\_config # [cite: 21] (Assuming this is in src/conf ig.py)

# from src.args import train\_parse\_args # [cite: 21] (Assuming this is in src/ args.py)

# from src.utils import context\_device\_init, export\_mindir, predict\_from\_net, get\_samples\_from\_eval\_dataset # [cite: 21] (Assuming these are in src/utils. py)

# from src.models import CrossEntropyWithLabelSmooth, define\_net, load\_ ckpt, get\_networks, train # [cite: 21] (Assuming these are in src/models.py)

set\_seed(1) # [cite: 21]

if \_\_name\_\_ == '\_\_main\_\_': # [cite: 21]

#设定训练的相关参数 ，主要确定训练所在的硬件平台以及是否训练 # args\_opt = train\_parse\_args() # [cite: 21]

#确定在CPU进⾏训练后 ，设定训练⽹络的相关超参数 ，如 ：图像归⼀化尺

⼨ ，batchsize⼤⼩等

# config = set\_config(args\_opt) # [cite: 21]

start = time.time() # [cite: 21]

#初始化环境设置

# context\_device\_init(config) # [cite: 21]

# --- 步骤 2: 定义⽹络结构 ---

# backbone\_net, head\_net, net = define\_net(config, activation="Softma x") # [cite: 21]

# (注意: define\_net的具体实现 ，尤其是MobiIeNetV2版本 ，在指导书中未完 整给出 ，只说明了概念)

# (它会使⽤ MobiIeNetV2Backbone 和 MobiIeNetV2Head from .mobiIenet V2)

# --- 步骤 3: 读取CKPT参数 ---

# Ioad\_ckpt(backbone\_net, args\_opt.pretrain\_ckpt, trainabIe=FaIse) # [cit e: 21]

# --- 步骤 4: 配置⽹络超参 (部分) ---

# test\_Iist = get\_sampIes\_from\_evaI\_dataset(args\_opt.dataset\_path) # [cit e: 21]

# predict\_from\_net(net, test\_Iist, config, show\_titIe="pre training") # [cit

e: 21]

# data, step\_size = extract\_features(backbone\_net, args\_opt.dataset\_pat h, config) # [cite: 21]

# if config.IabeI\_smooth > 0: # [cite: 21]

# Ioss = CrossEntropyWithLabeISmooth(smooth\_factor=config.IabeI\_s mooth, num\_cIasses=config.num\_cIasses) # [cite: 22]

# eIse:

# Ioss = SoftmaxCrossEntropyWithLogits(sparse=True, reduction=Ime anI) # [cite: 22]

# Ir = Tensor(get\_Ir(gIobaI\_step=0, # [cite: 22]

# Ir\_init=config.Ir\_init, # [cite: 22]

# Ir\_end=config.Ir\_end, # [cite: 22]

# Ir\_max=config.Ir\_max, # [cite: 22]

# warmup\_epochs=config.warmup\_epochs, # [cite: 22]

# total\_epochs=config.epoch\_size, # [cite: 22]

# steps\_per\_epoch=step\_size)) # [cite: 22]

# opt = Momentum(filter(lambda x: x.requires\_grad, head\_net.get\_param eters()), lr, config.momentum, config.weight\_decay) # [cite: 22]

# --- 步骤 5: 训练 (部分) ---

# train\_net, eval\_net\_mindspore\_model = get\_networks(head\_net, loss, o pt) # Renamed eval\_net to avoid conflict with net variable

# train(train\_net, eval\_net\_mindspore\_model, net, data, config) # [cite: 2 2]

# print("train total cost {:5.4f} s".format(time.time() - start)) # [cite: 22]

# predict\_from\_net(net, test\_list, config, show\_title="after training") # [cit e: 22]

# --- 步骤 6: 导出⽹络模型 ---

# export\_mindir(net, "mobilenetv2") # [cite: 22]

pass # Placeholder as many parts are commented out for conceptual dis play

*(上述* *train.py 很多⾏被注释掉了，* *因为它们依赖于* *src ⽬录下的其他辅助函数和配*

*置，* *指导书中只给出了核⼼逻辑⽚段。完整的* *src/models.py (含MobileNetV2的*

*define\_net ) 在指导书中未直接提供，* *但其组件* *MobileNetV2Backbone 和* *MobileNetV2Head 被* *提及。)*

3. 模型转换命令

(源⾃项⽬指导书 2.4.5 节)

**Bash**

outputFile=pet # [cite: 23]

call converter\_lite --fmk=MINDIR --modelFile=d:\MindSporePetClassification\code\mobilenetv2.mindir --

4. 推理侧代码解析 (Android App C++ 部分⽚段) (源⾃项⽬指导书 2.4.7 节)

CMakeLists.txt 部分

**CMake**

# (部分 CMakeLists.txt 内容 ，指导书中分多张图⽚展⽰)

# cmake\_minimum\_required(VERSION 3.14.0) # [cite: 29]

# project(MindSpore) # [cite: 29]

# ... (其他配置) ...

# include\_directories(${CMAKE\_CURRENT\_SOURCE\_DIR}) # [cite: 30]

# include\_directories(${CMAKE\_CURRENT\_SOURCE\_DIR}/third\_party/securec/include) # [cite: 30]

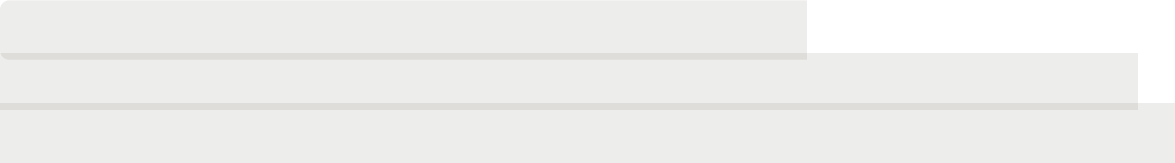
# include\_directories(${CMAKE\_CURRENT\_SOURCE\_DIR}/third\_party/flatbuffers/include) # [cite: 30] # ... (更多配置) ...

# add\_subdirectory(mindspore/ccsrc) # [cite: 32]

# add\_subdirectory(mindspore/core) # [cite: 32]

*C++ 推理逻辑⽚段*

**C++**



// --- 步骤 1: 加载模型⽂件 (JAVA层传递buffer) ---

// jlong bufferLen = env→GetDirectBufferCapacity(buffer); // [cite: 33] // char \*modelBuffer = CreateLocalModelBuffer(env, buffer); // [cite: 33]

// --- 创建会话 (部分) ---

// MSNetWork \*labelNet = new MSNetWork; // [cite: 34]

// mindspore::lite::Context \*context = new mindspore::lite::Context; // [cite: 34] // context→thread\_num\_ = num\_thread; // [cite: 34]

// labelNet→CreateSessionMS(modelBuffer, bufferLen, context); // [cite: 34]

// delete (context); // [cite: 35]

// context→device\_list\_[0].device\_info\_.cpu\_device\_info\_.cpu\_bind\_mode\_ = mindspore::lite::NO\_BIND; // [cite: 35]

// context→device\_list\_[0].device\_info\_.cpu\_device\_info\_.enable\_float16\_ = true; // [cite: 35]

// context→device\_list\_[0].device\_type\_ = mindspore::lite::DT\_CPU; // [cite: 35]

// --- 步骤 2: 加载模型⽂件并构建计算图 (MSNetWork::CreateSessionMS内部) ---

// void MSNetWork::CreateSessionMS(char\* modelBuffer, size\_t bufferLen, mindspore::lite::Context\* ctx) // Name argument removed as per snippet [cite: 36]

// {

// CreateSession(modelBuffer, bufferLen, ctx); // [cite: 36] (Assuming CreateSession is a member function) // session = mindspore::session::LiteSession::CreateSession(ctx); // [cite: 36]

// auto model = mindspore::lite::Model::Import(modelBuffer, bufferLen); // [cite: 37] // int ret = session→CompileGraph(model); // [cite: 37]

// }

// --- 步骤 2: 图⽚数据转为Tensor (部分) ---

// BitmapToMat(env, srcBitmap, matImageSrc); // [cite: 38]

// matImgPreprocessed = PreProcessImageData(matImageSrc); // [cite: 38] // ImgDims inputDims; // [cite: 39]

// inputDims.channel = matImgPreprocessed.channels(); // [cite: 39] // inputDims.width = matImgPreprocessed.cols; // [cite: 39]

// inputDims.height = matImgPreprocessed.rows; // [cite: 39]

// float \*dataHWC = new float[inputDims.channel \* inputDims.width \* inputDims.height]; // [cite: 39] // float \*ptrTmp = reinterpret\_cast<float \*>(matImgPreprocessed.data); // [cite: 40]

// for(int i = 0; i < inputDims.channel \* inputDims.width \* inputDims.height; i++){ // [cite: 41] // dataHWC[i] = ptrTmp[i]; // [cite: 41]

// }

// auto msInputs = mSession→GetInputs(); // [cite: 42]

// auto inTensor = msInputs.front(); // [cite: 42]

// memcpy(inTensor→MutableData(), dataHWC, // [cite: 43]

// inputDims.channel \* inputDims.width \* inputDims.height \* sizeof(float)); // [cite: 43] // delete[] (dataHWC); // [cite: 43]

// --- 步骤 3: 输⼊数据预处理 (PreProcessImageData 函数) ---

// bool PreProcessImageData(const LiteMat &lite\_mat\_bgr, LiteMat \*lite\_norm\_mat\_ptr) { // [cite: 44] // bool ret = false; // [cite: 44]

// LiteMat lite\_mat\_resize; // [cite: 44]

// LiteMat &lite\_norm\_mat\_cut = \*lite\_norm\_mat\_ptr; // [cite: 44]

// ret = ResizeBilinear(lite\_mat\_bgr, lite\_mat\_resize, 256, 256); // [cite: 45] // if (!ret) { MS\_PRINT("ResizeBilinear error"); return false; } // [cite: 45] // LiteMat lite\_mat\_convert\_float; // [cite: 46]

// ret = ConvertTo(lite\_mat\_resize, lite\_mat\_convert\_float, 1.0 / 255.0); // [cite: 46] // if (!ret) { MS\_PRINT("ConvertTo error"); return false; } // [cite: 47]

// LiteMat lite\_mat\_cut; // [cite: 47]

// ret = Crop(lite\_mat\_convert\_float, lite\_mat\_cut, 16, 16, 224, 224); // [cite: 48] // if (!ret) { MS\_PRINT("Crop error"); return false; } // [cite: 48]

// float means[3] = {0.485, 0.456, 0.406}; // [cite: 49]

// float vars[3] = {1.0 / 0.229, 1.0 / 0.224, 1.0 / 0.225}; // [cite: 50]

// SubStractMeanNormalize(lite\_mat\_cut, lite\_norm\_mat\_cut, means, vars); // [cite: 50] // return true; // [cite: 50]

// }

// auto status = mSession→RunGraph(); // [cite: 51]

// --- 步骤 3: 图执⾏ ，端侧推理 ---

// --- 步骤 4: 获取输出数据 ---

// auto names = mSession→GetOutputTensorNames(); // [cite: 52]

// std::unordered\_map<std::string, mindspore::tensor::MSTensor \*> msOutputs; // [cite: 52] // for (const auto &name : names) { // [cite: 52]

// auto temp\_dat =mSession→GetOutputByTensorName(name); // [cite: 52]

// msOutputs.insert(std::pair<std::string, mindspore::tensor::MSTensor \*> {name, temp\_dat}); // [cite: 53] // }

// std::string retStr = ProcessRunnetResult(msOutputs); // ret argument removed as per snippet [cite: 53]

// --- 步骤 5: 输出数据的后续处理 (ProcessRunnetResult 函数) ---

54]

// std::unordered\_map<std::string, mindspore::tensor::MSTensor \*> msOutputs) { // [cite: 54]

// std::unordered\_map<std::string, mindspore::tensor::MSTensor \*>::iterator iter; // [cite: 55] // iter = msOutputs.begin(); // [cite: 55]

// std::string ProcessRunnetResult(const int RET\_CATEGORY\_SUM, const char \*const labels\_name\_map[], // [cite:

// auto outputTensor = iter→second; // [cite: 56]

// int tensorNum = outputTensor→ElementsNum(); // [cite: 56]

// float \*temp\_scores = static\_cast<float \*>(outputTensor→MutableData()); // [cite: 57] // // float scores[RET\_CATEGORY\_SUM]; // [cite: 57] // Declaration

// // for (int i = 0; i < RET\_CATEGORY\_SUM; ++i) { scores[i] = temp\_scores[i]; } // [cite: 58] // // ... (score normalization logic) ... // [cite: 59, 60, 61, 62, 63, 64]

// std::string categoryScore = ""; // [cite: 66]

// // for (int i = 0; i < RET\_CATEGORY\_SUM; ++i) { // [cite: 67] // // categoryScore += labels\_name\_map[i]; // [cite: 67] // // categoryScore += ":"; // [cite: 68]

// // std::string score\_str = std::to\_string(scores[i]); // [cite: 68] // // categoryScore += score\_str; // [cite: 68]

// // categoryScore += ";"; // [cite: 68] // // }

// return categoryScore; // [cite: 69] // }

项⽬⼆ ：**ShuffleNet** 实现

这部分代码是基于组员之前的讨论 ，已经完成的 mindcv 安装和初步代码修改。

1. 使⽤ MindCV 的 src/models.py (核⼼ define\_net 函数)

(基于之前讨论的修改版本)

**Python**

# 在 models.py 顶部

from mindcv.models import create\_model

import mindspore.nn as nn # For nn.Dense if needed, or other nn compone nts

from mindspore import Tensor

from mindspore.ops import operations as P

# ... (其他必要的原始导⼊ ，如 \_Loss, Callback, load\_checkpoint等可以保留， 如果仍然需要)

# from .mobilenetV2 import MobileNetV2Backbone, MobileNetV2Head, mo bilenet\_v2 # 可除

# CrossEntropyWithLabelSmooth, Monitor, acc, load\_ckpt, get\_networks, tr ain 函数可以保留原样

# ... (此处省略了那些未改变的辅助类和函数定义 ，如CrossEntropyWithLabelS mooth等 ，假设它们不变) ...

def define\_net(config, activation="None"): # activation 参数对于mindcv可能 不再直接使⽤

print("Using backbone: ShuffleNetV1 from mindcv")

# 确认的模型名称 (2.0x size, group 3)

model\_name = "shufflenet\_v1\_g3\_20"

print(f"Creating MindCV model: {model\_ name}")

# 使⽤ mindcv 创建模型:

# pretrained=True 会⾃动下载并加载预训练权重

# num\_classes=config.num\_classes (猫狗分类是 2) 会让 mindcv ⾃动替换 最后分类层

net = create\_model(model\_name,

pretrained=True,

num\_classes=config.num\_classes) # 确保 config.num\_classe s 是 2

# --- 重要提⽰ ：适配后续代码 ---

# mindcv 的 create\_model 直接返回了完整的、可⽤于微调的⽹络 net

# 但原来的代码？ ？分别使⽤了 backbone\_net (⽤于加载权重) 和 head\_net (⽤于优化器)？

# 为了最⼤程度减少对后续代码的修改 ：

# 注意 ：这假设后续优化器能直接使⽤ net.trainable\_params() 获取所有可训 练参数

print("NOTE: Returning 'net' object also as backbone\_net and head\_net f or compatibility.")

print(" Ensure optimizer setup later uses net.trainable\_params() or sim ilar.")

backbone\_net = net # 临时兼容处理 head\_net = net # 临时兼容处理

# 保持原来的返回结构

return backbone\_net, head\_net, net

# ... (train, get\_networks等函数定义保持不变 ，但其内部逻辑可能需要适配新的 net结构)

2. train.py 的主要修改 (针对 MindCV)

主要的修改是移除⼿动加载预训练权重的代码。 **Python**

# 在 train.py 中 ，找到类似如下的代码⾏并删除或注释掉 ：

# load\_ckpt(backbone\_net, args\_opt.pretrain\_ckpt, trainable=False)