简 表

硬件环境（CPU/GPU）：
CPU: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
GPU: NVIDIA GeForce MX110

操作系统：Windows
版本：Windows 10 专业版
版本号：21H2
操作系统内部版本：19044.2006
体验：Windows Feature Experience Pack 120.2212.4180.0

采用的深度学习框架、工具、语言：
框架：pytorch
工具：pycharm、tensorboard、anaconda
语言：python

任务描述/问题定义：（限 200 字）
对图片进行分类，图片种类有五种，分别为雏菊、蒲公英、玫瑰、向日葵和郁金香，模型通过对共计三千余张的五类图片数据进行学习。学习后，可输入一张图片，经过模型运算，输出其属于五类花的概率值，并选出最大值作为预测结果。

数据集及来源（相关链接）：

https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz

采用的深度学习模型：	模型提出的年份及发表的会议/期刊：
MobileViT	于 2022 年发表于 ICLR 会议

最终设置的超参数（如 Learning rate, Batch size 等）：
Learning rate = 0.0002
Batch size = 8
num_heads =8 (多头注意力的头数)
cls_dropout = 0.1(dopout 的失活概率)
Epoch=100(迭代次数)

模型的效果（如准确率等与任务相关的评价指标）：

最终在测试集上的准确率为 87.7%

MobileVit 模型结构简介

MobileVit 模型是对 Vision Transformer 和 CNN 两种模型进行了混合使用，并参考了 MobileNetV2 的设计优点。这样的设计一定程度上减轻了 Vision Transformer 中参数多，算力要求高，缺少归纳偏执和迁移到其他任务比较繁琐等问题，是 Vision Transformer 模型应用在移动端设备的。

下面我会对该模型的结构进行介绍。这里需要说明的是，模型的复现与 MobileVit 论文(https://arxiv.org/abs/2110.02178)的编码实现有一些差别。

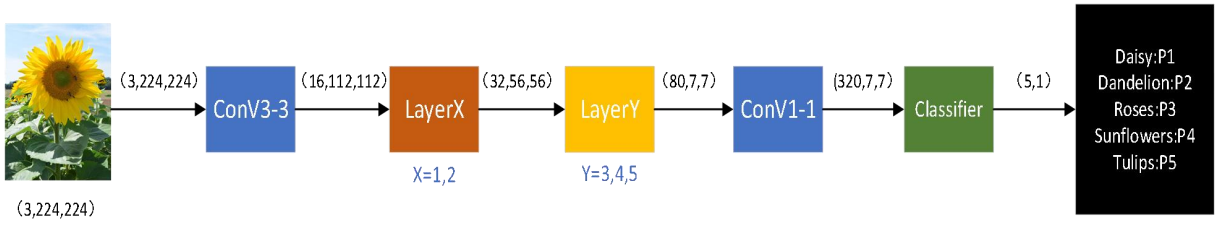

图 1　MobileVit 模型整体架构图

上图为 MobileVit 的整体架构图，图中 ConV3-3 和 Conv1-1 分别对应卷积核大小为 1x1 和 3x3 的卷积操作，LayerX 和 LayerY 包含了一系列的卷积和 transformer 操作，具体结构将会在后面进行介绍。Classifier 包含了全局池化操作和全连接操作。对于经过预处理的分辨率为 224x224 的三通道彩色图片，其经历一些列的卷积和 transformer 操作后，经过全连接和 softmax 后会得到 5 个种类花的概率值的映射关系。

其中 LayerX 包括 Layer1 和 Layer2 两个部分，Layer1 中含有一个 MV2 模块，Layer2 中含有两个 MV2 模块，需要注意的是，Layer2 中的第一个 MV2 模块**不含有残差连接**。这里说明一下，含有残差连接的模块输入输出不会有分辨率的变化。因为单从图像矩阵大小的角度来看，3x3 的卷积核对图像的下采样操作，会由于残差连接操作中和卷积前图像矩阵的加法操作而恢复原大小。而和原图像不含残差连接的输出分辨率会变为原来的四分之一。通道数也会相应增长。其中 MV2 的结构如图 2 所示。

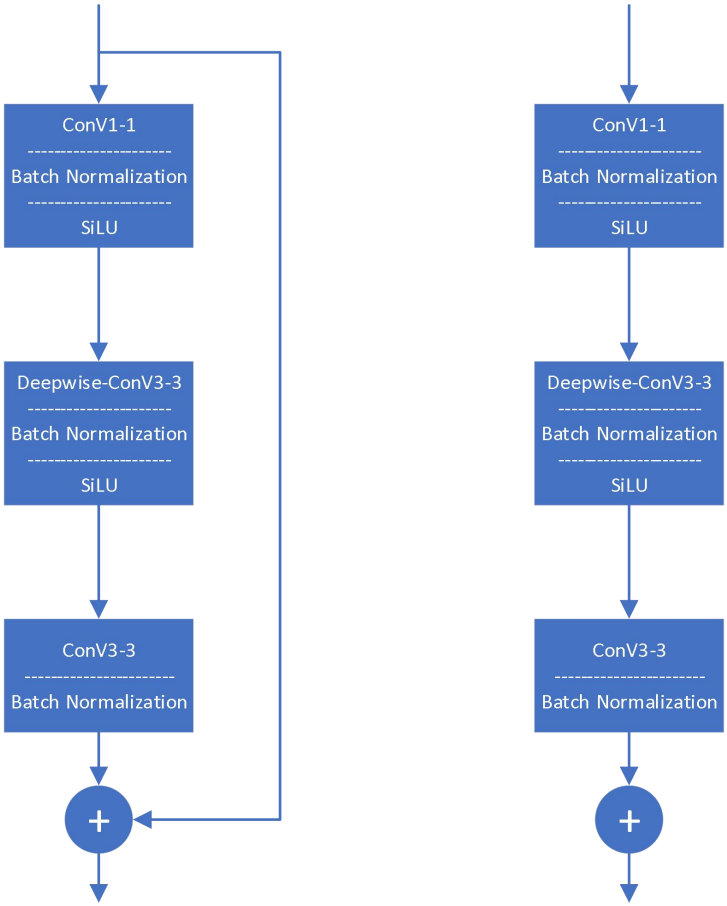

图 2　MV2 模块结构图（右侧为无残差结构的 MV2）

LayerY 包含 Layer3、Layer4 和 Layer5 三个层，这三个层的结构完全相同，都是由一个**无残差连接的** MV2 模块和一个 MobileViT block 模块组成。MV2 模块在前 MobileViT block 模块在后，下面将对 MobileVit 模块进行拆解，首先看一下 MobileViT block 模块的整体架构图 3（该结构图采用的是论文中的图片 https://arxiv.org/abs/2110.02178）。

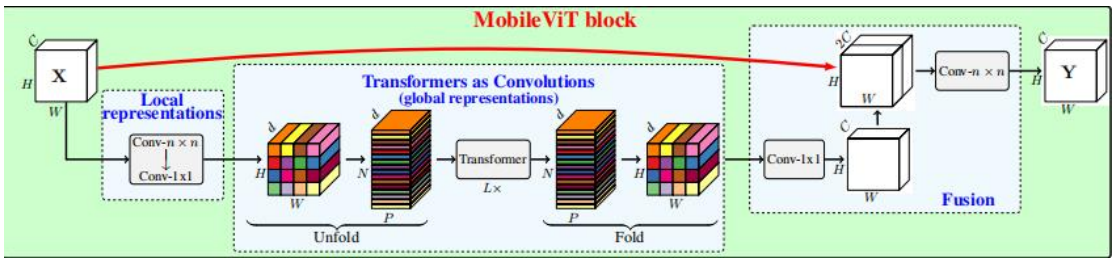

图 3　MV2 模块结构图（右侧为无残差结构的 MV2）

首先对图片中的参数进行解释，在源码实现中，n-n 的卷积核都采用的 3-3

的卷积核。全局表征中的 Transformer 中的 L 为 2。MobileViT 中与 Vision Transformer 中的一大不同点是，取消了 Vision Transformer 中的位置编码，而是采用 CNN 具有空间归纳偏执的特点予以替代，并在使用多头多头自注意机制时，并没有将每个像素空间和其与像素空间进行计算，而是采用了先通过卷积进行局部表征，之后再将每个 Patch 进行展平，这样每个 Patch 只有相同位置的像素空间会进行多头自注意力操作，并且由于之前做过局部表征，不会对全局相关性产生影响。这样的操作减少了大量参数，并且在 GPU 并行流水线充足的情况下能够有更快的推理速度。

对于 Transformer 的实现也与论文中有所不同，下面给出源码中 Transformer 的结构图 4。

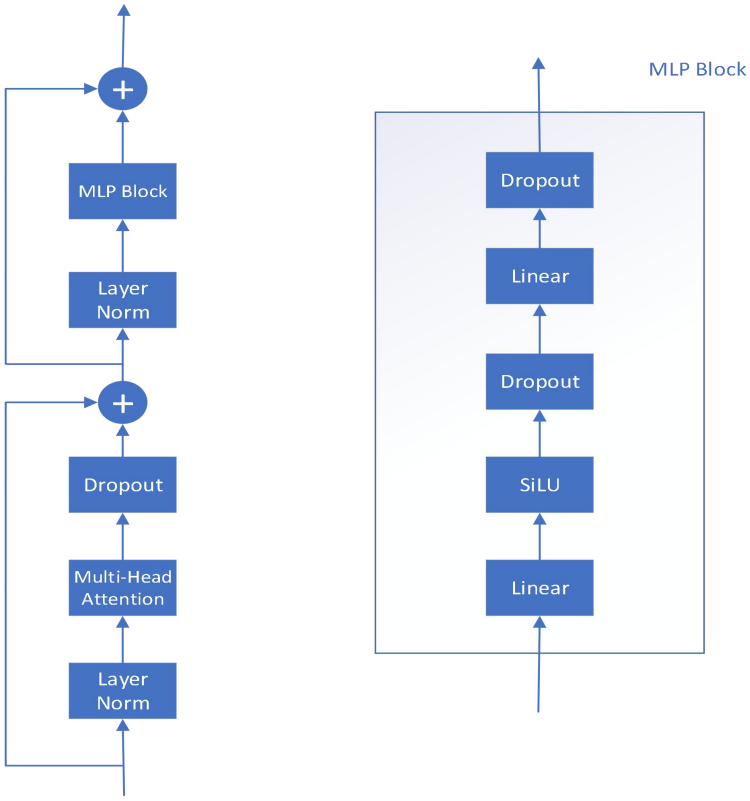

图 4　Transformer 模块结构图

数据集采用的是包含五类花图片的花数据集。由于电脑配置较低，这个数据集更适合实验的进行，其中训练集包括 2939 张图片，测试集包括 731 张图片。对于损失函数，采用的是交叉熵函数。优化器采用的是 Adam 和 weight decay 相结合的 AdamW。

训练曲线（Loss-Epoch）和运行截图如下

由于实验电脑的配置太低，所以没有选用不同分辨率大小、Batchsize 和不同 Pathsize 进行测验，仅改变了不同的迭代次数。不同的迭代次数下训练集和测试集的训练曲线如下图所示。

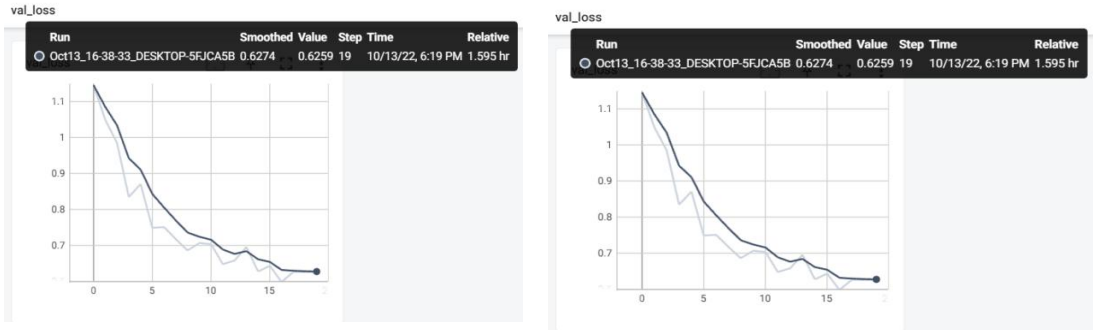

图 5　20 次迭代时训练集和测试集的 loss 曲线

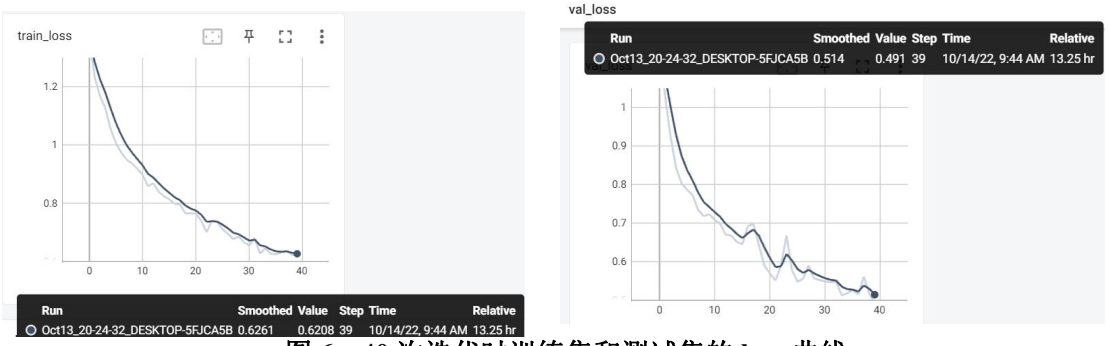

图 6　40 次迭代时训练集和测试集的 loss 曲线

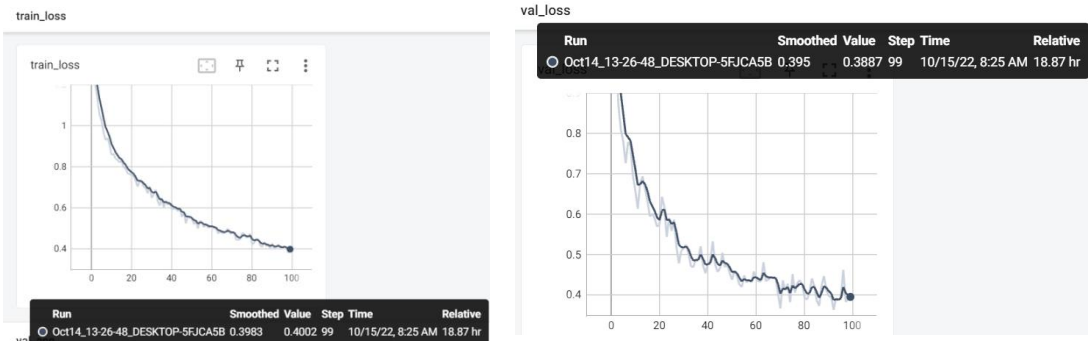

图 7　100 次迭代时训练集和测试集的 loss 曲线

可见当迭代次数达到 100 是训练集和测试集的 loss 曲线趋于平稳。故选用迭代次数 100 时的权重对其进行分类任务。此时，模型的测试集准确率如图 8 所示。

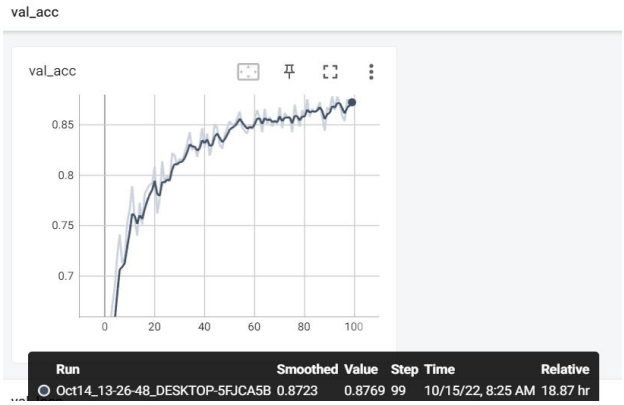

图 8　100 次迭代时训练集准确率

为了更加直观的体现模型的效果，从百度上随机下载了一张玫瑰花照片和一张向日葵的照片，并使用模型进行预测，其分类结果如图 9 和 10 所示。

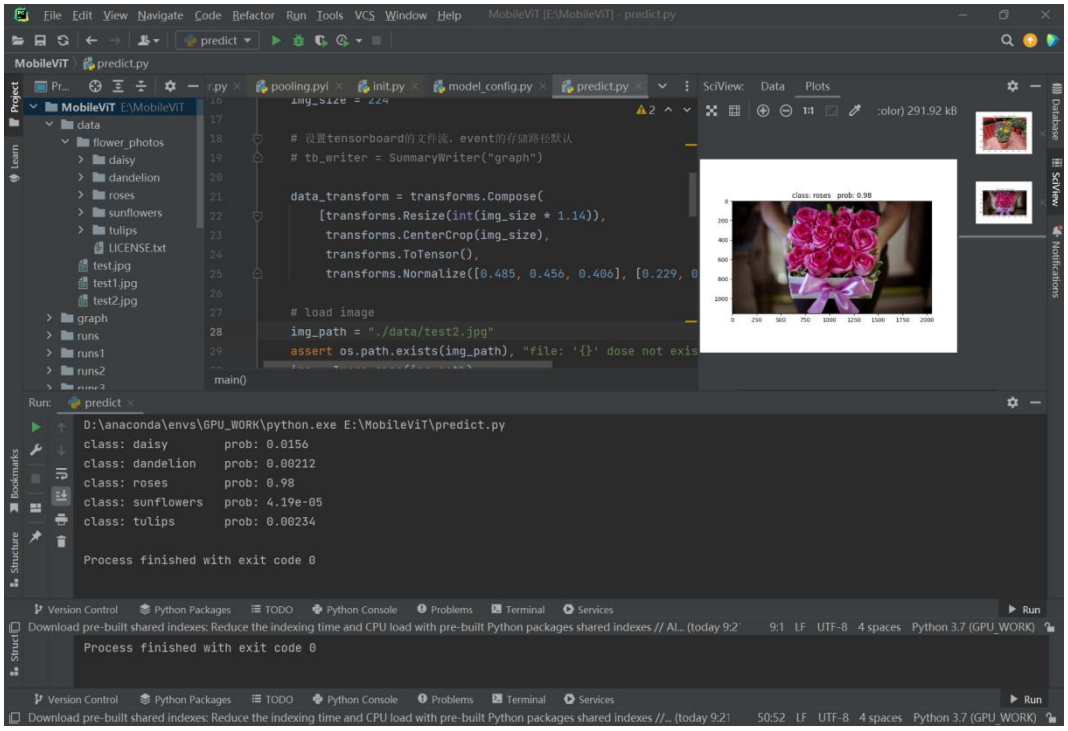

图 9　模型预测结果图

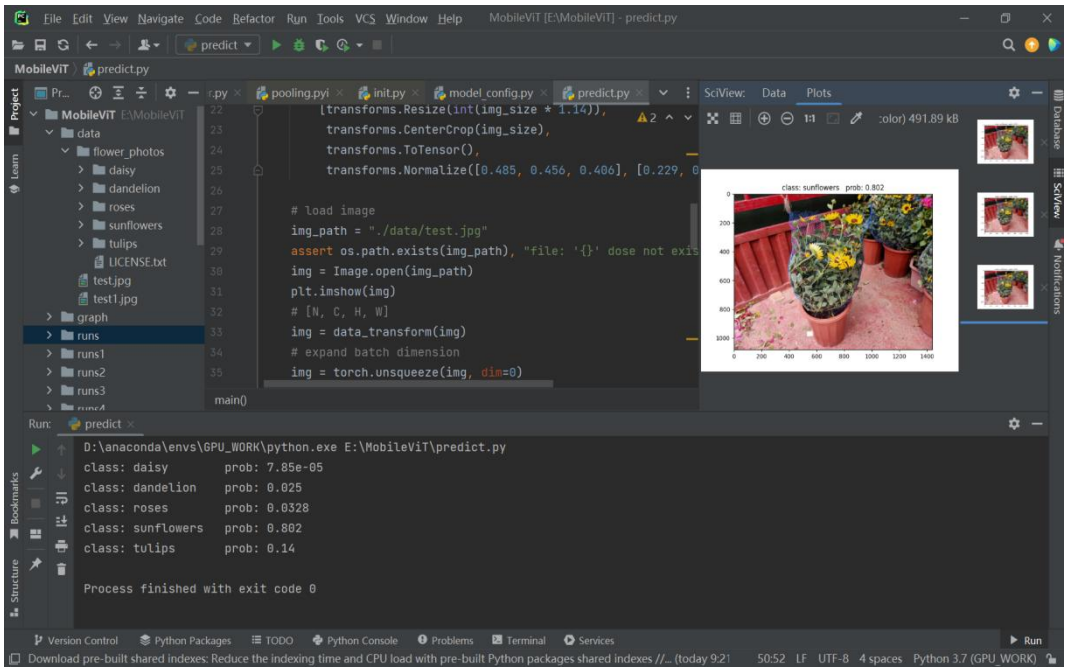

图 10　模型预测结果图 2