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：  MobileViT | 模型提出的年份及发表的会议/期刊：  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。

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）。

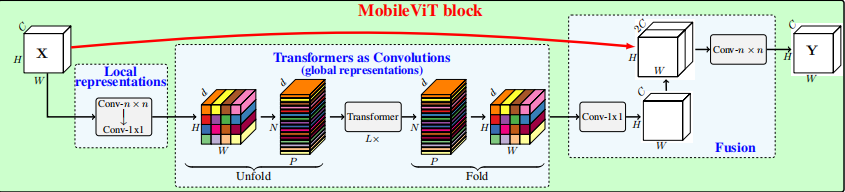


图3 MV2模块结构图（右侧为无残差结构的MV2）

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2.3 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。