

计算机视觉期末作业

王逸群 19307110397 程子琛 19307110417

2022.5-31

GitHub repo 链接: <https://github.com/quniLcs/cv-final>

网盘链接: [百度网盘计算机视觉期末作业](#)

1 语义分割

1.1 模型

基于[GitHub 上的 DeeplabV3实现](#).

该模型采用[deeplabv3](#)算法模型进行识别预测，调用[PixelLib](#)库实现驾驶视频的语义分割。

1.2 测试

通过 pip 安装 tensorflow、imgaug 和 pixellib 库。

从 pixellib 模块中导入语义分割类，并将其实例化，调用函数加载预训练模型。

测试中，将视频拆分成帧，对图像进行语义分割，设置参数，输出图像 15 frames/s，合并为视频，结果见[百度网盘链接](#)中语义分割文件夹。

2 目标检测

2.1 数据集

本项目基于 VOC 数据集进行训练和测试，其中，前两个任务中的 Faster R-CNN 网络使用 VOC2007 的训练（验证）和测试集，Mask R-CNN 预训练的模型使用 VOC2012 数据集进行训练和验证。

数据集包括四大类，细分至 20 小类，VOC2007 数据集 train/validation/test 共有 9963 张图片，包含 24640 个已标注的目标 objects；VOC2012 用于检测的数据集规模为：train/val : 11540 张图片，包含 27450 个已被标注的 ROI annotated objects。

2.2 网络结构

2.3 超参数设置

2.3.1 随机初始化训练

参数初始化：随机初始化 vgg16；

学习率：0.001；

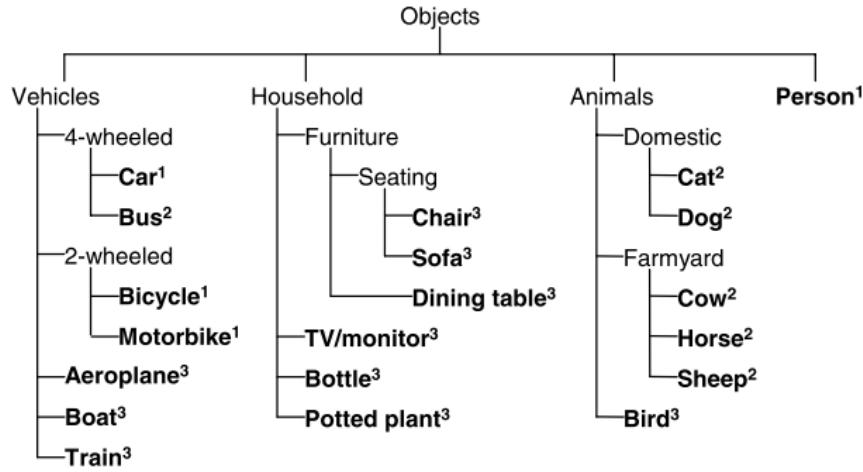


图 1: VOC 数据集分类

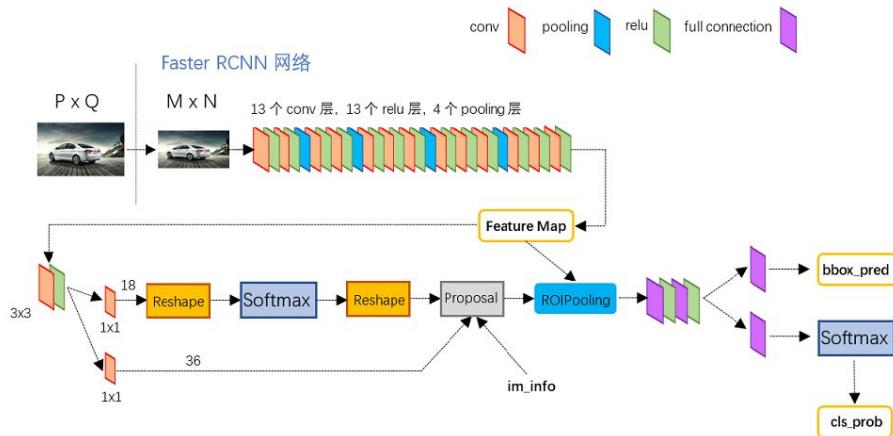


图 2: Faster R-CNN 网络架构

优化器: 带有 0.9 动量的随机梯度下降算法;

正则化参数: 0.0005;

回合数: 60;

批量大小: 1;

总循环数: $60 \times 5011 = 300660$;

损失函数: 交叉熵损失函数;

评价指标: mAP & loss。

2.3.2 ImageNet 预训练

参数初始化: ImageNet 预训练 vgg16;

学习率: 0.001, 15 个 epoch 后 decay 为 0.0001;

优化器: 带有 0.9 动量的随机梯度下降算法;

正则化参数: 0.0005;

回合数: 30;

批量大小: 1;

总循环数: $30 \times 5011 = 150330$;

损失函数：交叉熵损失函数；

评价指标：mAP & loss。

2.3.3 COCO 预训练 Mask R-CNN

参数初始化：COCO 预训练 Mask R-CNN backbone 和随机初始化 head；

学习率：0.01，每 3 个 epoch 后 decay $\frac{1}{3}$ ；

优化器：带有 0.9 动量的随机梯度下降算法；

正则化参数：0.0001；

回合数：22；

批量大小：4；

总循环数： $22 \times 1429 \div 4 = 7860$ ；

损失函数：交叉熵损失函数；

评价指标：mAP & loss。

2.4 训练

2.4.1 随机初始化训练

基于[GitHub 上的 Simple Faster-RCNN 实现](#)。

使用 VOC2007 数据集进行训练和测试，将数据集解压至 VOCdevkit 文件夹。

修改代码文件：`utils/config.py` 修改参数 `voc_data_dir` 为数据集对应路径；修改预训练模型路径以加载随机初始化参数。

`model/faster_rcnn_vgg16.py` 修改 line20 为 `model = vgg16(pretrained=False)`

源代码可视化部分基于 `visdom` 实现，对 `trainer.py` 文件 `train_step` 函数做调整，以输出训练过程 loss 曲线。

开始训练：运行 `model_train.ipynb` 中 `train()` 函数。

训练可视化：训练日志记录各批次的训练误差，通过 `Tensorboard` 可视化结果如下。初始训练 30 个 epoch，但结果不理想，因此继续训练 30 个 epoch。

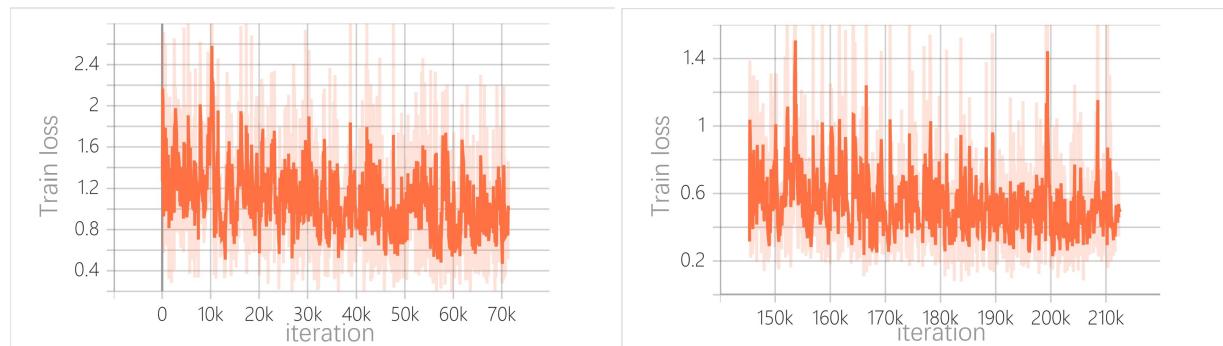


图 3：随机初始化训练误差

2.4.2 ImageNet 初始化训练

`model/faster_rcnn_vgg16.py` 修改 line20 为 `model = vgg16(pretrained=True)`

其他训练过程同上。

训练可视化：训练日志记录各批次的训练误差，通过 `Tensorboard` 可视化结果如下。

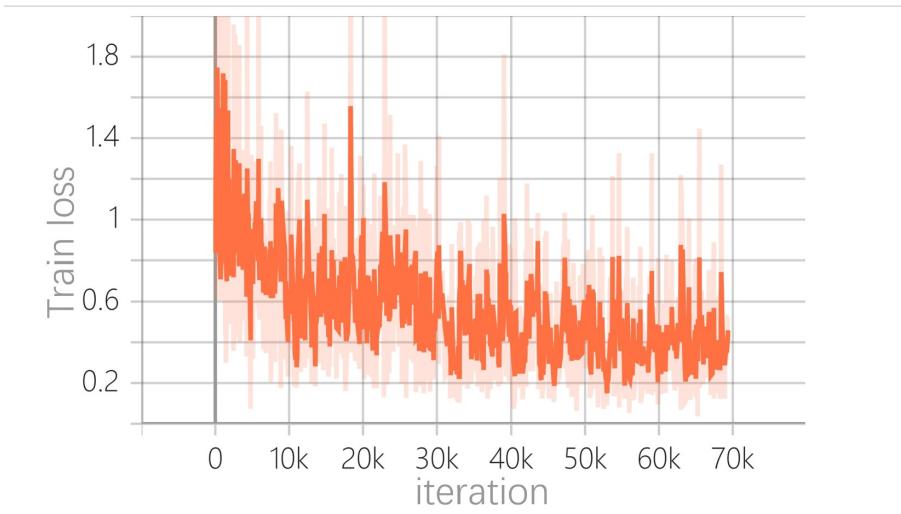


图 4: ImageNet 初始化训练误差

2.4.3 Mask R-CNN 初始化训练

基于[GitHub 上的 Faster-RCNN](#)实现。

使用 VOC2012 数据集进行训练和测试，将数据集解压至VOCdevkit文件夹。

预训练参数设置：运行参数读取.ipynb修改参数，Faster R-CNN 模型的 backbone 参数部分替换为 COCO 数据集上预训练的 Mask R-CNN 模型的 backbone 参数，head 部分使用 numpy 进行随机初始化。

开始训练：运行`python train_res50_fpn.py`命令。

训练可视化：训练日志记录各批次的训练误差，通过Tensorboard可视化结果如下。

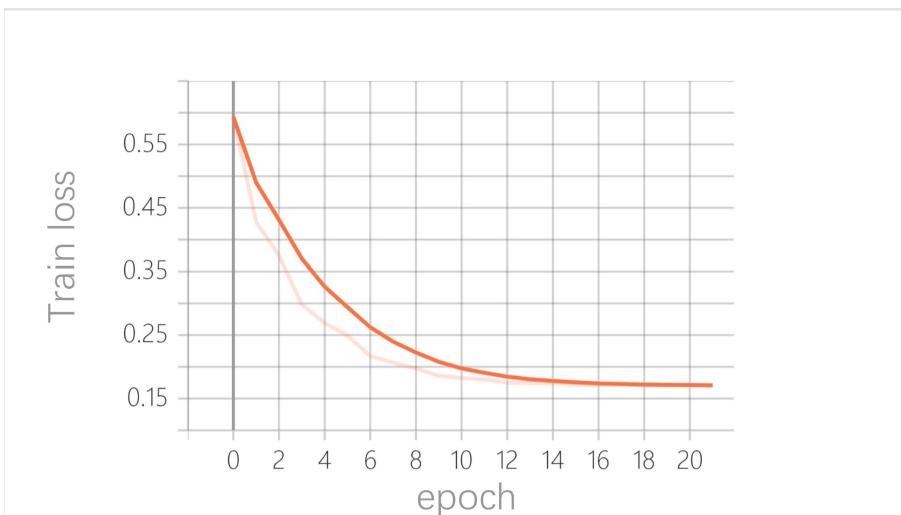


图 5: Mask R-CNN 初始化训练误差

2.5 测试

2.5.1 测试图像

以下为不在 VOC 数据集内，但包含有 VOC 中类别物体的图像检测结果。

运行model_test.ipynb使用最佳模型对图片进行测试。

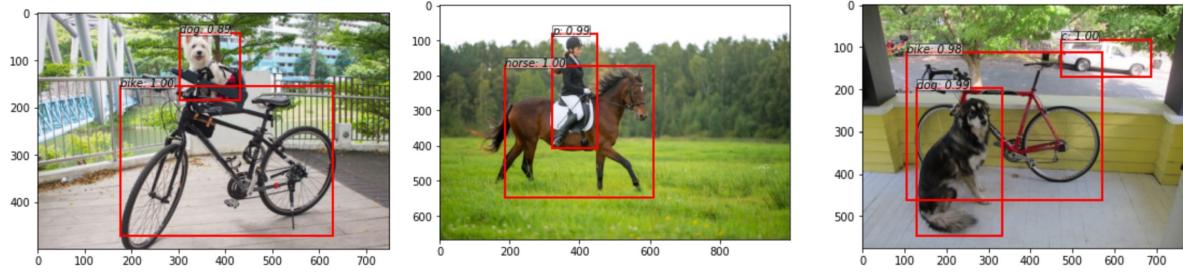


图 6: Faster R-CNN 测试结果

2.5.2 随机初始化训练

利用Tensorboard可视化测试 mAP 曲线如下。在第 60 个 epoch 处，mAP 仍呈上升趋势，但因为参数为随机初始化，mAP 值较低，考虑到现实情况，没有继续训练。

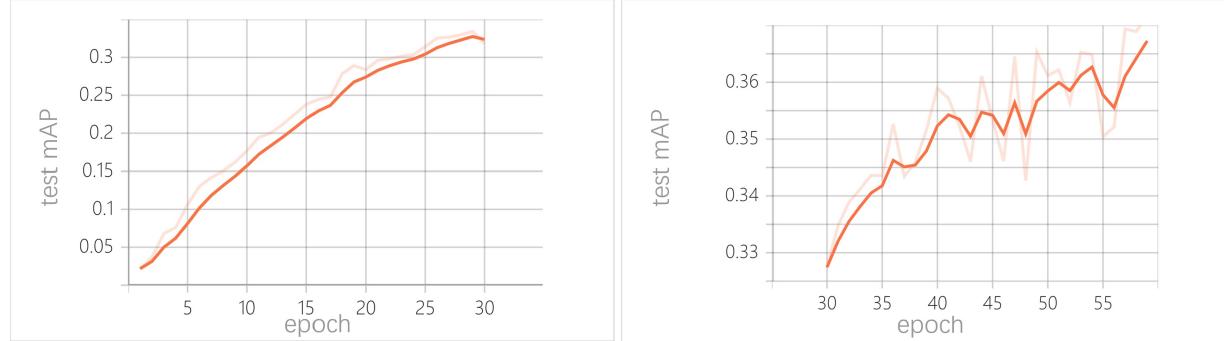


图 7: 随机初始化测试 mAP

测试损失函数曲线如下，出现过拟合情况，但从模型在测试集的 mAP 表现来看，训练效果对模型改进仍比较显著。

2.5.3 ImageNet 初始化训练

利用Tensorboard可视化测试 mAP 曲线如下。第 15 个 epoch 处，学习率衰减，mAP 有明显提升。

测试损失函数曲线如下，出现过拟合情况，但从模型在测试集的 mAP 表现来看，训练效果对模型改进仍比较显著。采用 early-stopping 选择最优模型，mAP 达到 0.7034.

2.5.4 Mask R-CNN 初始训练

利用Tensorboard可视化测试 mAP 曲线如下。第 22 个 epoch 处，mAP 变化趋于平稳，最高为 0.7643。

测试损失函数曲线如下。

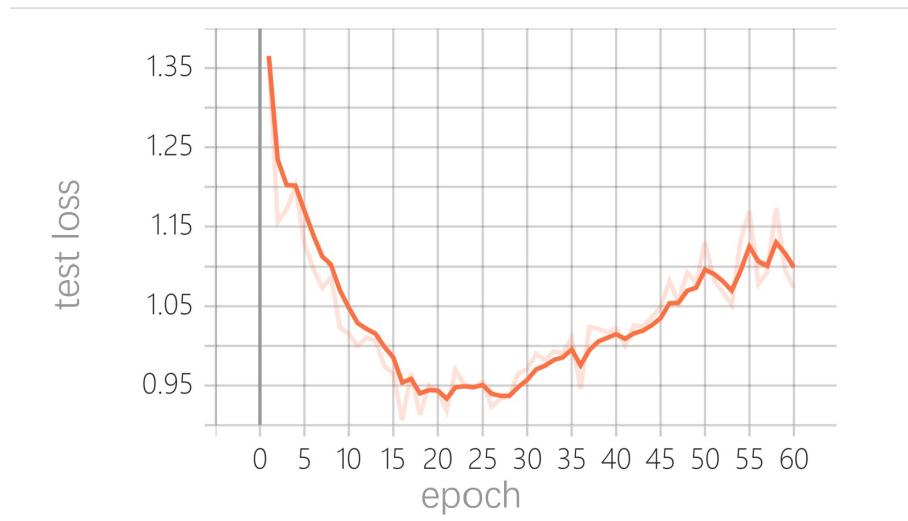


图 8: 随机初始化测试损失函数曲线

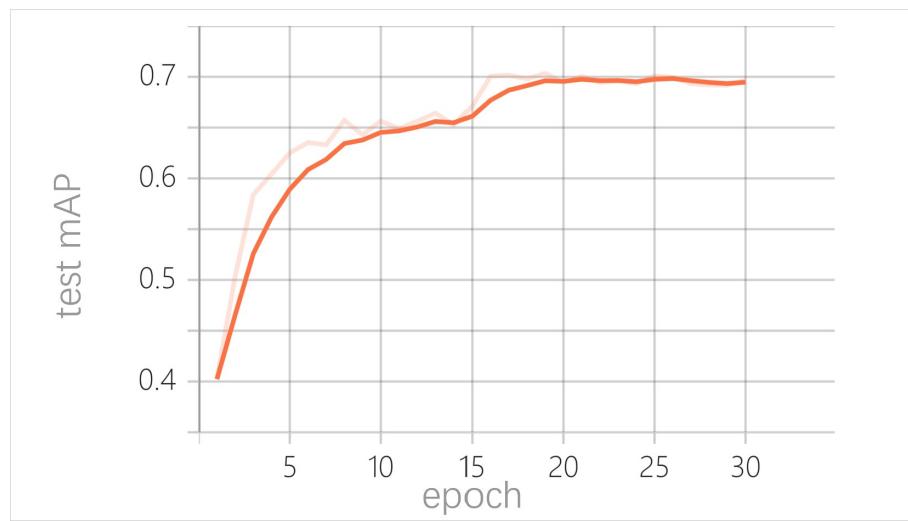


图 9: ImageNet 初始化测试 mAP

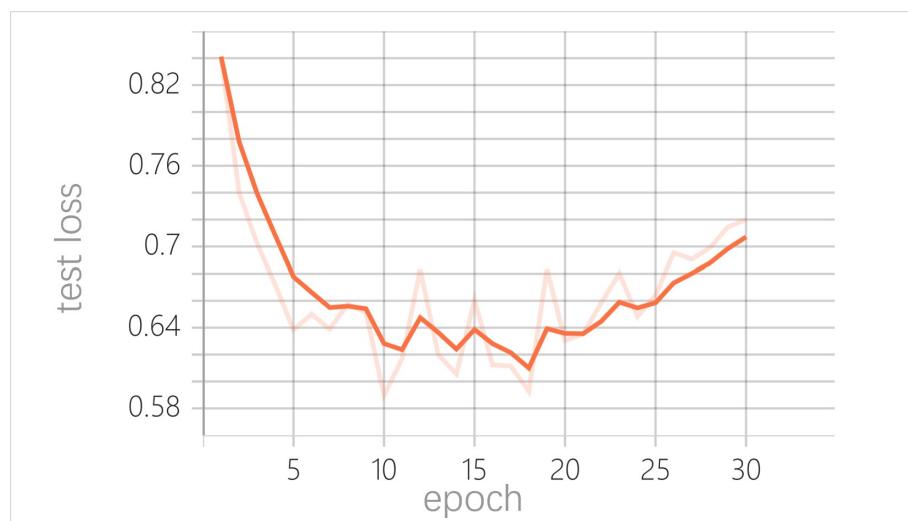


图 10: ImageNet 测试损失函数曲线

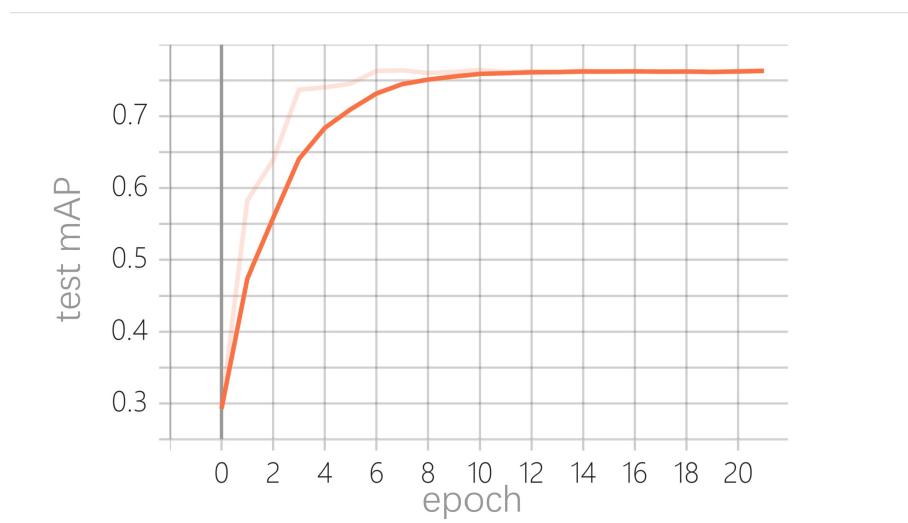


图 11: Mask R-CNN 初始化测试 mAP

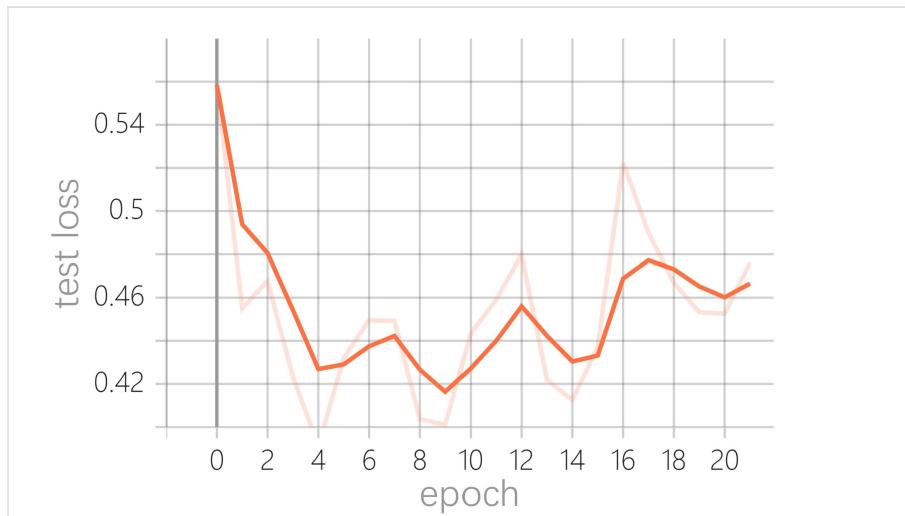


图 12: Mask R-CNN 初始化测试损失函数曲线

3 图像分类

3.1 数据集

本项目使用 CIFAR-100 数据集，其中包含 60000 张 32×32 的彩色图片，其中训练集 50000 张，测试集 10000 张，被平均分为 100 类。

3.2 网络结构

3.2.1 ResNet

本项目使用的第一种网络结构是 ResNet-18，其中激活函数为 ReLU，最大的特征为残差连接。后者包括两种单元结构如图13所示。

对于输入的图像，先进行步长为 1 的 $3 \times 64 \times 3 \times 3$ 卷积操作，并进行批归一化和激活，维度变为 $64 \times 32 \times 32$ ；接着通过两次第一种单元结构，维度不变；再通过第二种单元结构，维度变为 $128 \times 16 \times 16$ ；再通过第一种单元结构，维度不变；再通过第二种单元结构，维度变为 $256 \times 8 \times 8$ ；

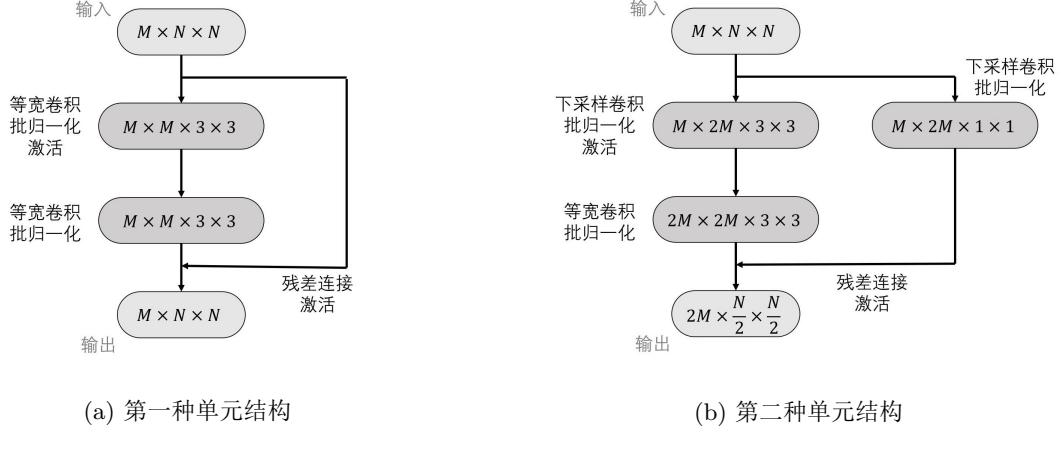


图 13: 残差连接的两种单元结构

再通过第一种单元结构，维度不变；再通过第二种单元结构，维度变为 $512 \times 4 \times 4$ ；再通过第一种单元结构，维度不变；最后通过全连接得到输出。由此，参数数量为 11220132.

3.2.2 ViT

本项目使用的第二种网络结构是 ViT(Vision Transformer)，其主体是一个 Transformer 编码器，其中激活函数为 GELU，如图14所示。

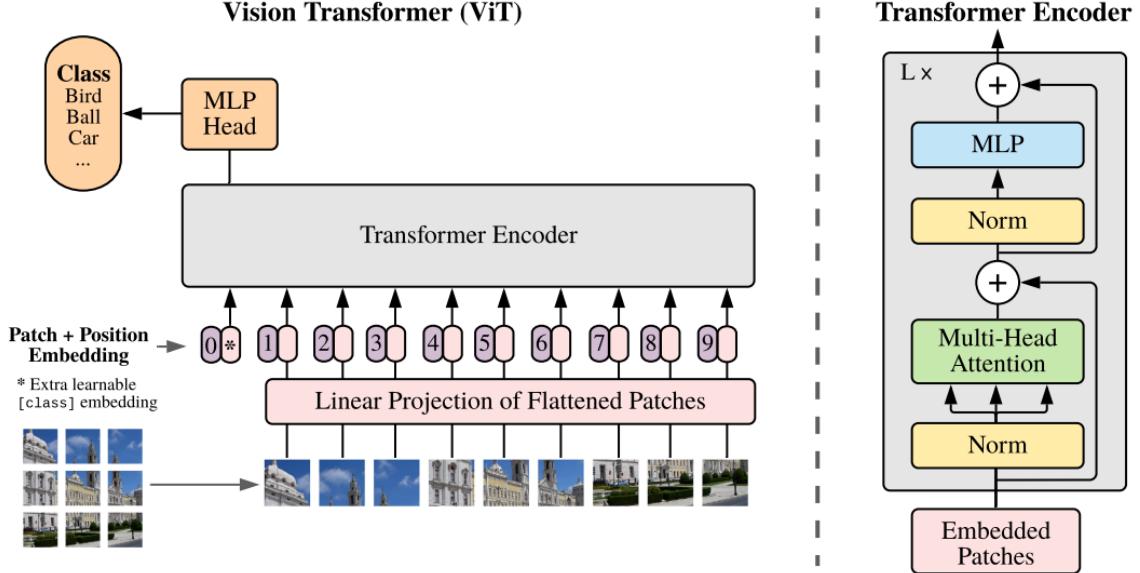


图 14: ViT 网络结构

对于输入的图像，根据给定的 patch size 划分成若干个 patch，并通过全连接层编码成给定维度的向量，即 patch 编码。这些 patch 编码和可学习的分类编码分别与各自的位置编码相加后进入 Transformer 编码器。在 Transformer 编码器中，每个 Transformer 层由层归一化、多头注意力机制、残差连接、层归一化、两层前馈神经网络、残差连接所组成。最后，分类编码对应的 Transformer 编码通过层归一化和全连接层得到输出。

参考 ViT 原论文、与 ViT 相关的 Jean-Baptiste Cordonnier 等人的工作、以及官方开源软件包中模型的默认参数，选择网络结构参数如表1所示。值得注意的是，多头注意力机制中隐藏层的总数据维度应与模型隐藏层数据维度保持一致，因此注意力机制隐藏层的数据维度可以由模型隐藏层数据维度与注意力机制头数计算得到。由此，参数数量为 13056612。

| 参数名称 | 原论文 | 实际参数选择 |
|----------------|------|--------|
| 图像边长 | 224 | 32 |
| patch size | 16 | 16 |
| 图像类别数 | 1000 | 100 |
| 模型隐藏层数据维度 | 768 | 512 |
| Transformer 层数 | 12 | 6 |
| 注意力机制头数 | 12 | 8 |
| 前馈神经网络隐藏层数据维度 | 3072 | 1024 |
| 注意力机制隐藏层数据维度 | 64 | 64 |

表 1: ViT 网络结构参数

3.3 超参数设置

数据增强：裁剪、水平翻转；
 学习率：由 0.02 开始每 20 个回合阶梯下降一个数量级；
 优化器：带有 0.9 动量的随机梯度下降算法；
 正则化参数：0.0005；
 回合数：60；
 批量大小：128；
 每回合循环数：391；
 总循环数： $60 \times 391 = 23460$ ；
 损失函数：交叉熵损失函数；
 评价指标：精确度。

3.4 实验结果

实验结果如图15和表2所示。

| 模型 | 训练集 top1 | 训练集 top5 | 测试集 top1 | 测试集 top5 |
|--------|----------|----------|----------|----------|
| ResNet | 0.99904 | 1.00000 | 0.72480 | 0.91870 |
| ViT | 0.44518 | 0.75418 | 0.35280 | 0.64170 |

表 2: 实验结果

可以看到，相比 ResNet，Transformer 模型结果较差，原因是其缺乏归纳偏置，如局部性和平移等变形。尽管存在位置编码，它也是随机初始化、从零开始学习的。

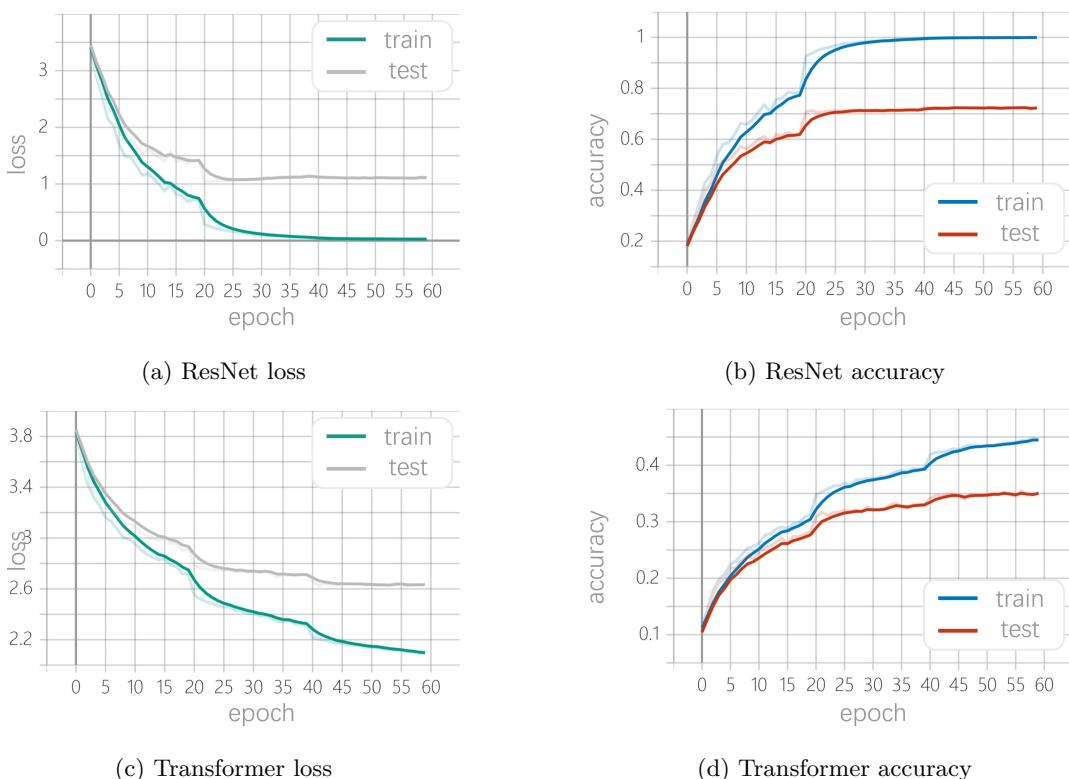


图 15: 实验结果