Assignment3 实验报告

龚畅 22210980103

代码开源在https://github.com/hmtbgc/NeuralNetworkAssignment/tree/main/assignment3

模型参数在链接：https://pan.baidu.com/s/1Ty6_MG3QjLQ_3TxN2JtRRA

提取码：2tfr

问题1：使用自监督处理图像分类

自监督学习简介

自监督学习是一种无需标注数据进行监督的学习方法，通过设计自动生成标签或目标任务来利用无标注数据进行训练。在图像中，自监督学习可以通过以下几种方式应用：

- 自编码器：自编码器是一种神经网络模型，将输入图像压缩为低维编码，然后解码为重构图像，通过比较原始和重构图像来学习图像的低维表示；
- 对比学习：对比学习通过将输入图像与其经过变换的版本进行比较，学习图像的表示。例如，通过将图像进行旋转、剪切或颜色变换，然后让模型判断两个图像是否来自同一张原始图像。
- 预测任务：自监督学习可以通过设计预测任务来学习图像的表示。例如，将图像划分为不同的块，并要求模型预测这些块的相对位置或颜色，从而使模型学习到图像的结构和语义信息。

自监督学习在图像中的应用可以提供大量的无监督训练数据，从而克服了标注数据的稀缺性和高成本问题。通过利用自监督学习，可以在无需人工标注的情况下，有效地学习图像的表示和语义信息，为下游的计算机视觉任务提供有力支持。

本项目使用的自监督学习方法：SimCLR

对于大量无标签图像，我们很难事先分辨哪两张图片属于同一类(正样本对)。SimCLR解决正负样本对的流程如下：

- 给定一个batch中N张图片，对其中任意一张x_i，做数据增强得到$x_i^{(1)}, x_i^{(2)}$。其中数据增强的方式包括如下三种：
  - 随机裁剪后再resize到原来大小；
  - 随机色彩失真；
  - 随机高斯模糊；
- 视$x_i^{(1)}, x_i^{(2)}$为正样本对，$x_i^{(1)}, x_j^{(1/2)}(j \neq i)$为负样本对

具体计算损失函数流程如下：(一个batch中N张图片，故共2N张增强图片，且$2i-1$和$2i$是正样本对)

- 先使用一个encoder f提取图片特征，得到$h_i = f(x_i)$
- 再使用一个projection head g进一步提取特征，得到$z_i = g(h_i)$
- 相似性计算：

$$s_{ij} = \frac{z_i^T z_j}{||z_i||||z_j||}$$

- 损失函数：

$$l(2i-1, 2i) = -\log \frac{\exp(s_{2i-1,2i}/\tau)}{\sum_{k \neq 2i-1} \exp(s_{2i-1,k}/\tau)}$$

$$l = \sum_{i=1}^{N} [l(2i-1, 2i) + l(2i, 2i-1)]$$

- 训练结束后，丢弃g，仅使用f作为其他任务的特征提取器。

实验设置

本项目使用CIFAR-10作为训练和测试数据集（数据划分遵循默认设置），比较监督学习和自监督学习在图像分类任务中的性能表现。两者使用的特征提取器均为resnet18，输出的h_i维度均为512。监督学习后再跟一个线性分类器MLP，并从头全参数训练。SimCLR在resnet18后再接一个projection head g:

```
self.g = nn.Sequential(
            nn.Linear(512, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, out_dim),
        )
```

自监督阶段训练完后，丢弃g，在resnet18后跟和监督学习相同的线性分类器MLP，并冻结resnet18的参数，仅训练MLP。SimCLR自监督学习阶段使用的数据为CIFAR-10的train部分，为50000张图片，未使用测试集部分，因此不会造成数据泄露的问题。后续linear evaluation使用数据集同监督学习。

实验结果

测试指标为top1 accuracy，选取测试集上最优结果进行报告：

SimCLR+linear evaluation	supervised
81.05%	93.68%

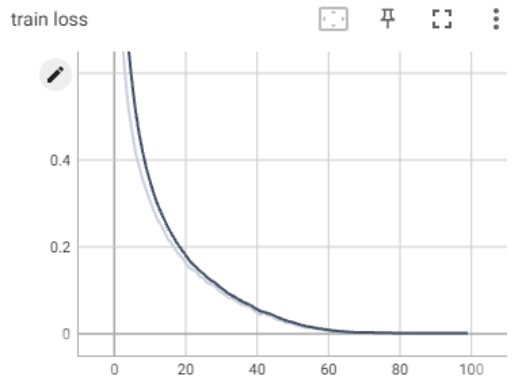
虽然论文中使用SimCLR+linear evaluation可以得到近似supervised的结果，但在实验中尝试多种超参数组合后依然难以达到相近的结果。可能的原因如下：

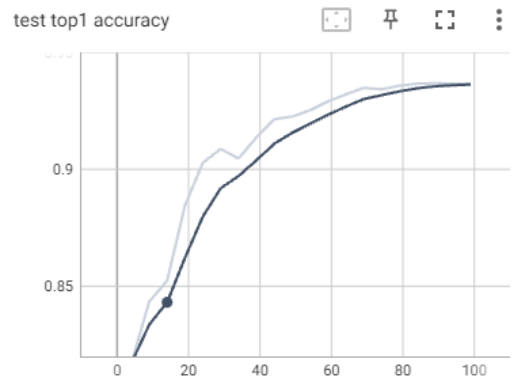
- 超参较敏感，需要高超的调试技巧
- 自监督的数据集较小，模型可能过拟合

参考github其他开源实现，都无法在CIFAR-10数据集上通过SimCLR+linear evaluation达到超过85%的分类准确率，因此可以认为本项目的实现是大致正确的。

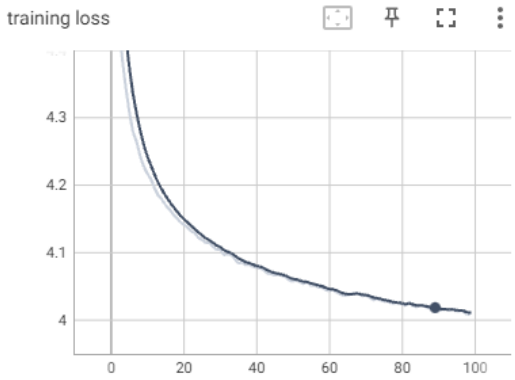
以下为train loss, test top1 accuracy 的变化图

supervised

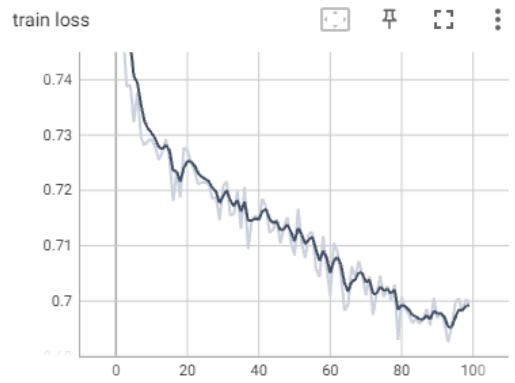


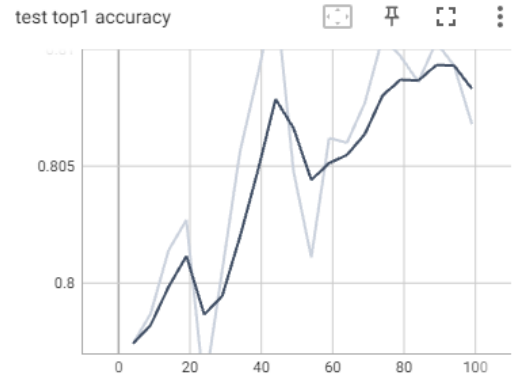


self-supervised



linear evaluation





问题2：使用Transformer模型训练图像分类

ViT简介

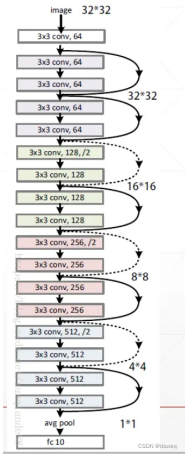
本项目使用的Transformer模型是ViT，计算流程如下：

- 先将输入的2D图像转为1D的patch embedding。假设输入的图片为$x \in \mathbb{R}^{H\times W\times C}$，$H, W, C$分别表示长、宽和通道数，并准备将其切分成$P \times P$大小的patch，则可以转化为$N = HW/P^2$长的patch 序列，每个patch的维数都是$(P, P, C)$。最后通过一个卷积操作将初始通道数$C$转化为输入的维数。
- 针对图像分类问题，在patch 序列前添加一个额外的cls token，最终的输出即为该token的embedding。
- 原始的transformer采用固定的position encoding，而ViT将其改为可学习的参数：在输入transformer之前将patch embedding 与 positional embedding相加。
- 其余部分同原始transformer，最后使用一个线性层MLP对cls token的输出embedding进行分类即可。

参数计算

本项目比较的对象为resnet18和ViT，要求两者可学习参数相近，下面进行详细的计算。

resnet18



resnet18结构如上，主要参数分布在卷积核和线性层（忽略batchnorm层）。针对一个输入通道I，输出通道O，卷积核大小为(K_1, K_2)的卷积操作，参数为IOK_1K_2，而一个输入维度为H_1，输出维度为H_2的线性层，参数为H_1H_2。因此resnet18的参数计算如下：

$$\begin{aligned}
&3*64*3*3+64*64*3*3*4+\\
&64*128*3*3+128*128*3*3*3+\\
&128*256*3*3+256*256*3*3*3+\\
&256*512*3*3+512*512*3*3*3+\\
&512*100\\
&=11,038,400
\end{aligned}$$

ViT

ViT主要参数分布在self-attention和feedforward层。一个self-attention操作包括4个转换矩阵：W_q, W_k, W_v, W_o。假设隐藏层维度为H，head数为M，则$W_q, W_k, W_v \in \mathbb{R}^{H\times H//M}, W_o \in \mathbb{R}^{H\times(H//M\times M)}$，而不同的head对应不同的$W_q, W_k, W_v$，且一般而言$H\%M == 0$，因此self-attention操作中总参数为$4H^2$。一个feedforward层包括两个MLP：$(H, H_1), (H_1, H)$，总参数为$2HH_1$。综上，一个encoder块中总参数为$4H^2+2HH_1$。实验中我们选取$H=512, H_1=1024$，总块数为6，则总参数为：

$$(4*512*512+2*512*1024)*6=12,582,912$$

实验结果

同第二次作业，我们采用CIFAR-100作为训练和测试数据集（数据划分遵循默认设置），且对数据进行三种不同的增强方式：Mixup,Cutout,Cutmix。评测指标为top1 error 和 top5 error，选取验证集上最优参数进行测试集上的测试，结果如下：

top1 error

Method	Baseline	Mixup($\alpha=0.4$)	Cutout	Cutmix($\alpha=0.4$)
ResNet18	26.01%	25.73%	26.99%	24.57%
ViT	50.64%	48.70%	53.08%	43.67%

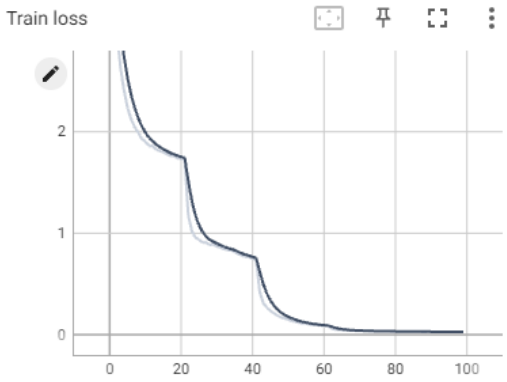
top5 error

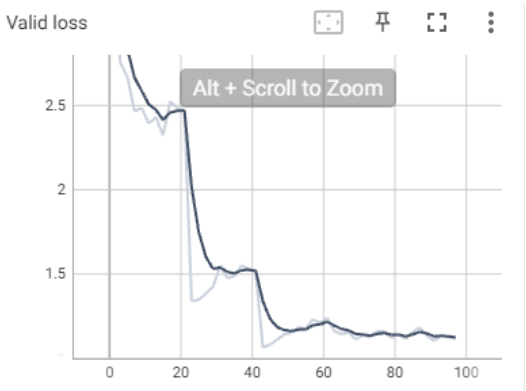
Method	Baseline	Mixup($\alpha=0.4$)	Cutout	Cutmix($\alpha=0.4$)
ResNet18	7.53%	7.83%	8.11%	6.55%
ViT	24.63%	22.72%	25.98%	17.32%

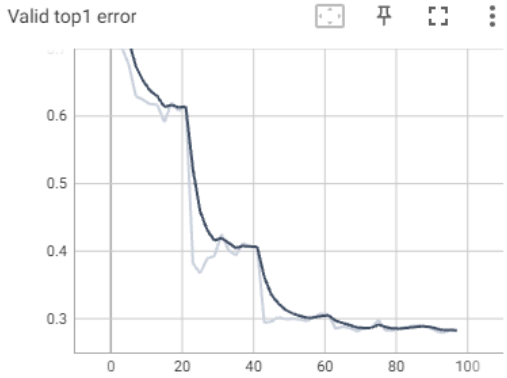
可以看到同参数下，ViT分类准确率远低于ResNet18。这符合原始ViT论文报告的结果：即使在ILSVRC-2012 ImageNet（共1.3M图片）上训练不同规模ViT，都要比ResNet低好多个点。相比于CNN，transformer encoder不包含与图像分类相关的归纳偏置，比如缺少旋转不变性、局部性等，导致ViT很容易过拟合。论文中采用的是“监督预训练+微调”的模式，评估ViT的迁移性能。虽然在下游19个VTAB任务中表现都好于ResNet架构的模型，但监督预训练时用到了包含303M张图片的私有数据集JFT，普通人无法复现。

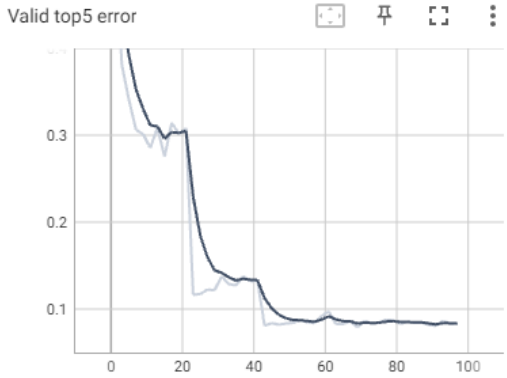
以下为没有使用数据增强时， tensorboard可视化ResNet18/ViT 的 train loss, valid loss, valid top1 error, valid top5 error 变化图

ResNet18

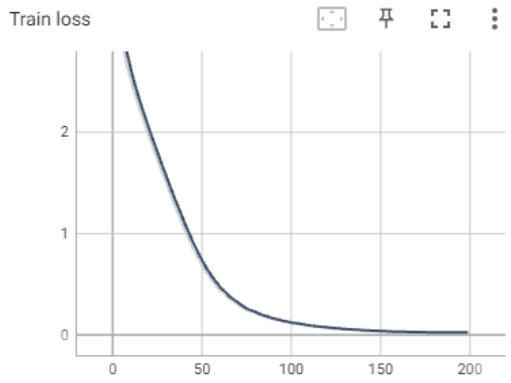

Train loss
2
1
0
0
20
40
60
80
100

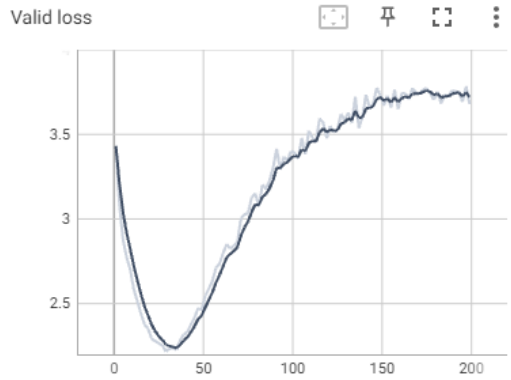

Valid loss
Alt + Scroll to Zoom
2.5
2
1.5
0
20
40
60
80
100

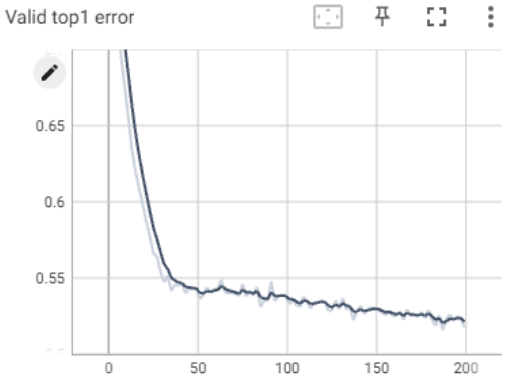

Valid top1 error
0.6
0.5
0.4
0.3
0
20
40
60
80
100

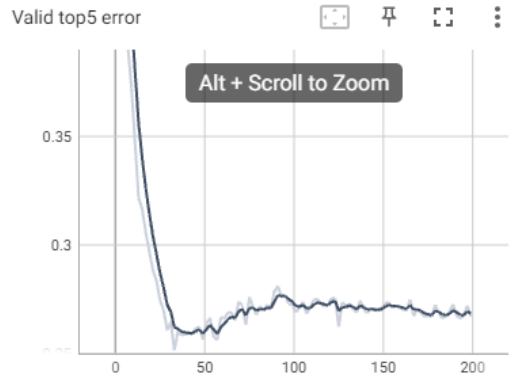

Valid top5 error
0.3
0.2
0.1
0
20
40
60
80
100

ViT









问题3：三维重建

采用Instant-NGP
步骤如下：

- 将代码库git clone 到本地，并按照requirements.txt安装相关python库
- windows下根据自己的显卡型号安装对应的release版本
- 拍摄想要重建的模型的视频，放在恰当位置
- 运行下面脚本，将自动下载colmap和ffmpeg，并把视频按照每秒2帧进行切割

```
python ./scripts/colmap2nerf.py --video_in /path/to/movie --video_fps 2 --run_colmap --aabb_scale 16 --overwrite
```

- 上述脚本将在/path/to/movie目录下生成images文件夹，手动去除其中模糊的图片
- 运行下面脚本

```
cd /path/to/movie
python /path/to/scripts/colmap2nerf.py --colmap_matcher exhaustive --run_colmap --aabb_scale 16 --overwrite
```

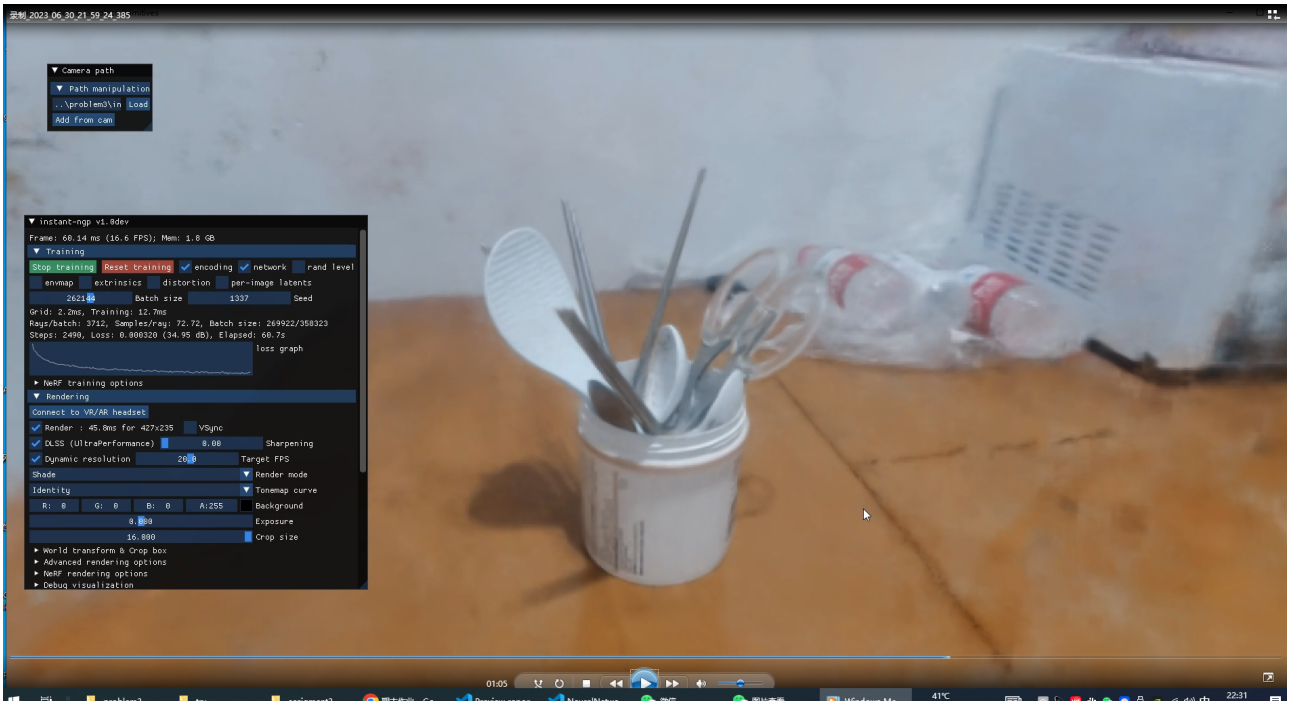
- 找到第二步下载的目录下的instant-ngp.exe文件，运行下面脚本

```
/path/to/instant-ngp.exe /path/to/movie
```

- 开始训练，并得到最终结果。具体结果可以参考github仓库中的演示视频。

例子：
原始拍摄：

重建结果：


录制 2023_06_30_21_59_24_385
Camera path
Path manipulation
..\problem3\in
Load
Add from cam
instant-ngp v1.0dev
Frame: 60.14 ms (16.6 FPS); Mem: 1.8 GB
Training
Stop training
Reset training
encoding
network
rand level
envmap
extrinsics
distortion
per-image latents
262144
Batch size
1337
Seed
Grid: 2.2ms, Training: 12.7ms
Rays/batch: 3712, Samples/ray: 72.72, Batch size: 269922/358323
Steps: 2490, Loss: 0.000320 (34.95 dB), Elapsed: 60.7s
loss graph
NeRF training options
Rendering
Connect to VR/AR headset
Render : 45.8ms for 427x235
VSync
DLSS (UltraPerformance)
8.00
Sharpening
Dynamic resolution
20
Target FPS
Shade
Render mode
Identity
Tonemap curve
R: 0
G: 0
B: 0
A:255
Background
0.000
Exposure
16.000
Crop size
World transform & Crop box
Advanced rendering options
NeRF rendering options
Debug visualization
01:05
41°C
22:31