计算机视觉期末报告

张克俭　　池钊升
20307110266　20307130107

2024 年 6 月 21 日

目录

1 任务一

1.1 任务描述

本任务对比监督学习和自监督学习在图像分类任务上的性能表现。首先，实现任一自监督学习算法并使用该算法在自选的数据集上训练 ResNet-18，随后在 CIFAR-100 数据集中使用 Linear Classification Protocol 对其性能进行评测；其次，将上述结果与在 ImageNet 数据集上采用监督学习训练得到的表征在相同的协议下进行对比，并比较二者相对于在 CIFAR-100 数据集上从零开始以监督学习方式进行训练所带来的提升；最后，尝试不同的超参数组合，探索自监督预训练数据集规模对性能的影响。

项目地址：`https://github.com/1376896121/FudanCV-Final`。

最佳模型权重:`https://pan.baidu.com/s/1ADv3W2GVLBNTJZ6Ip4DyRg?pwd=cquf`。

1.2 数据集

本任务中，自监督学习使用的数据集是 CIFAR-10，CIFAR-100，STL-10，Linear Classification Protocol 过程使用的数据集是 CIFAR-100。

CIFAR-10 包含 60,000 张 32x32 像素的彩色图像，分为 10 个类别，每个类别有 6,000 张图像，其中训练和测试集中各有 5,000 张和 1,000 张图像。CIFAR-100 同样包含 60,000 张 32x32 像素的图像，但分为 100 个类别，每个类别有 600 张图像，其中训练和测试集中各有 500 张和 100 张图像。并且 CIFAR-100 中 100 个类别组织为 20 个超级类，每个超级类下有 5 个子类。由于类别数目和复杂度的差异，CIFAR-100 的分类任务比 CIFAR-10 更具挑战性。

STL-10 是另一个常用的图像数据集，包含 130,000 张 96x96 像素的彩色图像，分为 10 个类别。与 CIFAR-10 和 CIFAR-100 不同，STL-10 的数据集特别设计用于半监督学习，包含更多未标记的图像。每个类别的标记训练图像有 500 张，测试图像有 800 张。此外，STL-10 的图像分辨率更高，这为模型提供了更多的细节信息。由于图像的数量和多样性，STL-10 在训练复杂模型和验证其性能方面提供了更丰富的数据。

1.3 SimCLR 自监督学习

SimCLR(Simple Framework for Contrastive Learning of Visual Representations) [1] 是一种用于无监督学习视觉表征的有效方法，提出了对比学习在计算机视觉领域实现的一种简单架构，其架构见图 1。首先，从同一张图像中生成两个不同的视图，使用随机的图像增广操作，如随机裁剪和颜色失真。这些增广后的图像通过一个基本编码器网络编码为表示，再通过一个投影头将这些表示投影到一个新的低维空间。然后，通过对比损失函数最大化同一图像的两个视图在投影空间中的相似性，同时最小化不同图像之间的相似性。在训练完成后，投影头被丢弃，仅保留编码器和中间表示用于下游任务。通过这种方法，SimCLR 能够在无监督学习中生成高质量的视觉表征，表现出色。

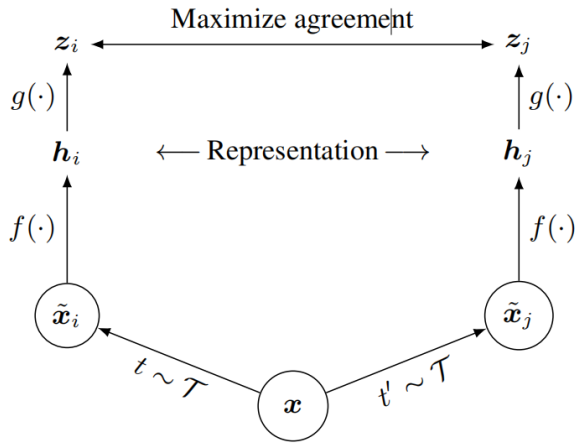


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

图 1: SimCLR 架构

1.4 Linear Classification Protocol

Linear Classification Protocol 是一种用于评估自监督学习模型表示能力的方法。在这种协议中，首先使用自监督学习方法在无标签数据集上训练一个模型，以学习出良好的数据表示。然后，固定住预训练模型的参数，仅在其顶部训练一个简单的线性分类器。通过在有标签的数据集上训练和评估线性分类器的性能，来间接衡量预训练模型所学表示的质量。这种方法的基本假设是，如果模型学到的表示良好，那么即使是一个简单的线性分类器也能够在下游任务中取得较高的准确率。Linear Classification Protocol 通常用于验证自监督学习方法的有效性，例如在图像分类任务中的应用。

1.5 训练过程

本任务在云服务器上完成，使用的显卡配置是 RTX3090:24GB。

首先，我们分别用 CIFAR-10，CIFAR-100，STL-10 数据集的训练集部分在相同超参数下应用 SimCLR 方法训练 Resnet-18，随后在 CIFAR-100 数据集中使用 Linear Classification Protocol 对其性能进行评测。其次，选择评测效果最好的训练数据集，改变自监督学习过程中的学习率，寻找能实现最佳效果的超参数。此外，我们还改变了自监督预训练数据集规模，探究规模对性能的影响。最后，将自监督学习得到的最佳表征和在 ImageNet 数据集上采用监督学习训练得到的表征用于 fine-tune，比较二者相对于不使用预训练（即在 CIFAR-100 数据集上从零开始以监督学习方式进行训练）所带来的提升。

1.6 实验结果与分析

1.6.1 不同自监督预训练数据集的训练结果比较

超参数设置见表 1。不同自监督预训练数据集训练得到的表征在 LCP(Linear Classification Protocol) 下得到的测试集准确率见表 2。其中 CIFAR-100 效果最好，CIFAR-10 其次，STL-10 最差。CIFAR-100 效果最好的原因可能是 CIFAR-100 的训练集与测试集之间数据更相似，包含更多的数据标签种类。CIFAR-10 比 STL-10 效果更好的原因可能是 CIFAR-10 与 CIFAR-100 的

数据更为相似，如分辨率都是 32*32，而 STL-10 中数据的分辨率为 96*96。

Hyperparameter	**Value**
SimCLR Learning Rate	3e-4
LCP Learning Rate	3e-4
Random Seed	42
Batch Size	128
Image Size	224
Epochs	100
Projection Dimension	64
Optimizer	Adam
Weight Decay	1e-6
Temperature (Related to N-Xent Loss Function)	0.5
Epoch Number	100
LCP Batch Size	256
LCP Epochs	500

表 1: 自监督学习使用的超参数设置

Dataset	**Test Accuracy**
CIFAR-10	50.7%
CIFAR-100	**55.4%**
STL-10	50.3%

表 2: 不同自监督预训练数据集 LCP 准确率

1.6.2 不同自监督学习率的训练结果比较

在 CIFAR-100 数据集上，更改自监督学习率为 1e-3 和 1e-4，得到不同学习率下的 LCP 准确率见表 3。学习率 1e-3 下的预训练 Loss 变化过程见图 2。Linear Classification 过程中的测试集准确率变化见图 3。

Lr	Test Accuracy
1e-3	**55.6%**
3e-4	55.4%
1e-4	54.4%

表 3: 不同学习率下 CIFAR-100 预训练的 LCP 准确率

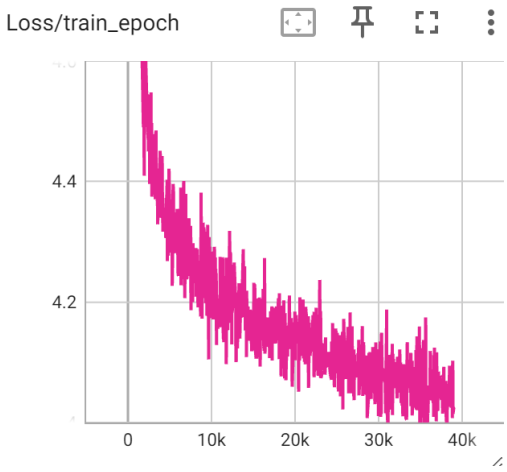


图 2: 最佳学习率下的预训练 Loss 变化过程

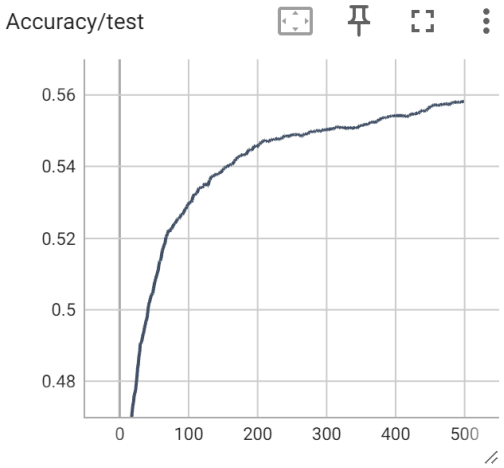


图 3: Linear Classification 过程中的测试集准确率变化过程

1.6.3 自监督预训练数据集规模对性能的影响

为探究自监督预训练数据集规模对性能的影响，我们选取 CIFAR-100 训练集的不同数据量的子集，在 3e-4 的学习率下预训练得到表征，同样应用 LCP，得到表 4。可见随着数据集规模下降，LCP 准确率下降，性能变差。

Amount	**Test Accuracy**
50000(Oringinal)	**55.4%**
25000	51.5%
10000	45.8%
5000	42.4%

表 4: 不同数据量 CIFAR-100 子集的 LCP 准确率

1.6.4 自监督预训练带来的提升

同样将 ImageNet 数据集上采用监督学习训练得到的表征用于 LCP，且在 CIFAR-100 数据集上从零开始以监督学习方式进行训练，与 SimCLR 自监督学习得到的最佳表征用于 LCP 的效果做对比。三种情况下的 Test

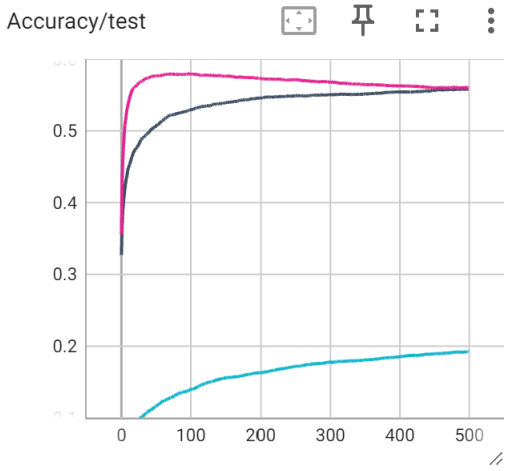


图 4: Test Accuracy: 粉线代表 Imagenet 表征，黑线代表 SimCLR 表征，天蓝色为随机初始化表征

Accuracy 曲线如图 4 所示，Test Lost 曲线如图5所示。可见两种预训练方式相比从零开始训练都有显著提升。

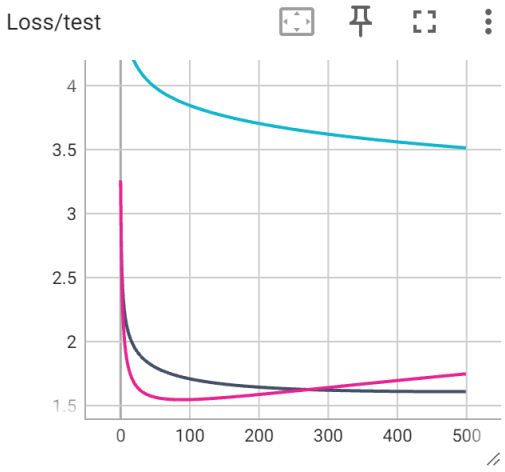


图 5: Test Lost: 粉线代表 Imagenet 表征，黑线代表 SimCLR 表征，天蓝色为随机初始化表征

2 任务二

2.1 任务描述

在 CIFAR-100 数据集上比较基于 Transformer 和 CNN 的图像分类模型主要任务如下：

- 分别基于 CNN 和 Transformer 架构实现具有相近参数量的图像分类网络
- 在 CIFAR-100 数据集上采用相同的训练策略对二者进行训练，其中数据增强策略中应包含 CutMix
- 尝试不同的超参数组合，尽可能提升各架构在 CIFAR-100 上的性能以进行合理的比较。

项目地址：`https://github.com/1376896121/FudanCV-Final`。

2.2 数据集描述

CIFAR-100 是一个用于图像分类和机器学习的广泛使用的数据集。它由加拿大多伦多大学的多伦多视觉小组（Toronto Vision Group）创建，是 CIFAR 数据集家族的一部分。CIFAR-100 包含 60000 张彩色图像，每张图像的尺寸为 32x32 像素。数据集分为 100 个类别，每个类别有 500 张训练图像和 100 张测试图像，这些类别被进一步分为 20 个超类（superclasses），每个超类包含 5 个细类（fine classes）。图像涵盖了广泛的对象和场景，包括动物、植物、交通工具、家用电器等，每张图像都被标注为一个具体的类别。

2.3 模型介绍

2.3.1 CNN

我们采用的 CNN 模型为 Resnet-152，参数量为 60.2M。ResNet-152 共有 152 层，包括卷积层、批归一化层、ReLU 激活函数和全连接层。其主要结构包括一个初始的 7x7 卷积层，随后是多个由 1x1、3x3 和 1x1 卷积

层组成的瓶颈残差块，最后通过全局平均池化层连接到全连接层进行分类。ResNet-152 在许多计算机视觉任务中表现优异，广泛应用于图像分类、目标检测和语义分割等领域。

2.3.2 Transformer

ViT（Vision Transformer）是一种基于 Transformer 架构的图像分类模型，利用 Transformer 的自注意力机制处理图像数据。它首先将输入图像划分为固定大小的非重叠小块（patches），然后将每个小块展平成一维向量，并通过线性投影映射到高维空间，形成“切块嵌入”。这些嵌入向量类似于 Transformer 在处理文本时的词嵌入。随后，ViT 将这些嵌入向量添加位置编码，以保留图像的位置信息，然后输入到标准的 Transformer 编码器中。通过多层自注意力和前馈神经网络，ViT 能够有效地捕捉图像的全局特征，最终通过分类头进行图像分类。与传统的卷积神经网络（CNN）不同，ViT 无需大量的卷积操作，依赖于全局自注意力机制，使其在大规模数据和强大计算资源下表现出色。

2.4 参数设置

Resnet-152 的参数配置见表5，Vision Transformer 的参数配置见表6。

Hyperparameter	**Value**
Learning Rate	1e-2
Random Seed	42
Batch Size	128
Image Size	224
Epochs	300
Optimizer	SGD
Momentum	0.9
Weight Decay	5e-4

表 5: Resnet-152 的超参数设置

Hyperparameter	**Value**
Learning Rate	1e-4
Random Seed	42
Batch Size	32
Image Size	224
Epochs	15
Projection Dimension	768
Optimizer	Adam

表 6: ViT 的超参数设置

2.5 训练策略

采用 CutMix 数据增强策略，随机从训练数据集中选择两张图像 A 和 B，以及它们对应的标签 y_A 和 y_B, 再随机从图像 B 中选择一个矩形补丁替换掉 A 中的一个矩形区域，最后计算混合数据的标签，计算方式如下，其中 λ 表示补丁的面积比例。

$$\lambda y_A + (1 - \lambda) y_B$$

Resnet-152 和 ViT 的训练都使用了 Imagenet 预训练权重。

2.6 训练结果可视化

Resnet-152 训练过程中的训练集的 loss 和验证集的 loss 如图 6和图 7。验证集上的准确率曲线如图 8所示。测试集上的最终准确率为 **69.67%**。权重地址见 github 项目页面。

ViT 训练过程中的训练集的 loss 和验证集的 loss 如图 9。验证集上的准确率曲线如图 10所示。验证集上最高的准确率有 72.62%, 在测试集上的准确率为 **82.63%**。权重地址见 github 项目页面。

ViT 的性能优于 Resnet-152，且需要的迭代次数更少。这可能是因为 ViT 能够更好地捕捉全局图像特征，适应更复杂的视觉任务。因此，在选择模型时，ViT 可能是一个更优的选择，特别是在需要更高准确率的应用场景中。

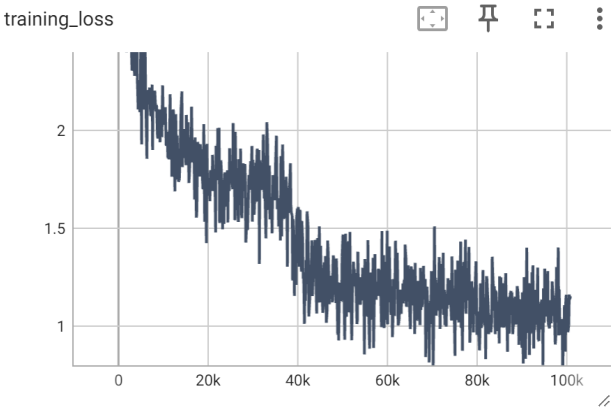


图 6: Train Loss for Resnet-152

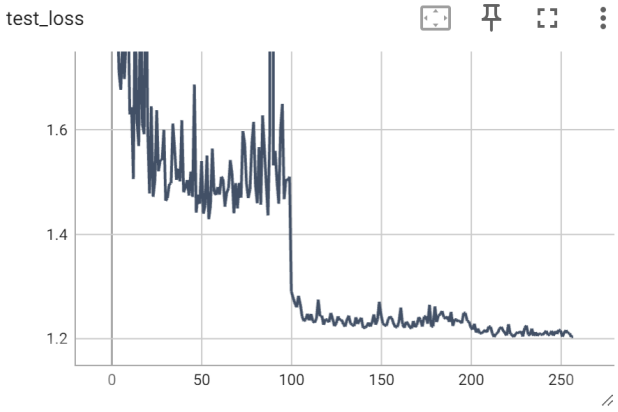


图 7: Test Loss for Resnet-152

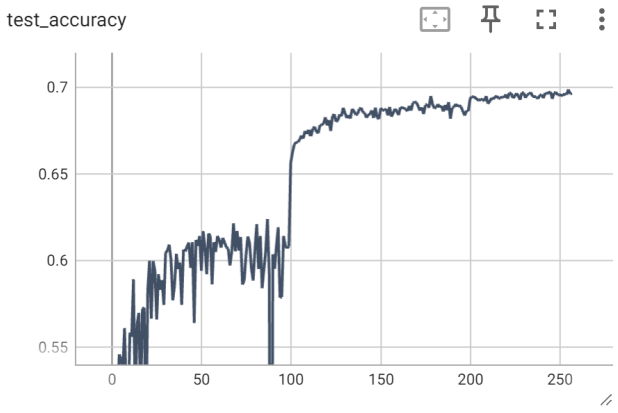


图 8: Test Accuracy for Resnet-152

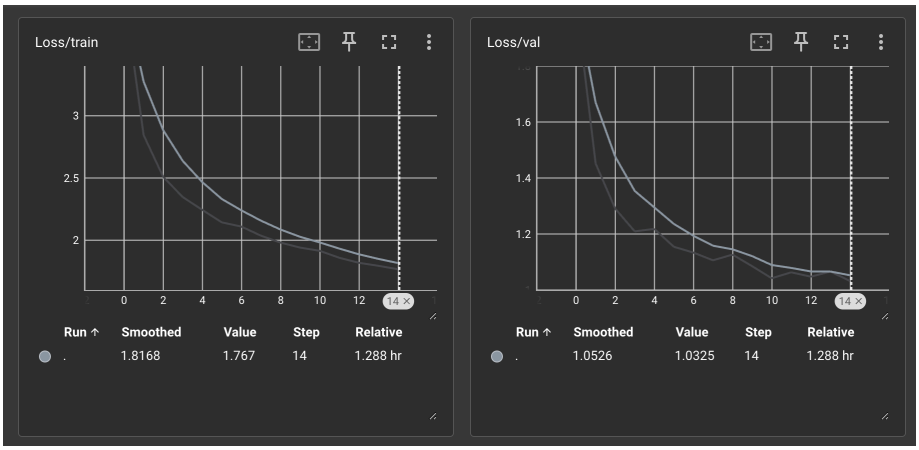


图 9: Loss curve for ViT

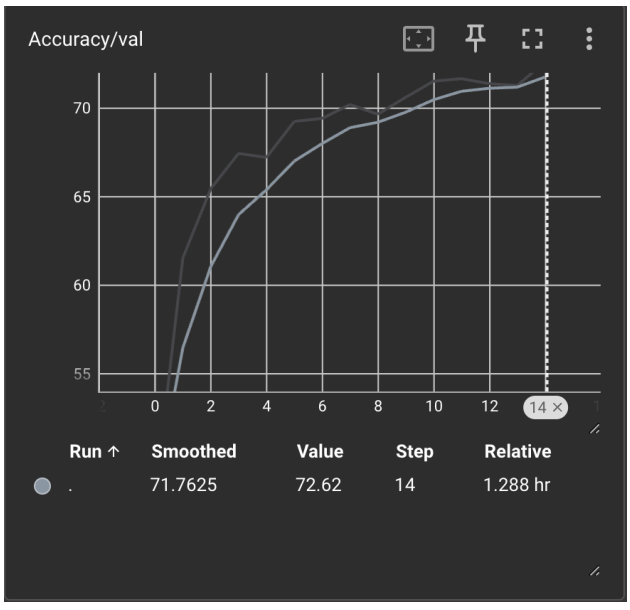


图 10: Accuracy curve on validation set for ViT

3 任务三

3.1 任务描述

基于 NeRF 的物体重建和新视图合成主要任务如下:

- 选取身边的物体拍摄多角度图片/视频，并使用 COLMAP 估计相机参数，随后使用现成的框架进行训练
- 基于训练好的 NeRF 渲染环绕物体的视频，并在预留的测试图片上评价定量结果

项目地址为：`https://github.com/Jason-Chi-xx/CV_final_nerf`

3.2 训练流程

3.2.1 数据处理

在开始训练 NeRF 模型之前，首先需要进行数据处理。NeRF 模型依赖于多视角的 2D 图像数据以及相应的相机参数。通常情况下，数据处理包括以下几个步骤:

- **数据收集**：从不同角度拍摄目标物体的 2D 图像。这些图像需要覆盖物体的所有视角，以确保模型能够学习到完整的 3D 结构。
- **相机参数提取**：每张图像需要对应的相机位姿参数（位移和旋转）。这些参数可以通过外部工具如 COLMAP 进行结构从运动（SfM）来估计。
- **图像预处理**：首先将图片转化为模型要求的格式，在本任务中需要将第二步获得的位姿转化为 llff 格式的数据。之后将图像进行归一化处理，例如调整图像的分辨率和色彩空间，确保输入数据的一致性。

3.2.2 框架选择

对于 NeRF 模型的训练，可以选择各种深度学习框架，常见的包括 TensorFlow、PyTorch 等，由于 PyTorch 更适合研究和实验，我们在本任务中选

择了 PyTorch 作为深度学习框架，并采用了被广泛使用的 nerf-pytorch 项目进行实验，项目地址为：`https://github.com/yenchenlin/nerf-pytorch`

3.2.3 模型结构

NeRF 模型主要由一个深度神经网络（通常是全连接网络）构成，用于映射空间坐标 (x, y, z) 和视角方向 (θ, ϕ) 到颜色和密度。模型结构的设计包括：

- **输入表示**：将 3D 空间坐标和视角方向作为输入。为了捕捉空间细节，通常会使用位置编码（positional encoding）来增强输入的表达能力。
- **网络层次**：典型的 NeRF 模型由多个全连接层组成，每一层后面跟随激活函数（例如 ReLU），以增加网络的非线性表示能力。
- **输出表示**：网络输出为每个空间点的颜色和密度值。颜色值表示点的 RGB 值，密度值用于体渲染过程中的体素密度。

3.2.4 训练过程

NeRF 模型的训练过程包括以下步骤：

- **损失函数选择**：使用重建损失（如均方误差）比较模型渲染的图像和实际的 2D 图像。目标是最小化重建误差，使得生成的图像尽可能接近真实图像。
- **优化算法**：使用优化算法如 Adam 进行参数更新。训练过程中需要调整学习率、批次大小等超参数，以加速收敛和避免过拟合。
- **优化和细化**：为提高模型的泛化能力，可以对训练图像进行数据增强，如随机裁剪、旋转和颜色变换。

3.2.5 数据后处理

在模型训练完成后，需要对生成的 3D 数据进行后处理，以便应用于实际场景：

- **渲染图像生成**：使用训练好的模型渲染出不同视角下的 2D 图像，验证模型的效果。
- **3D 模型重建**：根据渲染图像和密度值，生成 3D 点云或网格模型。这些模型可以用于可视化或进一步的 3D 应用。
- **优化和细化**：通过后处理技术，如噪声过滤和细节增强，提高生成的 3D 模型的质量。

3.3 图片展示

图. 11 展示了用于训练的物体的图像

3.4 COLMAP 估计

首先建立一个文件夹叫做 blueball，再对桌子上的小球进行 180 度环绕拍摄，拍摄完图片后再在 blueball 里新建一个叫做 images 的文件夹将这组图片放入 images。之后使用 colmap 对这组图片进行位姿估计，首先进行特征提取，再进行特征匹配，最后进行重建，重建结果如图. 12。重建的参数都放入 blueball 目录下的 sparse/0 文件夹中。随后再将位姿数据都转化为 llff 格式的数据，转化完后会获得一个叫做 poses_bounds.npy 的文件。

3.5 NeRF 训练

采用了现有的框架 nerf-pytorch 进行训练，首先将输入图片进行 8 倍下采样，再将设置里的相关路径改成存放训练数据的路径即可训练的 loss 曲线和 PSNR 曲线分别如图. 13(a) 图. 13(b)所示

在两张测试图片上测试了 PSBR 和 SSIM 指标，图片和图. 14。指标如表. 7所示。PSNR 值和 SSIM 值越大，表示图片越相似。其中 SSIM 取值为 $(0, 1)$。可以看到重建效果还是不错的。

视频存放在百度云盘里，

链接: `https://pan.baidu.com/s/1D90uhXOgNjAzjYJJLKQ7Cw?pwd=qc1z`

提取码: qc1z

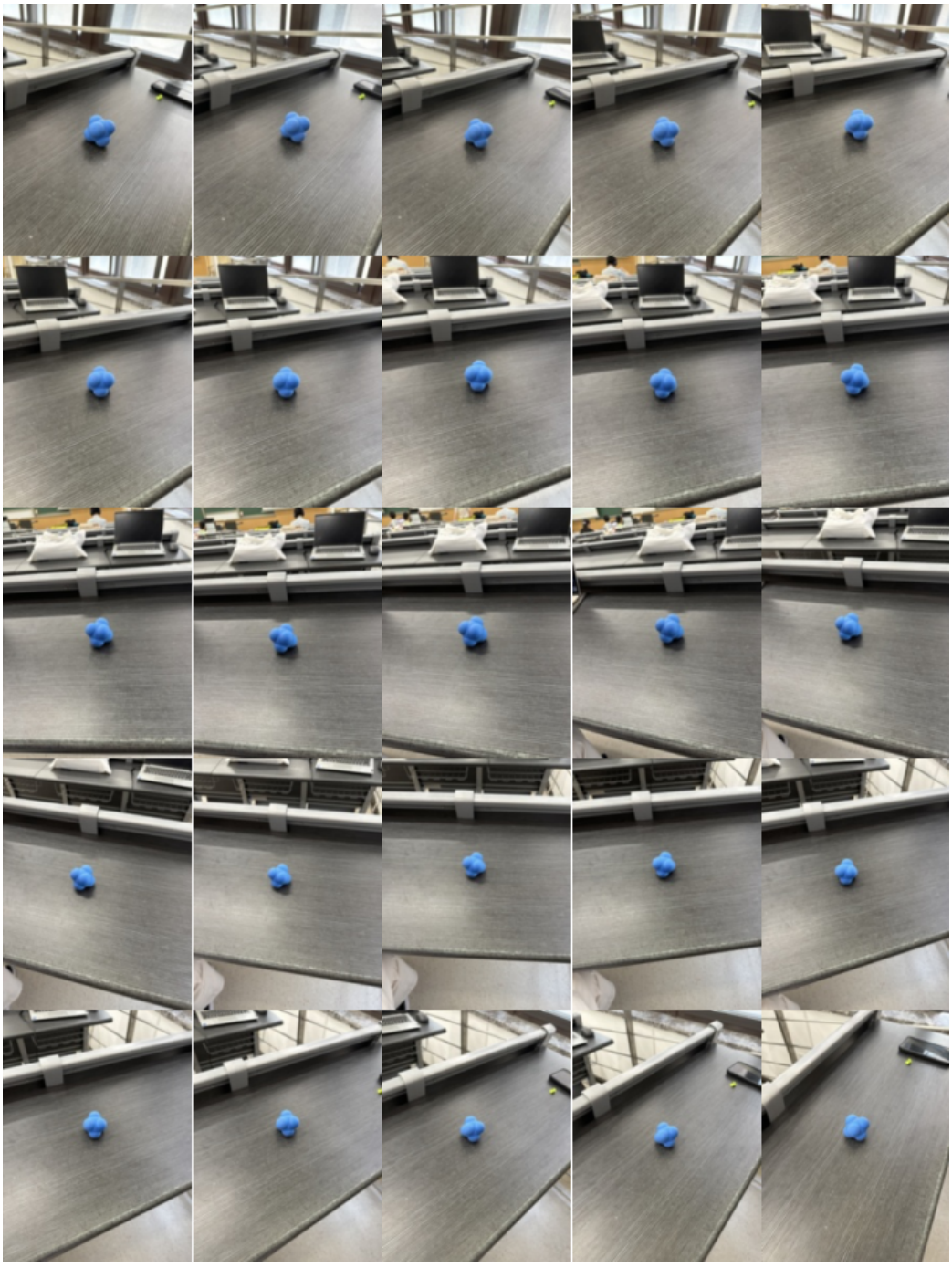


图 11: Display of the training images. Totally 25 images, showing different angle of a blue ball on the desk

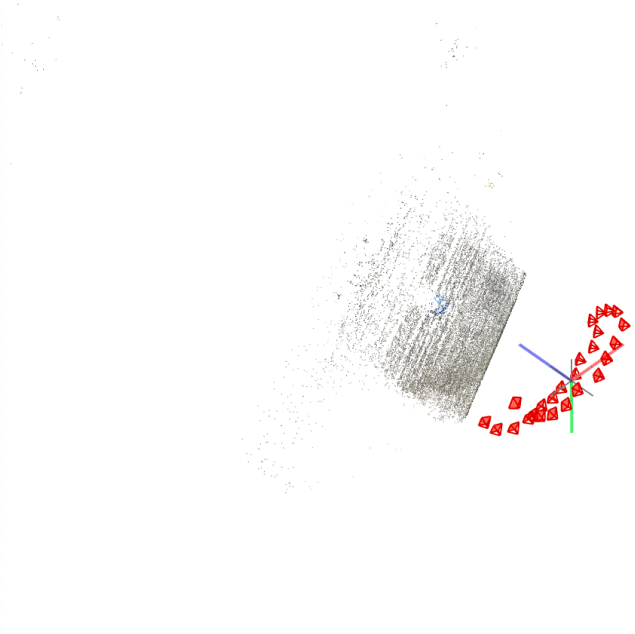
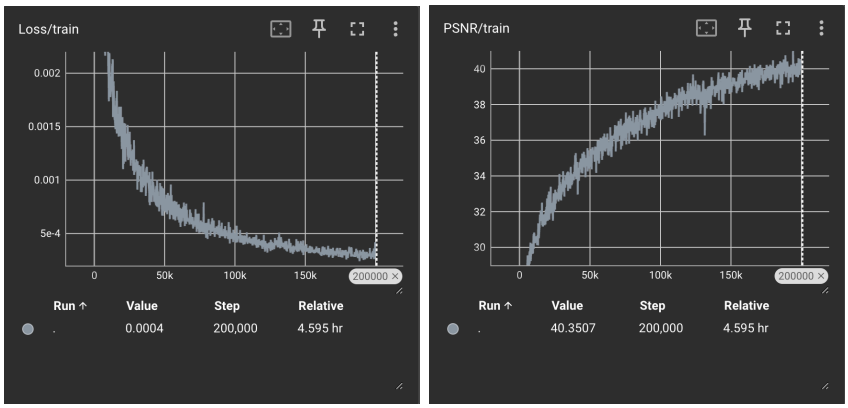


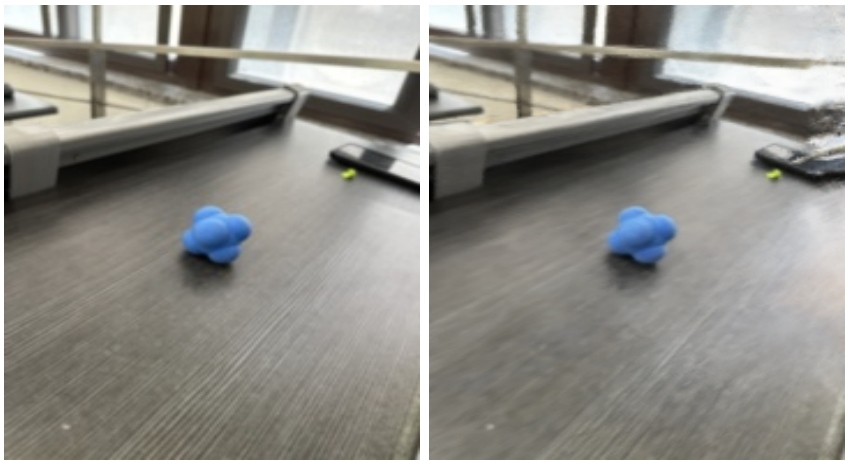
图 12: The camera pose reconstruction result of the blue ball



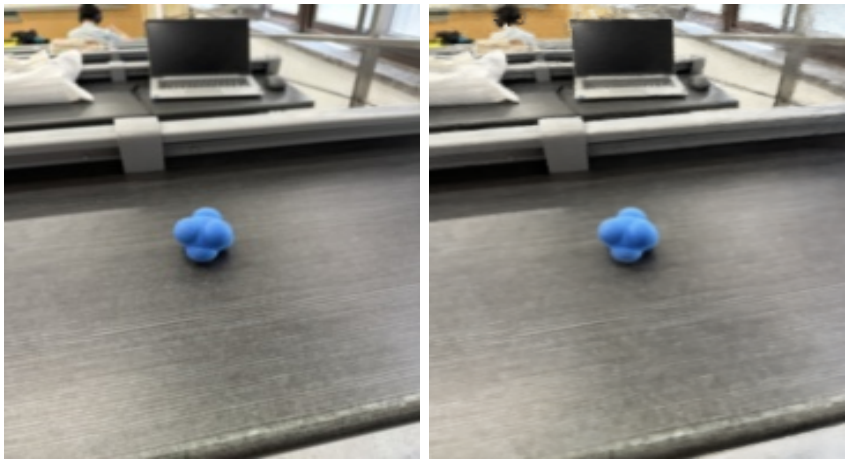
((a)) training loss curve ((b)) PSNR curve

	PSNR ↑	**SSIM** ↑
image 1	22.849	0.866
image 2	27.713	0.922

表 7: 测试图片上的定量指标



((a)) test image 1 ((b)) reconstructed image 1



((c)) test image 2 ((d)) reconstructed image 2

图 14: Two of the test images and their reconstructions counterpart

参考文献

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.