

**图像与视频处理课程报告**

**日 期： 2022年12月23日**

**题　　目：EfficientNeRF高效神经辐射场**

**学 院： 计算机学院**

**专业名称： 计算机技术**

**学 号： 3220220899**

**姓 名： 陈现章**

# EfficientNeRF高效神经辐射场

姓名：陈现章 学号：3220220899

**摘要：**神经辐射场NeRF以其对三维场景的高质量表示而被广泛应用于三维重建中，但它需要较长的单场景训练时间和单图像测试时间。 因此如何提高NeRF的渲染效率是亟待解决的难题，EfficientNeRF分析了采样点的密度分布和权重分布，分别在粗采样和细采样阶段提出有效采样和关键采样，从而显著提高采样效率。我对EfficientNeRF这篇文章进行了复现，并与其他的神经辐射场方法NeRF的结果进行了对比和分析。

**关键词：**神经辐射场、三维重建、高效渲染

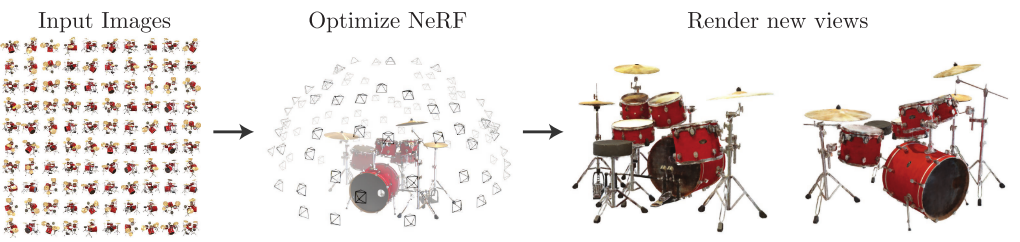
## 三维重建任务

### 任务背景

3D模型相较于2D图片，更能够体现出目标的真实信息，更符合人类的视觉感知。而随着对高精度3D模型的需求日益增长，如何进行图像的新视点合成是计算机视觉领域所共同关注的重点问题[1,2]。简单来说，基于图像的新视点合成即使用已知视点的几幅图片作为输入，对这些图片的几何、外观等性质进行提取，从而能够对其他的视点图片进行渲染合成[2]。

### 1.2 研究意义

Mildenhall 等[3]提出的神经辐射场NeRF (Neural Radiance Fields) 促进了神经渲染领域的快速发展，通过MLP多层感知机实现对3D目标和和场景的编码，以实现新视点图片的生成。MLP的使用能够将三维空间中的位置、视角信息映射到其内部的一个隐变量，通过对场景进行学习来合成新视角的图片[4]。该模型在深度全连接神经网络的参数编码中使用了连续体素，解决图像分辨率提高带来的限制。首先从稀疏图像视点获取图像数据，将视点位姿与观测视线方向组成 5D 坐标作为网络的输入，输出采样点的体积密度与相关颜色信息，最终能够获得繁杂场景中的新视点[3]。相较于传统方法，NeRF生成的图片在质量上有了明显的提升。图1展示了NeRF的整体流程。



**图1 输入同场景不同视角图片，合成新视角图片**

### 1.3 研究热点

然而，NeRF的高质量重建效果所付出的代价也是巨大的。首先NeRF不具有泛化性，对于每一个新的场景，都需要长时间的重新训练；而且由于这种MLP隐状态的使用，在渲染新视角图片时花费时间也非常长[5]；更为重要的是NeRF的本质是依赖于相同场景在不同视角下的几何一致性，当场景中存在动态物体时，这种一致性就不再存在，导致NeRF无法对动态物体的场景进行表达。

这些问题的存在成为了众多学者的研究重点，PixelNeRF[6]从ResNet[7]中引入图像特征，从而跳过对新场景的训练，这在一定程度上降低了训练成本，然而这不可避免的造成了精度的下降；NSVF[8]提出了一种将NeRF和稀疏体素结构相结合的混合场景表示方法，生成的稀疏体素用来引导和减少采样，从而提高推理速度；FastNeRF[9]和PlenOctree[10]在推理阶段预先存储各个点的MLP特征来加快渲染速度。KiloNeRF[11]为了减少推理时的时间开销，采用了1000个小MLP进行局部建模，每个MLP负责特定的3D区域，这些改进方法都在一定程度上加快了对场景的整体渲染速率。

### 1.4 应用场景

NeRF与传统的三维网格、点云和体素[5,12]不同，它在采用体绘制合成图像的同时，引入隐函数对场景进行建模，与基于Voxels Based体素的表示相比，NeRF克服了分辨率和存储能力的限制来合成高质量的结果[1]。它突破了常规三维重建方法的限制，不再依赖于特定的采样设备和独特的数据结构，而是普通的RGB图片和对应的姿态信息，并且相关技术的日益成熟已经能够让它在我们的日常生活中发挥重要的作用，例如：一些特定文物的三维重建保存、汽车自动驾驶、图像生成、视角生成、重新光照等应用[14]。

## 2 动机与理论方法

### 2.1 EfficientNeRF的动机

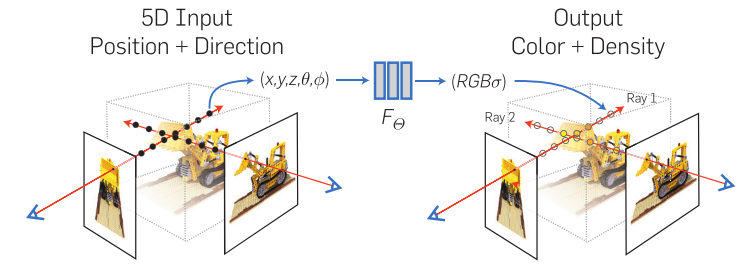
原始NeRF的pipeline结构由两部分组成：粗采样和细采样。在训练过程中，粗采样阶段得到整个场景的密度分布。它均匀地、密集地采样点，并通过一个MLP计算相应的密度。然而，对于具有均匀采样的场景，只有大约10%-20%的有效样本，5%-10%的关键样本，同时考虑到光线在传播过程中有很大一部分是背景，不包括背景信息，因此重复和无用的计算产生了大量的时间开销。

基于上章1.3中所提到的研究热点及NeRF的一些局限性，与PixelNeRF[6]、NSVF[8]、FastNeRF[9]、PlenOctree[10]和KiloNeRF[11]所采取的方式不同，EfficientNeRF[15]同时考虑了训练和测试的处理流程，综合了存储cache方法和关键点采样的方法，提出了一种能够在训练和渲染阶段都提高速度的方法。这是第一个能够在不降低精度的前提下，高效的利用场景中的有效点，高质量且快速的渲染重建出新视点图片的方法。

### 2.2 EfficientNeRF的方法

#### 2.2.1 NeRF的原理

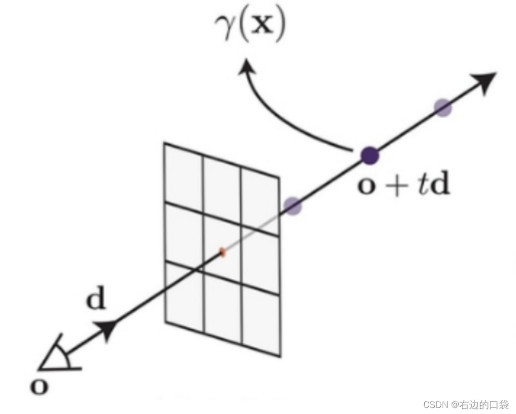
在介绍EfficientNeRF的改进之前，我们首先要介绍原始NeRF的基础原理和相应的公式。如图2所示NeRF学习的是一个隐式的映射，将输入的5D信息映射为4D的颜色和体密度属性。



**图2 NeRF的原理示意图**

如式(2.1)所示，其中代表了当前点的三维空间坐标，为当前视点的角度，为通过MLP的映射所获得的颜色信息，代表体密度，即光线穿过当前点被遮挡住的概率，体密度越大这点在光线上的颜色占比就越大。

图3中给定视角的光线可以通过式(2.2)表示出来，其中代表视点的初始位置，代表视点的方向，随着的变化，能够代表当前直线上的所有点。



**图3 从视点穿过像素的光线**

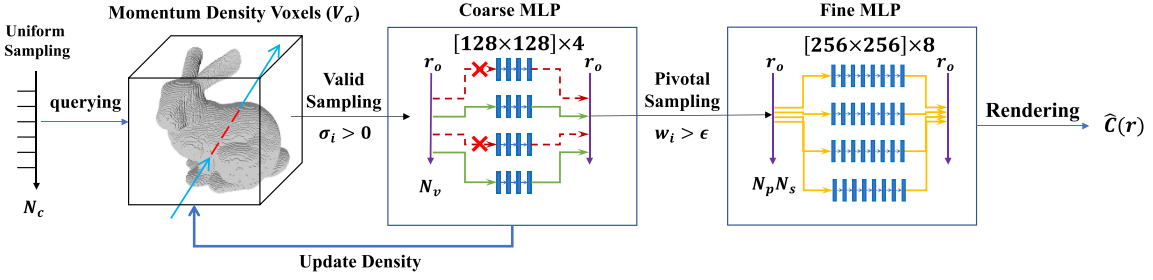
接着将这条光线上点的概率、体密度、对应的rgb颜色值相乘并积分，就得到了我们想要的像素值。但在实际的计算中，连续积分的计算量过大，因此往往如式(2.3)所示对整个过程进行离散化处理，在直线上采样N个点进行计算。在式(2.4)中，代表了采样点间的间隔，为比例系数[13]，为当前点的体密度，为当前点的概率密度，代表当前点的rgb颜色值。

在得到这一点的预测值后，与ground truth 进行L2损失计算,通过式(2.5)获取损失值，其中代表视点达到所有像素所组成的光线，最后利用Adam梯度下降算法优化整个模型的参数。

#### 2.2.2 有效采样（Valid Sampling）和 关键采样（Pivotal Sampling）

考虑到场景中只有大约10%-20%的有效样本，与5%-10%的关键样本，因此EfficientNeRF提出了不同阶段的Valid Sampling[14] 有效采样和 Pivotal Sampling[14]关键采样。

对于有效采样阶段，如图4所示**，**首先在整体架构上构造了一个Momentum density voxels，它用来存储粗采样阶段的中间过程信息，粗采样的Coarse MLP由4层128的全连接层构成，负责进行第一步的特征映射。根据式2.3和2.4中的变量，当为0时，这个点不起作用，因此在计算前首先进行判断，提取出大于0的点送入MLP计算，同时将得到的结果使用动量更新的方式填充Momentum density voxels。这样在迭代的过程中不仅减少了时间开销，还增大了有效点的权重。

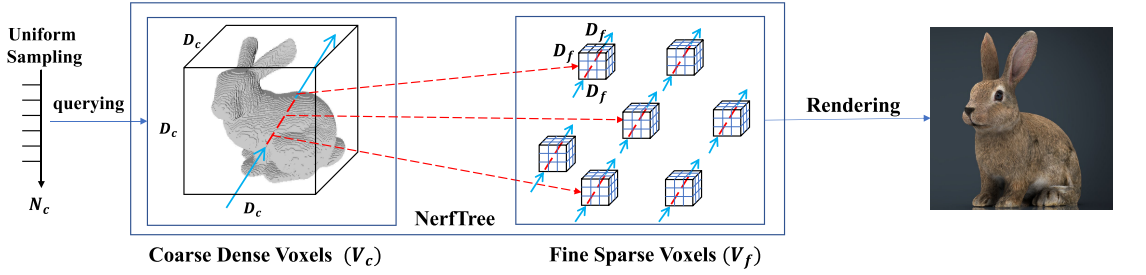


**图4 EfficientNeRF在两阶段采用的不同采样方法示意图**

对于关键采样阶段，如图4所示，细采样的Fine MLP采用了8层256的全连接进行隐式的特征表达。在第一步所得到的有效点基础上再进行一次关键采样，即剔除掉体密度小于给定阈值的点，因为它们对颜色的贡献也非常小，这样就完成了第二步的关键采样。最后将每条光线的颜色值汇总，得到了预测的新视点图像。

#### 2.2.3 缓存机制（Nerf Tree）

上述改进降低了训练的时间开销，如果能够将训练时产生的中间变量存起来用于推理时复用，那么推理渲染的时间也会大大降低。因此EfficientNeRF设计了一个基于八叉树的含有两层节点的NeRF Tree结构，如图5所示，第一层存储粗采样点的rgb和体密度值，第二层存储细采样点的rgb和体密度值，它能够离线存储整个场景，消除了重复的计算，大大加快了渲染的速度。



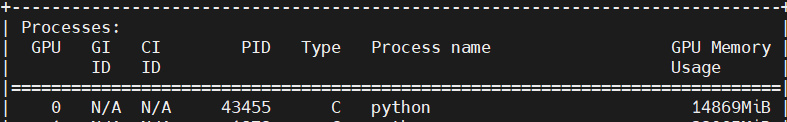
**图5 EfficientNeRF在渲染阶段使用的Nerf Tree原理及结构示意图**

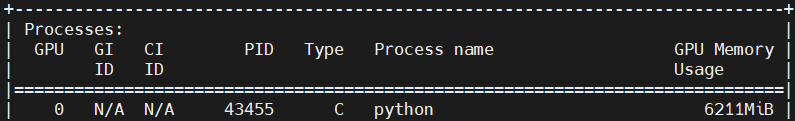
## 3 实验与分析

在本章我将介绍我对NeRF以及EfficientNeRF算法的复现过程以及相应结果的分析。本章将分为两个部分，实验配置、NeRF和EfficientNeRF的结果复现与对比分析。

### 3.1 实验配置

首先是原始的NeRF配置，最初的NeRF在训练时占用4.5G的显存，渲染时占用10G的显存，因此我使用了实验室配备的1080Ti服务器进行训练以及渲染。





**图6 训练时EfficientNeRF所需要的显存开销**

对于这篇文章所改进的EfficientNeRF，如图6所示在读入数据时需要近15G的显存预处理，是一个典型的使用空间换取时间的方法，但训练过程中的显存占用只需要6G，渲染时依旧需要15G左右的显存。所以，对于EfficientNeRF我使用了3090进行训练和渲染。

### 3.2 NeRF和EfficientNeRF复现与对比分析

在这里我参考了两份代码，原始的NeRF-pytorch代码以及EfficientNeRF改进的代码，我分别复现了它们的结果。针对Lego这一场景，使用原始的NeRF和改进的EfficientNeRF进行训练对比，如表1所示，在相同的训练时间下，EfficientNeRF能够迭代更多的次数，相较于NeRF有着更高的效率。同时，PSNR也有了大幅的提升，可见有效点和关键点采样的高效性和正确性。从视觉效果上看，图7可以清晰的看到，在Lego场景的铲斗边缘部分，EfficientNeRF渲染出的图片轮廓更加清晰锐利，进一步证实了有效性。从占用空间情况上看，原始的NeRF权重参数所占空间为14MB，而EfficientNeRF除自身权重的9MB外，还要根据场景的复杂度，附加1-2GB的Nerf Tree缓存结构，通过空间来换取时间。

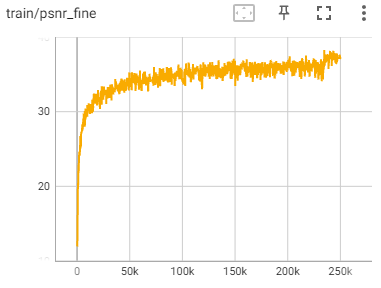
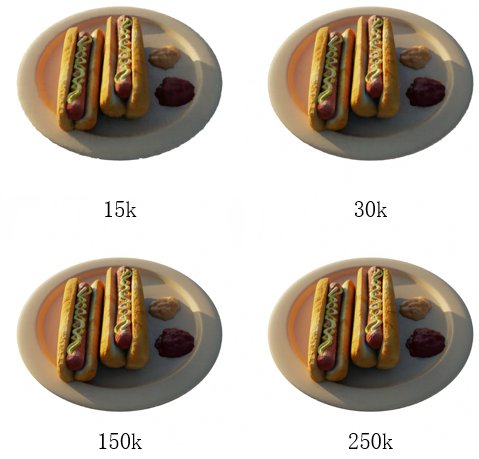


**图7 NeRF和EfficientNeRF对Lego场景的渲染图**

**表1 NeRF和EfficientNeRF在不同场景的性能表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **方法** | **场景** | **PSNR** | **粗采样点** | **细采样点** | **迭代次数** | **训练时间** |
| NeRF | Lego | 27.33 | / | / | 120k | 6.4h |
| EfficientNeRF | Lego | 34.42 | 4.23 | 10.77 | 250k | 6.4h |
| EfficientNeRF | Chair | 33.31 | 3.54 | 5.99 | 250k | 5.7h |
| EfficientNeRF | Drums | 26.69 | 3.16 | 10.57 | 250k | 6.2h |
| EfficientNeRF | Hotdog | 37.31 | 4.57 | 9.35 | 250k | 5.9h |
| EfficientNeRF | Ship | 30.27 | 8.51 | 28.66 | 250k | 8.2h |
| EfficientNeRF | Ficus | 32.80 | 1.97 | 10.03 | 250k | 5.2h |
| EfficientNeRF | Mic | 32.40 | 2.18 | 7.58 | 250k | 4.4h |
| EfficientNeRF | Materials | 31.20 | 4.04 | 10.63 | 250k | 5.1h |

接着我对比了EfficientNeRF对相同的场景在不同训练代数时的PSNR和局部细节的变化，选用Hotdog场景，如图8和表2所示，展示了它们在15k次、30k次、150k次以及250k次训练后的渲染结果。随着迭代次数的增加，渲染图片的盘子边缘细节更加圆滑，锯齿状边缘逐渐消失，PSNR也在稳步地提高。但从整体效果上来看，150k次和250k次的结果差别已经不大，已经可以满足我们的整体需求，模型在训练的后期阶段所学习的正是那些不易学习的局部细节特征。



**图8 相同场景不同迭代次数的渲染结果和PSNR曲线**

对于其余的6个场景，我也进行了训练和渲染，训练的结果数据在表1中，图片展示在图9中。这些训练都是在相同的超参数配置下完成的，超参数配置如下：训练16epoch、250k次迭代、起始128个粗采样点和64个细采样点、粗采样阶段4层128维的MLP、以及细采样阶段8层256维的MLP。通过表中的数据，我们可以得到很多有用的信息。

****



**图9 其余6个场景的渲染结果**

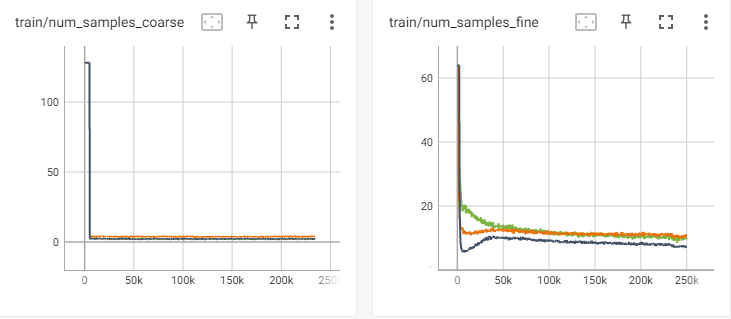
采样点数目与训练时间有着明显的线性关系，采样点数目越多，说明该场景越复杂，比如Ship场景，渲染时所需要的信息就越多，模型整体学习起来就比较困难，所以花费的时间更长。而对于Ficus和Mic场景，这两个场景中的物体只存在于中间部分，整体上偏向圆柱形，在空间中占的比例很小，因此在训练时有效点的个数很少，不需要过于复杂的结构表示，这与我们的直觉相符，能够形象的体现出有效采样和关键采样的高性能。

## 4 改进实验/消融实验与分析

在这一章我对EfficientNeRF存在的一些问题进行了3种不同的修改尝试：修改EfficientNeRF在粗采样阶段的采样方法、修改MLP网络的层数和细采样阶段的采样数目、以及为网络增加Dropout层用来对比模型的拟合效果。

### 4.1 采样方法改进和对比

观察粗采样和细采样阶段的采样点曲线，如图10，展示了ficus、materials和mic场景的采样点下降曲线，发现它们在训练的开始阶段就迅速下降并趋于稳定，虽然这在一定程度上加快了模型的训练以及推理时的速度，但是它不可避免地丢掉了部分信息。原文的EfficientNeRF是在判断当前点在给定方向下的体密度为0时直接删除该点，不再进行计算，但是当我们从不同角度观察该物体时有可能会利用到该点，因此造成了精度的下降。基于这一问题，我对它的采样方法做了一定的改进。



**图10 ficus、materials和mic场景下粗采样和细采样点的变化曲线**

我所采取的解决办法是先设立一个缓冲temp，设定一个阈值次数T，体密度为0的点并不直接删除，而是先进入这个缓冲当中并记录它将要被删除的次数，接着我们继续迭代下去，如果次数达到设定的阈值那么我们再将它删除；相反，如果出现一次不为0的情况，则从缓冲中取出该点，继续当作有效点，因为依旧有某一个视角会利用到这一点，这样在后期使用到它时，依旧会提供一定的信息，增强渲染图片的效果。在这里我通过重写代码NerfTree结构中的query\_coarse、query\_fine、update\_coarse和update\_fine函数的逻辑来实现这一目的，在每次查询之前都先访问缓冲temp获取未被删除、以及删除次数未达到T的点，再从中查询符合我们要求的点。

我选择了较为复杂的drums场景对这个改进点进行测试对比，如图11所示，从数值上来看，修改采样方法后的PSNR为25.09，与最开始的设置降低了1.6；从渲染出的图像质感来看，右侧的图片在鼓面处的透明质感更好，但细节更为模糊，花费7.4h比原始EfficientNeRF的6.2h多了1.2h的训练时间。造成这种现象的原因是：在采用我所尝试的方法时，实际上是原始的NeRF与EfficientNeRF的一个折衷表示，在取样和速度之间取了一个平衡，并没有从根本上解决和提高NeRF神经辐射场的性能。



图11 两种采样方式的模型曲线与渲染图片效果

### 4.2 Dropout层的添加与对比

对于NeRF只能作用于特定的场景这一问题，我对它进行了简要的分析。从整体架构上来看，整个NeRF由两个连续的MLP所构成：粗采样阶段和细采样阶段，它们有着不同的层数以便进行精细度不同的重建学习。对比传统的图像分类、识别等任务，它们的目的是在有限的训练集上训练，之后在从未见过的测试集上进行测试，这些模型普遍具有很强的泛化性，能够处理和判断新的事物。NeRF这种基于MLP的模型，架构上非常简单，如果将这个架构应用于传统任务，很容易就会发生过拟合现象，这在其他任务中都是要尽量避免的情况。

为了探究MLP架构容易过拟合的影响，我在模型的基础上进行了一定的修改。在传统的分类任务中，经常会利用随机的dropout来缓解训练后期的过拟合现象。因此，我修改了nerf.py文件里的NeRF module，分别在粗采样和细采样两个阶段的全连接层后面加入了Dropout层，以用来观察拟合对模型性能的影响。在这里我选择了ship场景进行对比测试，因为这个场景的细节信息相较其他场景更加丰富，能够更好的反映出这种改变所带来的问题。我采用了三种配置来进行对比训练，分别是原始不加Dropout层的结构、每个全连接层后面都加入Dropout的结构、以及只在粗采样和细采样两个阶段的最后加入Dropout的结构。

****

**图12 3种配置下Ship场景的重建效果**

由图12所示，原始EfficientNeRF的PSNR为30.27，增加了Dropout层后，两种修改后模型的PSNR分别为26.45和27.26，整体的视觉效果以及局部的细节都有着一定的下降。通过这种结果，我们可以简单的描述NeRF的本质：即NeRF的最根本思想是通过MLP去隐式的记忆某一个特定的场景，可以将它理解为对特定场景的一个信息压缩，MLP的参数高效的记录了场景的结构。MLP的拟合特性恰恰是这种场景渲染任务所需要的，它依托于拟合的性质，能够根据不同的视角返回精致的重建图像。

### 4.3 MLP层数与关键点采样的改进和对比

在模型训练时，人为规定的超参数会对模型的性能造成非常大的影响，因此为了进一步探究超参数对图片渲染结果的影响，我在原论文给定的超参数设置基础上进行的一定的修改。原论文的结果都是在4层粗采样MLP、8层细采样MLP、5个细采样点以及16个epoch的基础上训练而来，在这一部分，我调整了细采样阶段的层数、细采样点个数以及epoch训练代数，对比不同超参数对模型性能的影响，我选择了较为简单的chair场景对上述的调整进行训练。

**表2 不同配置下对Chair场景的训练**

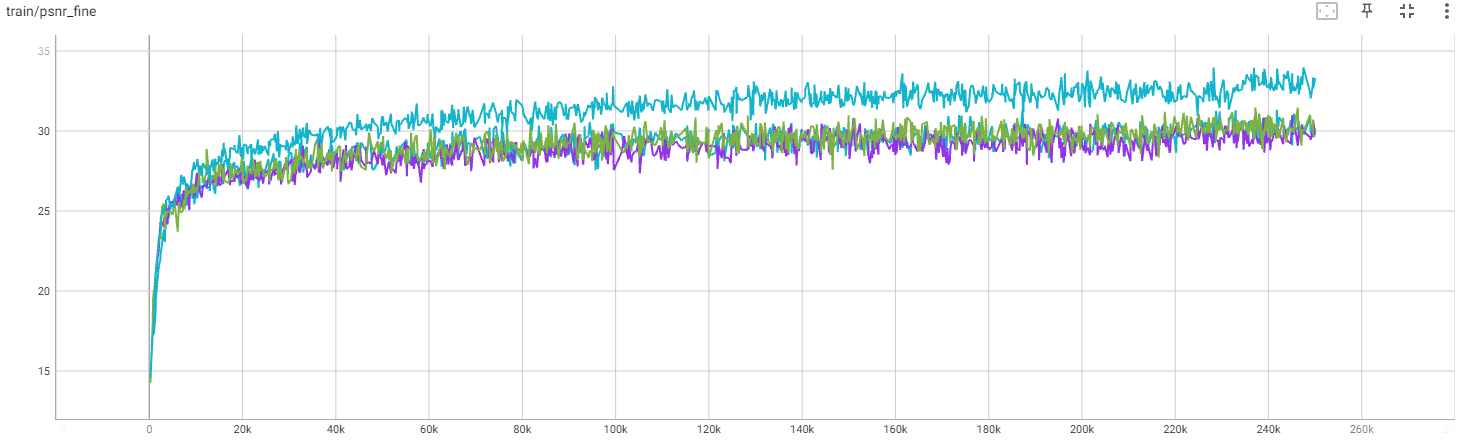
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **方法** | **场景** | **PSNR** | **粗采样层** | **细采样层** | **细采样点** | **Epoch** |
| EfficientNeRF | Chair | 29.65 | 4 | 8 | 10 | 8 |
| EfficientNeRF | Chair | 29.77 | 4 | 8 | 5 | 8 |
| EfficientNeRF | Chair | 30.17 | 4 | 4 | 10 | 8 |
| EfficientNeRF | Chair | 30.37 | 4 | 8 | 10 | 16 |

表2中展示了每个模型的参数配置、最终的PSNR结果、以及相应的训练时间开销，图13则展示了不同的设置所生成的图片效果。首先，在不同的细采样点个数设置下，细采样点个数越多，训练时的时间开销也越大，这与我们预测的结果相符；但是随着采样点的增多，模型渲染图片的质量并没有得到提升，分析原因可能是过多的采样获取到了当前点并不需要的样本，导致渲染时出现了混淆，因此降低了模型的渲染效果。



**图13 4种配置下对chair场景的渲染结果**

不同的细采样MLP层数对渲染的影响反而不大，可能是因为chair场景并不复杂，所以层数少一些的模型也能够很好的压缩场景的信息。如图14所示，随着epoch的提高，渲染结果的PSNR也在稳步地提升，但实际上在训练8-9个epoch后，整体的PSNR在一个稳定的区域内波动，之后也不会有太大的增幅，可以认为对该场景的训练已经达到了拟合的状态，已经学到了它所能学习的场景特征。



**图14 4种配置下chair场景的PSNR训练曲线**

## 5 总结与展望

NeRF神经辐射场的提出为三维体渲染打开了新的方向，成功的将神经网络与三维渲染结合在了一起，并实现了超越传统算法的性能。通过自己做的实验，可以看出NeRF在渲染时能够有着很高的精度，较传统方法的PSNR有了一定的提升。

首先复现了NeRF以及EfficientNeRF的实验结果，接着为了进一步体现MLP的这种隐式的场景建模能力，对EfficientNeRF的网络层以及采样方式进行了一定的尝试修改，包括：

1. 修改EfficientNeRF在粗采样和细采样阶段的采样逻辑。
2. 增加Dropout模块以减轻过拟合现象。
3. 修改隐藏层数目和采样点个数。

整体上来看，NeRF仍旧有很大的提升空间，未来可以从渲染方法、场景渲染时间等方面入手，解决NeRF建模花费时间过长的问题。同时由于NeRF不具有泛化性，对于不同的场景都需要重新训练，而且NeRF不能够应用于含有动态物体的场景，这大大影响了NeRF的灵活性，因此如何解决NeRF的这种弊端也是亟待解决的难题。

## 参考文献

1. Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. https://arxiv.org/abs/2011.13961, 2020.
2. Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. Deformable neural radiance ﬁelds. https://arxiv.org/abs/2011.12948, 2020.
3. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, ECCV, volume 12346, pages 405–421, 2020.
4. Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.: Local implicit grid representations for 3d scenes. In: CVPR (2020)
5. Stephen Lombardi, Tomas Simon, Jason M. Saragih, Gabriel Schwartz, Andreas M. Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. ACMTOG, 38(4):65:1–65:14, 2019.
6. Alex Yu, Vickie Ye, MatthewTancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. https://arxiv.org/abs/2012.02190, 2020
7. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
8. Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In Advances in Neural Information Processing Systems (NeurIPS), volume 33, 2020.
9. Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien P. C. Valentin. Fastnerf: Highfidelity neural rendering at 200fps. CoRR, abs/2103.10380, 2021.
10. Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In arXiv, 2021.
11. Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. CoRR, 2021.
12. Abhishek Kar, Christian H¨ane, and Jitendra Malik. Learning a multi-view stereo machine. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, NeurIPS, pages 365–376, 2017.
13. Nelson L. Max. Optical models for direct volume rendering. IEEE Trans. Vis. Comput. Graph., 1(2):99–108, 1995.
14. CHEN Z Q, ZHANG H. Learning implicit fields for generative shape modeling[C]//2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). New York: IEEE Press, 2019: 5932-5941.
15. Hu T, Liu S, Chen Y, et al. EfficientNeRF Efficient Neural Radiance Fields[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 12902-12911.