1. 数据准备

ModelNet:

ModelNet是普林斯顿大学的一个项目，该项目的目标是为计算机视觉，计算机图形学，机器人学和认知科学领域的研究人员提供全面的物体3D CAD模型。为了构建数据集的核心，该项目使用从SUN数据库中获得的统计数据编译了世界上最常见的对象类别列表。之后通过查询每个对象类别术语，使用在线搜索引擎收集属于每个对象类别的3D CAD模型。该项目提供了10类子集和完整的数据集供下载。

ShapeNet:

ShapeNet是一个不断努力建立一个丰富注释的3D形状的大型数据集，该数据集为世界各地的研究人员提供3D模型数据，以便进行计算机图形学，计算机视觉，机器人学和其他相关学科的研究。 ShapeNet 根据WordNet层次结构进行组织。WordNet中的每个有意义的概念，可能由多个单词或单词短语描述，被称为“同义词集合”或“synset”。 WordNet中有超过10万个同义词，其中大多数是名词（80,000+）。

ShapeNet由几个不同的子集组成：

ShapeNetCore:

ShapeNetCore是整个ShapeNet数据集的一个子集，具有单个3D模型和手动验证类别和对齐注释。它涵盖55个常见对象类别，其中约有51,300个独特的3D模型。 PASCAL 3D +这一流行的计算机视觉3D基准数据集的12个对象类别全部由ShapeNetCore覆盖。

ShapeNetSem:

ShapeNetSem是一个更小，更密集的注释子集，由12,000个模型组成，分布在更广泛的270个类别中。除了手动验证的类别标签和一致性对齐之外，这些模型还用现实世界的维度进行了注释，对类别级别的材料组成进行了估计，并对其总体积和重量进行了估计。

本文使用了ShapeNetCore-55数据集。

PCL:

PCL(Point Cloud Library)是一个独立的大型开放项目，其被应用于2D图像和3D点云的处理。

PCL是在吸收了前人在点云相关研究的基础上建立起来的大型跨平台开源C++编程库，它实现了大量与点云相关的通用算法和高效数据结构，涉及到点云获取、分割、滤波、检索、配准、识别、追踪、特征提取、曲面重建、可视化等。支持各种主流操作系统平台，如Linux、Windows、Mac OS X、Android以及部分嵌入式系统。其在3D信息获取与处理领域上具有与OpenCV在2D信息获取与处理领域中同等的地位，PCL可以免费进行商业和学术应用。

本文的实验通过PCL中的库函数实现.off格式文件向八叉树的转变。

CNN(卷积神经网络):

CNN（Convolutional Neural Network）是一类深度前馈人工神经网络，已成功应用于分析视觉图像。CNN使用一种多层感知器的变体设计来获取最少的预处理成本。基于CNN共享权重结构和平移不变性等特征，CNN也被称为移位不变或空间不变式人工神经网络（SIANN）。CNN受到生物中神经元之间的连接模式的启发（类似于动物视觉皮层的组织）。个体皮层神经元只在被称为感受域的视野受限区域对刺激作出反应。不同神经元的感受域部分重叠，从而覆盖整个视野。与其他图像分类算法相比，CNN使用相对较少的预处理。这意味着网络学习传统算法中的过滤器是手工设计的。这种独立于先前知识和人力设计的特点是一大优势。CNN在图像和视频识别，推荐系统和自然语言处理中有着广泛的应用。

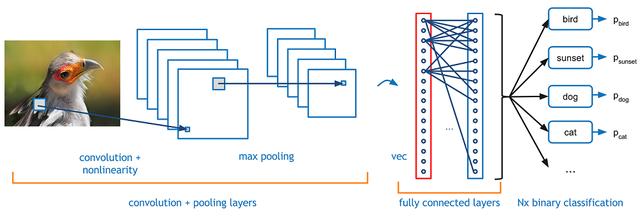
CNN一般由以下几个层组成（图1）：

图1.一个卷积神经网络的简单例子

卷积层

卷积层是CNN的核心构件。该图层的参数由一组可学习的过滤器（或内核）组成，这些过滤器具有较小的接受区域，但会延伸到输入区域的全部深度。在正向过程中，每个过滤器都卷入输入体积的宽度和高度，计算过滤器条目和输入之间的点积，并生成该过滤器的二维激活图。因此，网络会在输入中的某个空间位置检测到某种特定类型的特征时，学习激活的过滤器。沿着深度维度堆叠所有滤波器的激活图形成卷积层的全部输出体积。因此，输出体积中的每个条目也可以被解释为神经元的输出，该输出查看输入中的小区域并与同一激活图中的神经元共享参数。

池层

CNN的另一个重要概念是汇集，这是一种非线性下采样。有几个非线性函数来实现池，其中最大池是最常见的。它将输入图像分割成一组不重叠的矩形，并为每个这样的子区域输出最大值。直觉是一个特征的确切位置并不比其他特征的粗略位置重要。池化层用于逐渐减小表示的空间大小，以减少网络中的参数数量和计算量，并因此控制过拟合。周期性地在CNN体系结构中的连续卷积层之间插入池层是常见的。汇集操作提供了另一种形式的翻译不变性。池层在输入的每个深度切片上独立运行，并在空间上调整其大小。最常见的形式是一个大小为2x2的滤波器的汇聚层，在输入的每个深度切片上沿着宽度和高度加上2个下采样的步幅2，放弃75％的激活。在这种情况下，每个最大操作超过4个数字。深度维度保持不变。除了最大汇集之外，汇集单位还可以使用其他功能，如平均汇集或L2规范汇集。平均汇总通常用于历史上，但最近已经不受欢迎，与最大汇集相比，这在实践中效果更好[34]。由于代表大小的大幅度减少，趋势是朝着使用更小的过滤器[35]或完全丢弃合并层[36]。

ReLU层

ReLU是整流线性单位的缩写。此层应用非饱和激活函数。它增加了决策函数和整个网络的非线性特性，而不影响卷积层的接受域。

虽然其他函数也能用来增加非线性，例如饱和双曲正，但ReLU通常比其他函数更受欢迎，因为它可以快几倍地训练神经网络[39]，而不会对泛化精度造成严重损失。

全连接的层

最后，在几个卷积和最大池层之后，神经网络中的高层推理通过完全连接的层进行。正如在常规神经网络中看到的，完全连接层中的神经元与前一层中的所有激活都有关联。因此，它们的激活可以通过矩阵乘法和偏移偏移来计算。

损失层

损失层指定训练如何惩罚预测标签与真实标签之间的偏差，并且通常是最后一层。那里可以使用适合不同任务的各种损失函数。 Softmax损失用于预测单个K类互斥类。 Sigmoid交叉熵损失用于预测其中的K个独立概率值。欧几里得损失用于回归到实值标签。

Caffe:

为了进行CNN操作，实验使用了Convolutional Architecture for Fast Feature Embedding(Caffe)框架。Caffe是一个清晰，可读性高，快速的深度学习框架，在视频、图像处理方面有着广泛的应用。

LMDB:

实验中需要将数据集的3D数据转化放入LMDB中使用，LMDB（Lightning Memory-Mapped Database），是一种以键值存储的形式提供高性能嵌入式事务的数据库。 LMDB将任意键/数据对存储为字节数组，具有基于范围的搜索能力，支持单个键的多个数据项，并且具有在数据库末尾添加记录的特殊模式（MDB\_APPEND），可以提供显着的写入性能增加超过其他类似的产品。 LMDB不是一个关系数据库，它完全通过关键值存储。LMDB也可以在多线程或多处理环境中同时使用，读取性能按设计线性缩放。 LMDB数据库一次只能有一个写入器，但与许多类似的键值数据库不同，写入事务不会阻止读取器，读者也不会阻止写入器。 LLMDB不需要事务日志（因此不需要两次写入数据就可以提高写入性能），因为它通过设计本身维护数据完整性。

1. 八叉树

给定一个三维模型（例如面向三角网格或面向法线的点云），首先构建输入三维模型的八叉树(图2)，并在八叉树中打包CNN操作所需的信息。 在这个八叉树结构的帮助下，所有的CNN操作都可以在GPU上高效地执行。

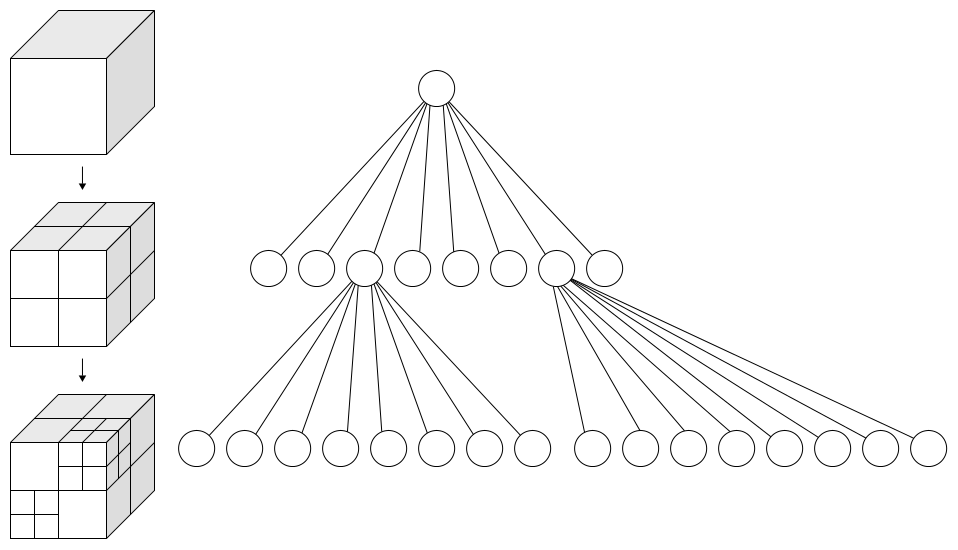


图2(a)



图2(b)

图2.(a)八叉树构造原理(b)三维物体的八叉树表示

八叉树构造

为了构建输入三维模型的八叉树，首先将三维形状均匀缩放到一个轴对齐的单位三维边界立方体中，然后以广度优先的顺序递归地细分三维形状的边界立方体。在每一步中，遍历在当前深度I处由3D形状边界占据的所有非空八分区，并在下一个深度l + 1处将它们中的每一个细分为八个子八分区。重复该过程直到预定义的八叉树深度达到标准。图2显示了为二维图像构建的深度为2的四叉树（二维物体标记为红色）。

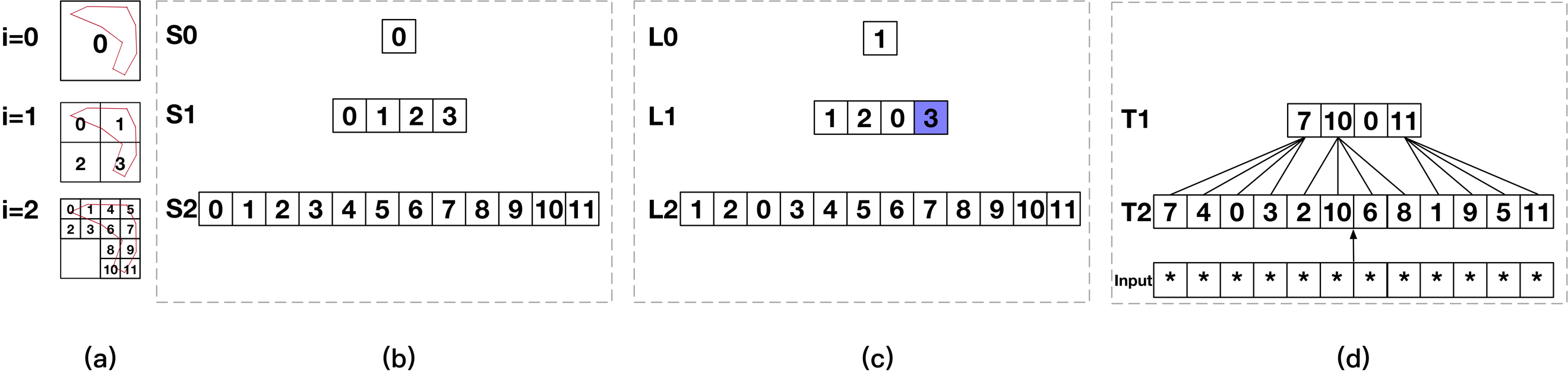


图2. 二维四叉树图

(a) 给定以红色标记的输入2D形状，构造一个深度为2的四叉树。未被形状占据的正方形是空节点。四元节点内的数字是它们的随机值。

(b)随机值矢量S l, l = 0...2，其中每一个向量存储该深度处的四元节点的随机值。

(c) 标签向量L l, l = 0...2，其中每一个向量存储该深度的四元节点的标签。对于空节点，它们的标签被设置为零。对于非空节点，节点的标签p指示它是该深度处的第p个非空节点。标签缓冲区用于查找从父节点到其子节点的对应关系。

(d) 对输入信号的CNN卷积结果存储在特征映射T 2中。当通过CNN操作（例如汇集）对T 2进行下采样时，下采样结果被分配给T 1的第1，第2和第4个入口。使用该深度处的标签向量L I来找到两个深度处的节点之间的对应关系。

构建八叉树后，收集每个八分区中CNN操作所需的一组属性，并将它们的值存储在八叉树中。具体来说，本文计算一个随机值[Wilhelms and Van Gelder 1992]和八叉树中每个八分区的标签。同时，本文的方法从存储在最佳叶节点中的3D形状中提取CNN的输入信号，并将结果CNN特征记录在每个八分区中。如图2所示，本文按照深度顺序组织存储在八叉树中的数据。在每个深度处，本文按照其键值的升序对八分区进行排序，并将它们的属性值打包成一组一维属性向量。所有属性向量共享相同的索引，并且向量的长度是当前深度处的八分区的数量。

随机值。

深度为l的八分区O的随机值使用独特的l个3比特的字符串组（O）：= x1y1z1x2y2z2...xlylzl来编码其在3D空间中的位置，其中每个3比特组xi，yi，zi∈{0,1}定义了它在其父八分区的三维立方体中的相对位置。八分区O的整数坐标（x，y，z）由x =（x1 x2 ... xl），y =（y1 y2 ... yl），z =（z1 z2 ... .zl）确定。按照升序将它们的所有八分区按照它们的随机值排序，并将所有八分之一的随机值以l深度存储在随机值矢量Sl中，其随后用于构建3D的八分相邻域卷积。在本文的实现中，每个随机值都存储在一个32位整数中。图2（a）说明了所有四叉树节点的随机值，图2（b）演示了每个深度的相应随机值阵列。

标签。

在CNN计算中，池化操作经常用于将在第l深度八分之一处计算的特征下采样到第（l - 1）深度处的其父亲八分区，因此需要快速找到相邻深度处八分体的父子关系。为此，本文为第l层深度处的非空八分体分配标签p，这表明它是第l层深度的排序八分体列表中的第p个非空八分体。对于空八分区，简单地将其标签设置为零。l深度处所有八分区的标签存储在标签矢量Ll中。图2（c）说明了四叉树每个深度的标签向量。对于非空四边形节点，其标签为3（L 1 [2] = 3），因为它是第一个深度处的第三个非空节点，而具有随机值0的节点是第一个，具有随机值1的节点是第二个。给定一个非空的节点，其索引j在l深度处，第（l + 1）深度处第一个子节点的索引k = 8×（L l [j] - 1） 。这是基于两个推断。首先，只有第一层深度的非空八分区被细分。其次，由于我们按照其洗牌键的升序对八分区进行排序，八分区的八个子八分区将被顺序存储。此外，子八分区和他们的非空父母在他们自己的属性向量中遵循相同的顺序。如图2（c）所示，第二深度的最后四个节点由第一深度的第三个非空节点（标记为蓝色）创建。

输入信号。

本文使用在最优叶子八分区处计算的平均法向量作为CNN的输入信号。对于空叶子八分区，简单地分配一个零矢量作为输入信号。对于非空叶子八分区，用一组点来对嵌入叶子八分区中的三维形状表面进行采样，并将所有采样点的法线作为该叶子八分区处的输入信号进行均值处理。将所有叶子八分区的输入信号存储到输入信号矢量中。矢量的大小是八叉树中最优叶子八分区的数量。

与许多基于体素的CNN方法中使用的二元指示函数[Maturana and Scherer 2015; Wu等人2015]相比，正常信号非常稀疏，即表面上只有非零值，并且在最优叶子八分区中采样的平均法线能更好地表示局部3D形状的方向，并为CNN提供更精确的3D形状信息。图3比较了一个体素化的3D模型和一个八叉树表示的由在叶八分区取样的方向圆盘渲染的相同3D模型，其中叶八分区的大小与三维像素大小相同。如图所示，八叉树表示（c）比基于体素的表示（a）更接近真实3D形状（b）。

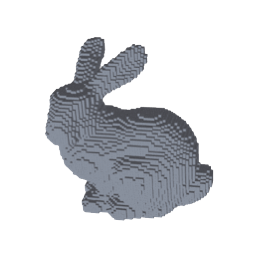
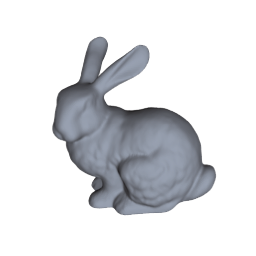
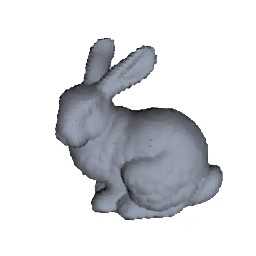
  

图4.对同一个3D模型的不同表示

(a)基于体素的表示

(b)原模型

(c)基于八叉树的表示

CNN特点。

对于在l深度处定义的每个3D卷积核，在特征映射向量T l中记录l深度处的所有八分圆上的卷积结果。

最小批3D模型组。

对于CNN训练中使用的最小批的一组3D对象，其八分区不同。为了支持GPU上的高效CNN训练，将这些八叉树合并为一个超八叉树。对于每个八叉树深度l，将所有3D对象的属性向量（S l，L l和T l）连接到超八叉树的S l \*，L \* l和T l \*之后，通过使用每个随机值的最高8位来更新S l \*中的随机值来存储对象索引。接着更新超八叉树中的标签向量L \*以表示整个超八叉树中每个非空八分区的索引。之后，超八叉树可以直接用于CNN训练。

1. 八叉树上的CNN操作

CNN中最常见的操作是卷积，合并以及反向解卷积和解卷积。 借助八叉树数据结构，所有这些CNN操作都可以在GPU上高效实现。

3D卷积

为了将卷积运算符应用于一个常量，需要在相同的八叉树深度处选取相邻的八分区。 为了有效地计算卷积，本文用展开的形式写出卷积运算符Φc：

这里O ijk代表O的相邻八分区，T（·）代表与O ijk相关的特征向量。 而T（n）（·）代表（n）个第n个信道，W ijk是卷积运算的权重。 如果O ijk不存在于八叉树中，则T（O ijk）被设置为零矢量。 在这种形式中，卷积运算可以转换为矩阵乘积[Chellapilla et al。2006; Jia等人 2014]并在GPU上高效计算。

内核大小为K的卷积运算符需要访问八分区的K 3 - 1个相邻八分区。一种解决方案是预先计算并存储邻近信息。但是，CNN通常是在以Batch为单位进行训练的。当K非常大时，这会导致大量的I / O处理和大量内存占用。需要建立一个散列表：H：key（O）→7索引（O）以方便搜索，其中索引（O）记录S在O中的位置。因为散列表的分期时间复杂度是恒定的，所以这个选择可以被视为CNN的计算成本和内存成本的平衡。给定存储在矢量S 1中的随机值，可以恢复八分区的整数坐标（x，y，z），然后可以在恒定时间内容易地计算相邻八分区的坐标，其相应的随机值也是如此。给定相邻八进制数的随机值，并行高效地搜索哈希表以得到它们的索引，根据该索引收集相邻数据信息，然后可以应用卷积运算。

如果卷积步长为1，则将上述操作应用于当前八叉树深度处的所有现有八分区。对于每个八分区，哈希表将被搜索K 3 - 1次。然而，同一亲本下的八个同胞八分相邻域有很多重叠，并且只有（K + 1）3个八分区包括八个八分区。通过搜索八个兄弟八分区中的每一个的（K + 1）3-8个相邻八分区，可以进一步加速邻域搜索。具体而言，如果内核大小为3，则此优化可以将相邻搜索操作加速两倍以上。

使用2的步长进行卷积也是非常容易的。具体地说，对于同一父亲下的8个同胞八分区，卷积可以应用于第一个同胞八分区，而忽略其他同胞，这相当于下采样特征映射的分辨率为2倍。对于步长为2r（r> 1）的卷积，可以将操作应用于属于高度为r的每个子树的第一个八分区，然后特征地图将被下采样2 r（r> 1）。由于八叉树的特殊层次结构，卷积的步幅被限制为2的整数次幂。然而，在CNN文献中，任意步长的卷积是不常见的。

当以大于1的步幅执行卷积操作时，下采样并且缩短数据向量T的长度。让数据从下到上流动。然后存储在L l中的信息可以用来获得对应关系。以图2所示的例子为例，T 2的初始长度为12.当下采样发生时，矢量的长度将为3.但是在八叉树的第一深度处有四个八分区，并且长度的T 1应该是4.结合存储在L 1中的信息和下采样矢量，T 1可以容易地更新。

Pooling

合并的主要功能是逐步缩小表示的空间大小。池层在特征地图的每个通道上独立运行，并在空间上调整其大小。最常见的形式是应用跨度为2的内核大小为2的过滤器的最大池化层。在本文的八叉树结构上应用池是非常方便的。由于同一父项下的每个8个同胞八分区都连续存储，因此在八叉树上应用最大池操作符可以减少从每8个连续元素中挑选出最大元素，这可以在GPU设备上高效实现。然后，将特征映射的分辨率下采样2倍，并且来自父躯干的信息可用于指导进一步的操作。

由于池在实践中可以看作是一种特殊的卷积，因此按照前面介绍的方法对其他内核大小或步长大小进行一般池化操作，即找到相应的相邻八分区并应用指定的操作，如最大池或平均池。

Unpooling

Unpooling操作是合并的逆操作并执行上采样，这种操作在CNN可视化[Zeiler and Fergus 2014]和图像分割中被广泛使用[Noh et al。 2015年]。max-unpooling操作通常与max-pooling操作一起使用。在应用max-pooling操作之后，每个池区中最大值的位置可以记录在以连续阵列存储的一组开关变量中。相应的最大解卷积操作利用这些开关将当前特征地图的信号放置到上采样特征地图的适当位置中。图5展示了max-unpooling如何在四叉树上工作。由于八分区的连续存储，本文可以重新使用为基于图像的CNN开发的高效的解耦实现。

解卷积

解卷积算子也被称为转置卷积和反向卷积[Long et al。 2015年; Zeiler and Fergus 2014]，可以用来放大和加密激活图，这可以通过反转卷积的前向和后向来实现。基于先前提出的八分区上的卷积，可以相应地实施解卷积操作。

与在全空间（包括空白区域）执行CNN操作的基于全体素的CNN不同，本文方法中的CNN操作仅应用于八分区。也就是说，哪里有八分区，哪里才有CNN计算。将信息传播到空白区域并通过空白区域交换信息不是必需的，并且需要更多的内存和计算。通过限制八叉树中的信息传播，形状信息可以沿形状更有效地交换。虽然在这一点上没有理论证明，但在实验中证实了此方法的优点。

通过将数据存储和CNN计算限制在八分之一中，基于八叉树的CNN的存储和计算成本为O（n 2），其中n是最佳级别的每个维度中的体素分辨率。相反，基于全体素的解决方案的记忆和计算成本是O（n 3）。此外，由于所有数据均连续存储在内存中，因此O-CNN与常规网格上定义的2D和3D CNN共享相同的高性能GPU计算。训练期间在第k次迭代中需要的散列表和邻居信息可以在单独的线程的第（k-1）次迭代中被预先计算，这不会导致计算时间延迟。在5.1节中进行详细的评估和比较。

1. 网络结构

CNN的网络结构近年来发展迅速。越来越广泛的网络在完成许多任务方面显示出其优越性。现有的3D CNN已经使用不同的网络来增强其功能。但是，这使得很难区分他们方法的主要优点来自哪里。为了清楚地展示本文基于八叉树表示的优点，本文设计了一个简单的网络，遵循LeNet的概念[Lecun et al。 1998]。

O-CNN

反复在八叉树数据结构上从下到上应用卷积和合并。使用ReLU函数（f：x∈R→7 max（0，x））来激活输出并使用批量归一化（BN）来减少内部协变量[Loffe and Szegedy 2015]。操作序列“卷积+ BN + ReLU +pooling”为一个基本单元，并且如果卷积应用于第1个深度八分区，则用U1表示它。 U l的特征映射的通道数设置为2 max（1,9-l），卷积核大小为3.O-CNN由以下形式定义：

input → U d → U d−1 → · · · → U 2

称之为O-CNN（d）。为了对齐来自不同八叉树结构的所有特征，强制所有第二深度八分区存在，并在第二深度的空八分区上使用零矢量填充。

用于形状分析的O-CNN。

对于对象分类，添加两个完全连接（FC）图层，一个softmax图层和两个Dropout图层[Srivastava et al。 2014]在O-CNN（d）之后，即

O-CNN（d）→Dropout→FC（128）→Dropout→FC（N c）→softmax→输出。

这里128是FC中的神经元数量，N c是分类类别的数量。 Dropout用于避免过度拟合。

1. 实验结果分析

物体分类：

训练细节

通过动量为0.9，重量衰减为0.0005，批量为32的随机梯度下降（SGD）来优化O-CNN。丢失率为0.5。初始学习率为0.1，每10个循环减少10倍。大约40个周期后停止优化。网络的超参数在形状分类和检索实验中是固定的。它们在具有少量形状的类别的对象分段中进行了微调。

八叉树数据准备。

对于物体分类的任务，3D训练数据集主要来自ModelNet40 [Wu et al。 2015]和ShapeNet-Core55 [Chang et al。 2015]，其中包含大量的三角形网格与各种形状。ModelNet中的许多网格包含很多伪像翻转法线，非流形结构和重叠三角形。因此，为了用正确的正常信息构建八叉树数据结构，首先使用ray shooting算法来从形状中采样具有定向法线的密集点。具体来说，在对象的截断边界立方体的面心上放置14个虚拟相机，从每个方向向对象均匀地拍摄16k条平行光线，计算光线与表面的交点，并定位表面点的法线朝着相机。放弃形状不可见部分的点。然后，在点云上构建八叉树结构，并计算叶子八分区内点的平均法向量。所有形状的八叉树结构都存储在数据库中并保存在硬盘上。

对象分类的目标是将类别信息分配给每个对象，这是理解3D形状的基本任务。

数据集。

使用ModelNet40数据集[Wu et al。 2015]进行训练和测试，其中包含来自40个类别的12,311个CAD模型，并且使用多类标签进行了良好的注释。数据集中提供了训练和测试集，其中9,843个模型用于训练，2,468个模型用于测试。数据集中模型的直立方向是已知的。通过将每个模型沿直立方向均匀旋转来为每个模型生成12个姿势来增加数据集。

训练细节。

为了观察O-CNN在不同分辨率下的行为，本文训练了六个网络：O-CNN（3），O-CNN（4）和O-CNN ，O-CNN（5），O-CNN（6），O-CNN（7），O-CNN（8），即叶腋的分辨率分别为8 3，16 3，32 3，64 3，128 3 ，256 3。损失函数被建模为通常用于分类的交叉熵。

定位池。

由于每个模型都被旋转到12个姿势，在测试阶段，每个姿势的输出层的激活可以汇集在一起​​以提高预测的准确性。在这种情况下，定向池被缩减为投票方式，这已被[Maturana和Scherer 2015]采用。更有效地，还可以汇集最后一个卷积层的激活，然后微调最后两个FC层。这一策略已被[Qi et al。 2016年Su等人2015年]。但是这种方法的缺点是需要对另一个神经网络进行训练。为了简单起见，选择投票策略进行分类。在表1中，提供进行voting和没有voting的结果。

比较和讨论。

本文用最先进的3D CNN做了分类准确性的比较：VoxNet [Maturana和Scherer 2015]，SubVolSup [Qi et al。 2016]，FPNN [Li et al。 2016]，PointNet [Qi et al。 2017]，几何图像[Sinha et al。 2016]和VRN [Brock et al。 2016年]。为了公平比较，只考虑单一CNN的表现，并忽略CNN集合的结果。

从表1发现本文的O-CNN比VoxNet和FPNN有明显的优势。分辨率为16 3的O-CNN（4）已经比FPNN（64 3）好，并且比使用距离场和正常场信息的FPNN + normal（64 3）稍差。与非基于体素的方法相比，O-CNN（4）仅比PointNet差。当增加分辨率时，O-CNN（5）击败所有其他方法。通过投票策略，O-CNN只比使用24个轮换副本进行训练和投票的VRN差。

有趣的是，O-CNN（3）已经具有很好的准确性，高于85％。这一事实与人类的认知是一致的：人们可以很容易地从远处识别3D形状的类型。结果表明本文的具有形状正常信息的八叉树信息量很大。

本文的O-CNN的准确性随着决议逐渐增加。但是当分辨率超过64 3时，精度下降很小。这可能是因为ModelNet40数据集仍然不够大，而且当网络更深时，训练期间会出现一些过度适应的风险。

表示比较。为了进一步验证在正常信号下使用八叉树结构的优越性，使用与O-CNN（5）相同的网络架构进行了实验，同时使用不同的输入表示法。表示变化包括：（1）具有二进制信号的完整体素; （2）在空体素中具有零矢量的正常信号的全体素; （3）八叉树与正常信号，即本文的O-CNN的表示; （4）八叉树与二进制信号进行比较，即用占用比特代替正常信号。第二种变化可以看作是具有正常信号的八叉树的广义版本。对于在测试中具有完整体素表示的CNN，为网络架构调整了VoxNet [Maturana和Scherer 2015]的实施。在ModelNet40数据集上训练和测试网络的结果汇总在表2中。

需要注意的事，正常信号有助于在八叉树和全三维像素结构上实现更好的性能，这验证了本文在3.1节中的声明，即正常信号保留了原始形状的更多信息，并且优于二进制信号。

此外，具有正常信号的八叉树提供了所有方法中最高水平的准确性，而具有正常信号的完整体素在任何地方执行计算都不会产生更好的结果。这表明仅限制八分之一的计算是一个合理的策略，这导致O-CNN的良好性能。本文对未来的研究留下了严格的理论分析。

最后，本文发现具有二进制信号的八叉树比具有二进制信号的完整体素结构具有更差的性能。这是因为表示原始形状的指标函数是在一个体积中定义的，而八叉树是从点云中构建的。在将正常信号替换为占用位后，相当于丢弃指示函数的内部部分，与全部体素表示相比造成信息损失，并且使得难以区分对象的内部和外部。

内存和计算效率的比较。

将O-CNN的记忆和计算成本与全体素CNN的成本进行了比较。与在表示比较中进行的测试类似，本文的网络结构来完成基于体素的CNN。值得注意的是，在本文的网络结构中，随着八叉树深度增加，O-CNN中使用的每个深度处的特征映射的信道数量减少2倍，O-CNN（d）的存储器开销可减少为O （n），这使得O-CNN能够以高分辨率分析3D形状。

实验中运行1000次向前 - 向后迭代，包括所有CPU-GPU通信，计算每次迭代的平均时间，并记录峰值GPU内存消耗。表3和表4总结了内存和时间成本的统计数字。很明显，O-CNN在所有分辨率下运行得快得多，并且当分辨率大于16 3（即d≤4）时占用较少的内存。

CNN可视化。

关于用于图像理解的CNN，已知的是所学习的卷积滤波器的输出[Goodfellow et al。 2016]会在出现重要图像功能时激活。在O-CNN上也观察到了这种现象，这有助于更好地理解O-CNN。

在图6中，通过对相应八分区上的输入形状的响应进行颜色编码，来说明O-CNN（5）的U5，U4，U3中的一些滤波器的表现。U5中的滤波器捕捉到形状的低级几何特征，而U4和U3中的滤波器捕捉到高级形状特征。在U5中，滤波器A倾向于捕获前向平面区域，滤波器B倾向于大的圆形区域。在U 4中，滤波器C和D捕获更多的全局形状特征。在U 3中，滤波器E和F对形状类别更敏感，并且当形状的类别不适合时，它们几乎没有响应。

形状检索：

如今，3D模型变得更加广泛可用，为了管理和分析它们，3D形状检索方法是必不可少的。本文在大型数据集ShapeNet Core55上测试本文的O-CNN，并与3D形状检索比赛中的最新方法进行比较--SHREC16 [Savva et al。 2016年]。

数据集。 ShapeNet Core55数据集共包含51190个3D模型，包含55个类别和204个子类别。模型标准化为单位长度立方体，并具有一致的直立方向。数据集的70％用于培训，10％用于验证，20％用于测试。本文使用与对象分类相同的方法来执行数据增强。

恢复。形状检索的关键是为每个形状生成一个紧凑的信息特征，可以检索出最相似的形状。本文训练一个O-CNN作为特征提取器，网络结构与用于分类的网络结构相同。在训练阶段，只有类别信息才能使交叉熵损失函数最小化。为简单起见，丢弃数据集中的子类别信息。 O-CNN输出是输入形状的类别概率，用作每个形状的特征向量。由于每个对象都旋转到12个姿势，相应地有12个特征向量，利用这些特征向量以前呈现的方向池用于为每个形状生成一个特征向量。对于每个查询形状，可以从该特征矢量预测标签。通过收集具有相同标签的所有形状，然后根据查询形状与检索到的形状之间的特征矢量距离对它们进行排序来构造查询形状的检索集。图7显示了O-CNN的三个模型的前5个检索结果（6）。

性能比较。对于测试集中的每个查询，都会返回一个检索列表。五个指标用于评估结果的质量：精确度，召回率，mAP，F分数和NDCG。入口处的精度是与此入口相关的检索实例的一小部分，而召回是相关实例的一小部分，直至此入口。粗略地说，召回随着检索列表的长度而增加，而精度降低。 mAP是平均精确度，F-score是精确度和召回率的调和平均值，可以看作是精确度和召回率的总结。 NDCG测量排名质量，并在计算此度量时考虑形状之间的子类别相似度。这四个指标上的所有查询形状的平均性能是使用[Savva et al。 2016]，并总结在表5中。精确查全曲线如图8所示。

本文比较了本文的O-CNN和SHREC16中五种最先进的方法，这些方法都基于基于多视图的CNN [Bai et al.2016; Savva等人。 2016年Su等人2015年]。从表5本文可以看出，本文的O-CNN（5）与最先进的结果相当，并且O-CNN（6）在所有测试方法中产生最佳结果。此外，尽管在训练本文的网络时子类别信息被丢弃，但本文也在NDCG上获得最优分数，这表明，用本文的八叉树表示，学习的特征是非常区分性的，并且可以很好地区分相似的形状。

1. 总结

本文提出了基于八叉树的卷积神经网络（O-CNN），它利用八叉树表示的稀疏性和形状的局部方向来实现紧凑的存储和快速计算，实现了与现有工作更好或相当的性能。关于三种形状分析任务的实验证明了O-CNN的功效和效率。预计O-CNN将推进更多关于3D理解和处理的工作，包括零件分析，同类检索等。

未来， O-CNN有望被用来解决更多的形状分析和处理难题，尤其是对于高分辨率的O-CNN必不可少的细粒度任务，如形状去噪，形状对应，物体生成和场景分析。改进O-CNN的方法也很多，如下所示：

自适应八叉树。

在本文的八叉树结构中，不考虑形状的几何变化。实际上，对于几乎平坦的区域，使用更大的八分区来表示它们而没有细分是很好的。通过根据局部几何结构自适应地构造八叉树，可以进一步提高O-CNN的计算和存储效率。

一般性点阵。

在本文的O-CNN中，在八叉树数据结构中组织3D数据和计算，这可以看作是一个层次稀疏网格。 有可能建立基于其他晶格的类似的层次结构，如四面体晶格[Graham 2015]和超晶格[Jampani et al。 2016]，后者可用于更高维度的CNN。

网络结构。

尽管CNN的结构在改善性能方面起着重要作用，但本文没有实验过深度残留网络或递归神经网络等先进结构。 未来，如果O-CNN与这些先进的网络结构相结合，将可能完成更具挑战性的任务。