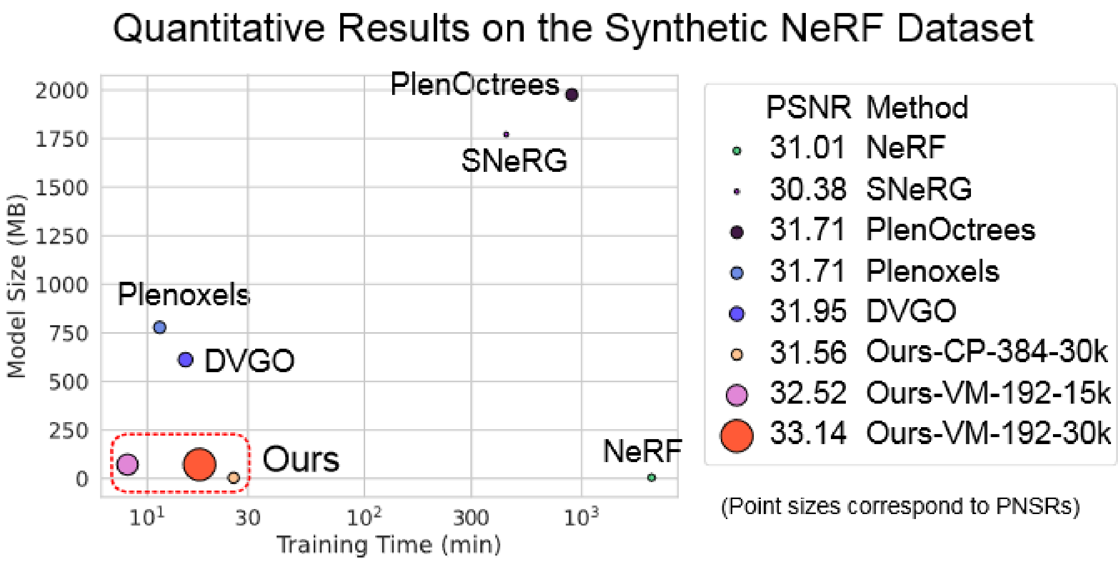
TensoRF-张量辐射场论文笔记

> TensoRF: Tensorial Radiance Fields (ECCV 2022)

跟NeRF使用多层感知机隐式建模场景表达的方式不同，TensoRF将场景建模为一个四维的张量，张量中的每一项代表了一个体素，体素内包含了体积密度和多维的特征信息

论文的中心思想是使用张量分解技术，将4D张量分解成多个低秩的张量分量，以小见大



从上图中可以看出，张量辐射场可以达到：

1. 更好的质量

2. 更快的速度

3. 更小的模型体积

张量辐射场除了渲染质量更好之外，与同时期使用体素方式的研究相比占用更少的内存使用

张量辐射场在30分钟内就可以完成重建，并且模型的大小小于4M，这比NeRF更快，以及更小巧

使用VM分解方式的可以达到10分钟的时间，以及更好的质量，模型大小小于75M

TensoRF是第一个从张量的角度来看待辐射场建模，并提出了辐射场重建作为一个低秩张量重建的问题

张量分解

论文中的使用的张量分解技术是通用的，论文中使用了CP分解和VM分解，当然也可以尝试使用其他的张量分解方式

比如我们看到的啥BTD分解，贴张图在这里，[原文](#)在此

4) Block Term Decomposition

块项分解(Block Term Decomposition, BTD)在中作为一种更强大的张量分解被引入，它结合了CP分解和Tucker分解。因此，**BTD比原始的CP和Tucker分解具有更强的鲁棒性**。CP近似于一个张量的秩一张量的和，而BTD是一个低秩Tucker格式的张量的和。或者，**通过将每个模态中的因子矩阵串联起来，将每个子张量的所有核心张量排列成一个块对角核心张量，BTD可以视为Tucker的一个实例**。因此，考虑一个n阶张量$\mathcal{X} \in R^{I_1 \times I_2 \times \cdots \times I_d}$，其BTD可以表示为:

$$\mathcal{X} = \sum_{n=1}^{N} \mathcal{G}_n \times_1 \mathbf{A}_n^{(1)} \times_2 \mathbf{A}_n^{(2)} \times_3 \cdots \times_d \mathbf{A}_n^{(d)}$$

上式中，N表示CP秩，即块项个数，$\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_d}$是第N个多线性秩块项的核心张量，其秩为$(R_1, R_2, \cdots, R_d)$。

> 当BTD应用于压缩FC层时，生成的紧凑层称为块项层(Block Term layer, BTL)。在BTL中，输入张量$\mathcal{X} \in R^{I_1 \times I_2 \times \cdots \times I_d}$从原始输入向量$X \in R^I$张量化，原始权矩阵W被重塑为$W \in R^{O_1 \times I_1 \times O_2 \times I_2 \cdots \times O_d \times I_d}$
> 然后，我们可以通过BTD用因子张量$\{A_n^{(d)} \in R^{Od \times Id \times Rd}\}_{n=1}^{d}$分解W。通过在$BTD(\mathcal{W})$和$\mathcal{X}$之间进行张量收缩算子，得到输出张量$\mathcal{Y} \in R^{O_1 \times O_2 \times \cdots \times O_d}$，可以将其矢量化作为最终的输出向量。
> 对于Conv层，文献声称通过将4D核重构为矩阵$W \in R^{S \times CHW}$，可以将该层转化为BTL。具体来说，矩阵应进一步重构为$1 \times H \times 1 \times W \times S_1 \times C_1 \times S_2 \times C_2 \times \cdots \times S_d \times C_d$。

通过CP/VM分解，紧凑地编码了体素网格中的空间变化的特征

体积密度和视角相关的颜色值可以从特征中解码出来

最常见的两种张量分解方式

1. Tucker decomposition
2. CP decomposition

这两种分解方法可以看成是张量奇异值分解的推广

CP分解可以认为是一种特殊的Tucker分解

Tucker在这里可以看到比较详细的介绍

在理解CP分解之前需要知道两个向量的知识

1. 向量外积

2. Rank-one 秩一张量

向量外积

$$c = \mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^\top = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} = \begin{bmatrix} a_1b_1 & a_1b_2 & \ldots & a_1b_n \\ a_2b_1 & a_2b_2 & \ldots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_mb_1 & a_mb_2 & \ldots & a_mb_n \end{bmatrix}$$

在这里的a b 两个向量的长度并无要求

同理，在后面的3D张量的分解中 a b c 三个向量的长度也没有要求（其实就是跟3D张量的各个维度的长度一样）

> 注意：向量的外积和向量的叉乘并不是一个意思

Rank-one tensor （秩一张量）

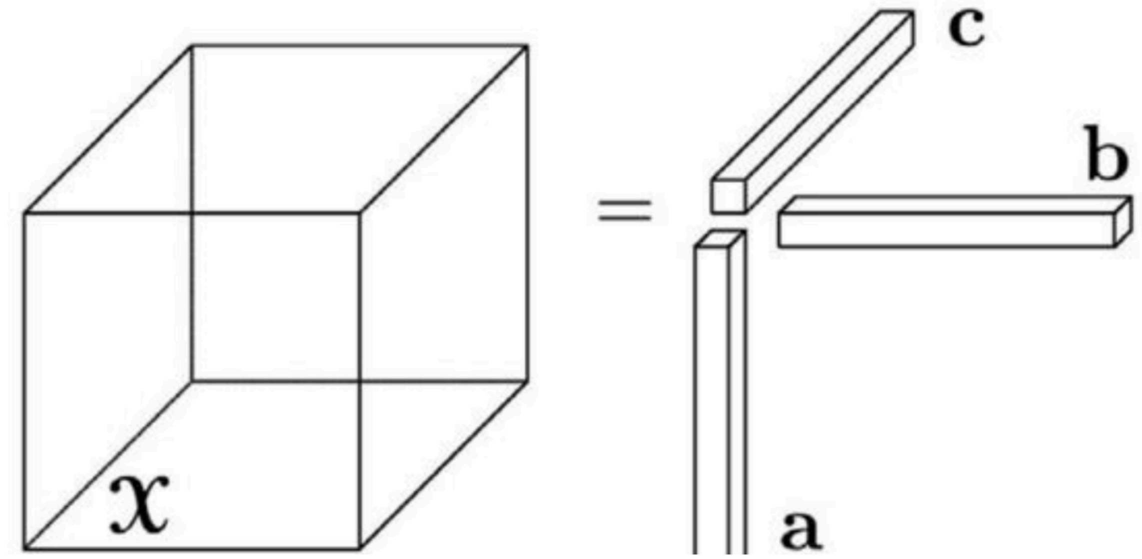
如果一个张量可以写成N个向量的外积，这个张量就是秩一张量

$\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$

$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)}$

同时，张量中的每一个元素为：

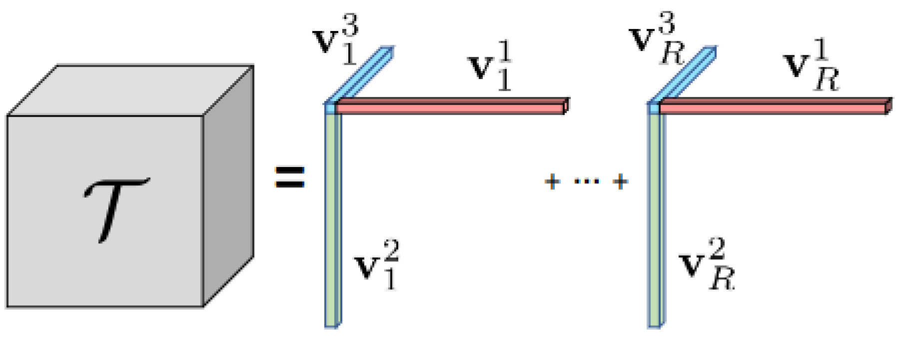
$x_{i_1 i_2 \cdots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \cdots a_{i_N}^{(N)} \quad \text{for all } 1 \leq i_n \leq I_n$



CP分解

cp分解是将其变成一些向量外积的和

更细致的说，cp分解是将其分解为秩一张量之和



多个秩一张量的和，表达式：

$\boldsymbol{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$

这里补充一个内容，张量的秩的概念，张量的秩表示的意义跟矩阵的秩差不多

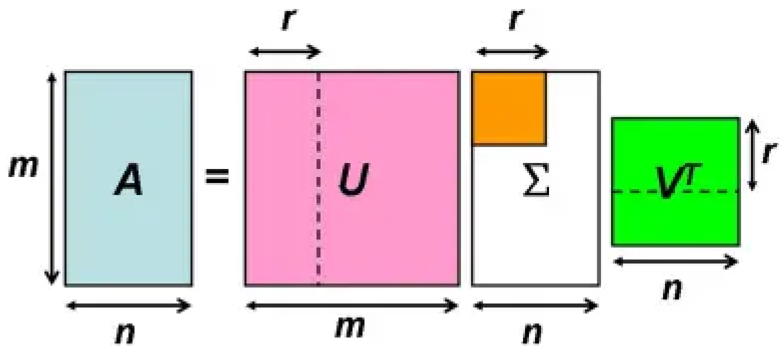
矩阵的秩是其行秩和列秩中的最小值

张量的值就是将其分解成秩一张量和的那个最小的 张量的数量，就是上面公式中 R的数量的最小值

CP分解 与 SVD

CP分解可以看成是SVD分解在张量上的推广

从张量分解成向量的外积，这里反向思考SVD分解



D

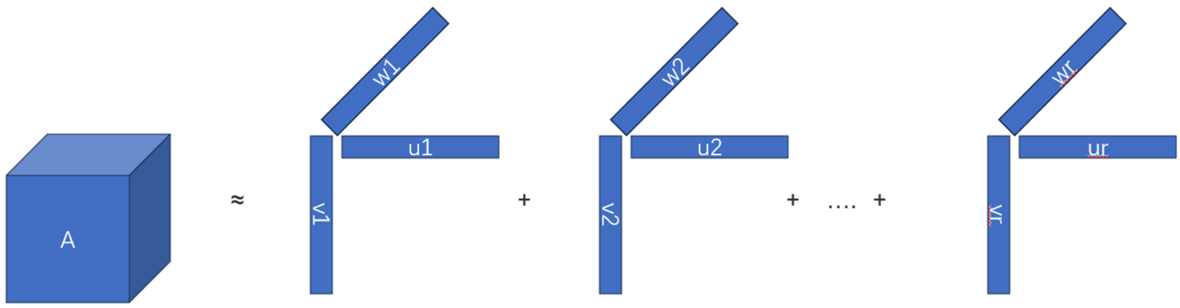
假设A矩阵可以分解成多个矩阵的加和

其中每一个矩阵的构成方式如下：

这里以第一个为例，第一个子矩阵就是U矩阵中的第一个向量和V矩阵中的第一个向量的外积，这两个向量的外积得到的矩阵，其大小就是与A矩阵相同的，以此类推还有第二个，第三个矩阵。

Sigma矩阵中的奇异值（主对角线上的值），就认为是这些子矩阵的加权系数

那么以这种视角看待SVD分解，SVD分解和CP分解的形式基本上统一的



前面不加系数，也就是说 (w, u, v) 为单位向量，要求分解的值的话，就直接用数值分析方法去做逼近，把 (r) 的值固定，解一个优化问题：

$$\arg\min_{u,v,w} \left\| A - \sum_{i=1}^{r} u_1 \circ v_1 \circ w_1 \right\|_2$$

以上就是论文里提到的张量分解的传统分解方式 CP 分解啦。

VM分解

在论文中，主要介绍的内容是VM分解方式，VM分解方式与CP分解相似

VM的含义是 Vector-Matric 向量和矩阵的意思

$$\mathcal{T} = \sum_{r=1}^{R_1} \mathbf{v}_r^1 \circ \mathbf{M}_r^{2,3} + \sum_{r=1}^{R_2} \mathbf{v}_r^2 \circ \mathbf{M}_r^{1,3} + \sum_{r=1}^{R_3} \mathbf{v}_r^3 \circ \mathbf{M}_r^{1,2}$$

上面是论文中的VM分解公式

- $\mathcal{T}$：是一个三维张量（体积数据）。
- v_r^i：是方向 (i) 上的向量，向量通常用于捕捉一维方向的特征信息。例如，向量 (v_r^1) 可以表示 x 轴方向上的向量特征。
- $M_r^{i,j}$：是矩阵，通常用于捕捉二维平面上的信息。例如，矩阵 (M_r^{2,3}) 可以表示在 y 和 z 平面上的特征信息。
- R_1, R_2, R_3：分别代表每个分解项的秩，秩表示了分解中包含的成分数，这三个秩可以不同，从而提供了更多的自由度和灵活性来优化分解的结果。
- "∘" 是外积（outer product），用于构建张量的每个分量。

但是在后续的内容中，认为R_1, R_2, R_3这三个值可以设置成相同的值，因为这三项就是代表了x, y, z三个方向上向量和空间的关系，一个简单的想法便是，它们在空间中的贡献是相同的

第一项中的v可以假设其是x轴方向的向量，那么M便是yz平面的矩阵，以此类推后面两项。

个人理解

至于为什么用VM而不是CP分解，作者也进行了一些解释。我个人理解CP分解虽然非常的紧凑，但是由于低秩表示能包含的信息实在太少了，并且它因为是一小块一小块的，所以组件（数量）太多了，计算的时候复杂度就会太高；而VM有种空间换时间的意思，把分解不做的那么彻底，这样就可以在单个组件中包含更多信息，训练速度更快，但是也仍然比原先的$O(n^3)$低一个数量级的空间储存。

辐射场

以上是张量分解的定义

完成了张量分解之后，下一步便是将张量中的具体的值与场景中的体积密度，表面特征建立联系

$\sigma, c = \mathcal{G}_\sigma(\mathbf{x}), S\left(\mathcal{G}_c(\mathbf{x}), d\right)$

上式是论文中给出的定义公式

体积密度的详细的公式如下：

$\mathcal{G}_\sigma = \sum_{r=1}^{R_\sigma} \mathbf{v}_{\sigma,r}^X \circ \mathbf{M}_{\sigma,r}^{YZ} + \mathbf{v}_{\sigma,r}^Y \circ \mathbf{M}_{\sigma,r}^{XZ} + \mathbf{v}_{\sigma,r}^Z \circ \mathbf{M}_{\sigma,r}^{XY} = \sum_{r=1}^{R_\sigma} \sum_{m \in XYZ} \mathcal{A}_{\sigma,r}^m$

sigma的定义基本就是VM的公式，体积密度其本身就是一个3D的张量

表面特征的详细公式如下：

$$\begin{aligned}
\mathcal{G}_c &= \sum_{r=1}^{R_c} \mathbf{v}_{c,r}^X \circ \mathbf{M}_{c,r}^{YZ} \circ \mathbf{b}_{3r-2} + \mathbf{v}_{c,r}^Y \circ \mathbf{M}_{c,r}^{XZ} \circ \mathbf{b}_{3r-1} + \mathbf{v}_{c,r}^Z \circ \mathbf{M}_{c,r}^{XY} \circ \mathbf{b}_{3r} \\
&= \sum_{r=1}^{R_c} \mathcal{A}_{c,r}^X \circ \mathbf{b}_{3r-2} + \mathcal{A}_{c,r}^Y \circ \mathbf{b}_{3r-1} + \mathcal{A}_{c,r}^Z \circ \mathbf{b}_{3r}
\end{aligned}$$

表面特征是一个4D的张量，在空间XYZ的三维基础上，多了一个特征维度（这个值在代码中式27）

上面中的b向量代表的就是从XYZ空间上的值向第四维度的转换 （v M b，这三个内容合起来是4D张量），但和XYZ维度相比，**这一维度通常是低维的，秩往往也很低**。因此，这一维度不参与tensor分解。

特征向量b通过为每个分解项提供额外的维度，可以提升模型对复杂表面特征的表达能力。由于表面特征往往比体积密度更复杂，需要更多的自由度来捕捉表面的纹理、光照变化等细节，特征向量的引入为这些高维特征提供了更多的参数，帮助模型对复杂场景的精确表示。

注意，我们有$3R_c$个向量$\mathbf{b}_r$，以对应总的因子数。

总体来说，我们将整个辐射场分解为$3R_\sigma + 3R_c$个矩阵$M_{\sigma,r}^{YZ}, \ldots, M_{c,r}^{YZ}, \ldots$以及$3R_\sigma + 6R_c$个向量$\mathbf{v}_{\sigma,r}^{X}, \ldots, \mathbf{v}_{c,r}^{X}, \ldots, \mathbf{b}_r$。这些矩阵和向量（$\mathbf{b}_r$除外），可以看作描述了**场景几何**和外观沿其相应轴的空间分布。

将所有的$\mathbf{b}_r$stack 到一起，我们可以得到一个$P \times 3R_c$的矩阵$\mathbf{B}$，矩阵$\mathbf{B}$可以被看作一个**全局外观字典**（global appearance dictionary），抽象出整个场景的外观共性。

如何使用颜色特征与密度特征等到某一角度的图像?

要求某一像素点的颜色，就从像素点中心发出一根射线，对射线上的点进行采样。

采样点密度的计算

使用三线性插值进行计算就行。三线性插值是昂贵的，但由于线性插值和外积运算都是线性的，对一个 component tensor（即 $\mathcal{A}$）做线性插值，等价于对其 vector/matrix 因子做线性/双线性插值。

采样点颜色的计算

三线性插值得到某一点的颜色特征值（27维），把采样点的 xyz 坐标进行编码（3 + viewpe6 + 3 + pospe * 3）放到神经网络中，得到这一点的 RGB(3) 的输出。

用于这些采样点的颜色值与密度值，就可以使用体渲染公式，得到某一像素点的颜色了。

$$\alpha_n = 1 - e^{-\sigma_n \delta_n}$$

$$\hat{C} = C_0\alpha_0 + C_1\alpha_1(1-\alpha_0) + C_2\alpha_2(1-\alpha_0)(1-\alpha_1) + \cdots + C_n\alpha_n(1-\alpha_0)(1-\alpha_1)\ldots(1-\alpha_{n-1})$$

如何利用现有图像对数据进行训练?

对每个 `grid dense` 的σ_i和C_i进行随机初始化，同时对 MLP 神经网络进行训练。

Loss Function

$$\mathcal{L} = \|C - \hat{C}\|_2^2 + \omega \cdot \mathcal{L}_{reg}$$

- L1 loss

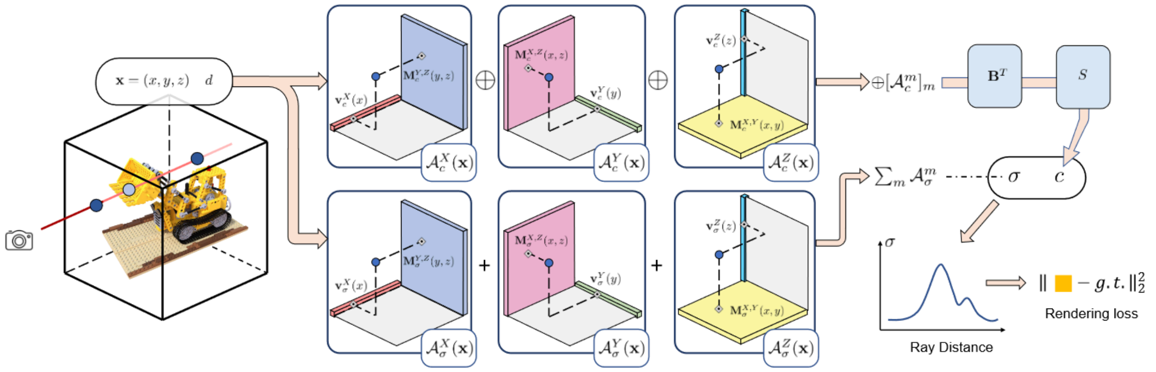
$$\mathcal{L}_{L1} = \frac{1}{N}\sum_{r=1}^{R_\sigma}(\|\mathbf{M}_{\sigma,r}\| + \|\mathbf{V}_{\sigma,r}\|)$$

- TV loss

$$\mathcal{L}_{TV} = \frac{1}{N}\sum\left(\sqrt{\Delta^2 A_{\sigma,r}^m} + 0.1 \cdot \sqrt{\Delta^2 A_{C,r}^m}\right)$$

约束 plane 的变化缓慢一些，梯度不要太大。

模型



这里与NeRF不同的就是如何计算Sigma和RGB的中间那个区域，其他的部分，如体渲染，最后Loss的计算都是相同的