ShapeConv：面向室内 RGB-D 语义分割的形状感知卷积层（2021）

2022.11.22

1 技术核心

RGB-D 语义分割中，现有的方法大多采用 homogeneous convolution（所有的卷积核大小一致的标准卷积）来消耗 RGB 和深度特征，忽略了它们的内在差异。

RGB 值捕获了投影图像空间中的光影外观属性，而深度特征编码了局部几何形状以及在更大背景下的位置，与位置相比，形状可能更具有内在性，与语义的联系更强，因此对分割的准确性更关键。

Figure 1. Visual demonstration of why the *shape* of an RGB-D image matters. Regarding the images on the top, lines with the same color share a same *shape*, yet with different *base*. The corresponding patches are shown on the bottom.

图 1:

上图解释了不同 patch 的 shape 和 base 的不同，相同的颜色线条代表的部分属于同一个 shape，但是属于不同的 base，**base** 用来描述物体**在哪里**，而 **shape** 则描述物体**是什么**。

作者希望相同椅子的相同 patch 具有相同的特征，并且在学习过程中达到形状不变性（shape invariance）。当在相同 patch 中使用基础卷积算子时（vanilla convolution operator），由于其 base 的差异，所得的特征会不同，从而阻碍了实现形状不变性。

因此，作者引入了 Shape-aware Convolutional layer(ShapeConv) 来处理深度特征，其中深度特征首先被分解为一个 **shape 分量**和一个 **base 分量**，然后引入了两个可学习权重来独立地配合他们，最后对这两个分量的加权组合进行卷积。

ShapeConv 是与模型无关的，可以集成到大多数 CNN 中，以取代用于语义分割的普通卷积层。

2 实施方法

2.1 ShapeConv for RGB-D Data

2.1.1 直觉法

给定输入 patch $\mathbb{P} \in R^{K_h \times K_w \times C_{in}}$，$K_h$ 和 K_w 是通道的空间维数，C_{in} 代表输入特征图的通道数，普通卷积的输出特征通过以下方法得到：

$$\mathbb{F} = Conv(\mathbb{K}, \mathbb{P}) \tag{1}$$

如图 2（a）所示，其中，$\mathbb{K} \in R^{K_h \times K_w \times C_{in} \times C_{out}}$ 代表卷积层中通道的可学习参数；C_{out} 代表输出特征图的通道数。$\mathbb{F} \in R_{C_{out}}$ 的每个元素可由下式计算：

$$\mathbb{F}_{C_{out}} = \sum_{i}^{K_h \times K_w \times C_{in}} (\mathbb{K}_{i,C_{out}} \times \mathbb{P}_i)$$

由上式可以很容易看出 $\mathbb{F}$ 随 $\mathbb{P}$ 的不同值而变化，由普通卷积层学习到的相关的输出特征为 $\mathbb{F}_{\nVdash} = Conv(\mathbb{K}, \mathbb{P}_{\nVdash})$，$\mathbb{F}_{\nvDash} = Conv(\mathbb{K}, \mathbb{P}_{\nvDash})$，由于 $\mathbb{P}_{\nVdash}$ 和 $\mathbb{P}_{\nvDash}$ 不同（距观测点的位置不同，即 base 不同），因此他们的特征不同，可能会导致不同的预测结果。

但是 $\mathbb{P}_{\nVdash}$ 和 $\mathbb{P}_{\nvDash}$，在图 1 的 patch 中（红色区域），属于同一个种类——椅子。普通卷积不能很好地处理这种情况。鉴于此，作者提出通过对 RGB-D 语义分割中 shape 的有效建模来填补这一空白。

2.1.2 ShapeConv Formulation

基于上述分析，本文提出将输入 patch 分解为两个分量：表示 patch 所在位置的 base 分量 $\mathbb{P}_B$ 和表示 patch 所在位置的 shape 分量 $\mathbb{P}_S$。因此，我们取 patch 得均值为 $\mathbb{P}_B$，其相对值为 $\mathbb{P}_S$：

$$\mathbb{P}_B = m(\mathbb{P})$$

$$\mathbb{P}_S = \mathbb{P} - m(\mathbb{P})$$

这里 $m(\mathbb{P})$ 是 $\mathbb{P}$ 上的均值函数 $\mathbb{P}_B \in R^{1 \times 1 \times C_{in}}$，$\mathbb{P}_S \in R^{K_h \times K_w \times C_{in}}$。ShapeConv 引入了两个可学习权重 $\mathbb{W}_B \in R^1$ 以及 $\mathbb{W}_S \in R^{K_h \times K_w \times K_h \times K_w \times C_{in}}$，来分别对应 base 和 shape 分量，并将输出的特征采用逐元素相加的方法进行组合，形成与原始 $\mathbb{P}$ 大小相同的 shape-aware patch。ShapeConv 的公式由下式给出：

$$\begin{aligned} \mathbb{F} &= ShapeConv(\mathbb{K}, \mathbb{W}_B, \mathbb{W}_S, \mathbb{P}) \\ &= Conv(\mathbb{K}, \mathbb{W}_B \Diamond \mathbb{P}_B + \mathbb{W}_S * \mathbb{P}_S) \\ &= Conv(\mathbb{K}, \mathbf{P_B} + \mathbf{P_S}) \\ &= Conv(\mathbb{K}, \mathbf{P_{BS}}) \end{aligned} \tag{2}$$

其中，$\Diamond$ 以及 * 分别代表 base-product 和 shape-product 操作，他们各自的定义为下式：

$$\begin{cases} \mathbf{P_B} = \mathbb{W}_B \Diamond \mathbb{P}_B \\ \mathbf{P}_{B_{1,1,c_{in}}} = \mathbb{W}_B \times \mathbb{P}_{B_{1,1,c_{in}}} \end{cases} \tag{3}$$

$$\begin{cases} \mathbf{P_S} = \mathbb{W}_S * \mathbb{P}_S \\ \mathbf{P}_{S_{k_h,k_w,c_{in}}} = \sum_i^{K_h \times K_w} (\mathbb{W}_{S_{i,k_h,k_w,c_{in}}} \times \mathbb{P}_{S_{i,c_{in}}}) \end{cases} \tag{4}$$

其中，c_{in}，k_h，k_w，分别代表 C_{in}，K_h，K_w 维度中元素的索引。

作者通过将 $\mathbf{P_B}$ 与 $\mathbf{P_S}$ 相加的方式构造 shape-aware patch $\mathbf{P_{BS}} \in R^{K_h \times K_w \times C_{in}}$

2.2 ShapeConv in Training and Inference

2.2.1 训练阶段

ShapeConv 可以有效地利用 patch 中的 shape 信息，但是由于公式（3）（4）中的两个乘积运算，用 ShapeConv 代替 cnn 中的普通卷积层会带来更多的计算成本。为了解决这个问题，作者将这两个操作从 patch 转移到卷积核中。

$$\begin{cases} \mathbf{K_B} = \mathbb{W}_B \Diamond \mathbb{K}_B \\ \mathbf{K}_{B_{1,1,c_{in},c_{out}}} = \mathbb{W}_B \times \mathbb{K}_{B_{1,1,c_{in},c_{out}}} \end{cases}$$

$$\begin{cases} \mathbf{K_S} = \mathbb{W}_S * \mathbb{K}_S \\ \mathbf{K}_{S_{k_h,k_w,c_{in},c_{out}}} = \sum_i^{K_h \times K_w} (\mathbb{W}_{S_{i,k_h,k_w,c_{in}}} \times \mathbb{K}_{S_{i,c_{in},c_{out}}}) \end{cases}$$

这里的 $\mathbb{K}_B \in R^{1 \times 1 \times C_{in} \times C_{out}}$，$\mathbb{K}_S \in R^{K_h \times K_w \times C_{in} \times C_{in}}$，它们分别代表卷积核中的 base-component 和 shape-component，$\mathbb{K} = \mathbb{K}_B + \mathbb{K}_S$。

因此可以重新构造 ShapeConv：

$$\begin{aligned} \mathbb{F} &= ShapeConv(\mathbb{K}, \mathbb{W}_B, \mathbb{W}_S, \mathbb{P}) \\ &= Conv(\mathbb{W}_B \Diamond m(\mathbb{K}) + \mathbb{W}_S * (\mathbb{K} - m(\mathbb{K})), \mathbb{P}) \\ &= Conv(\mathbb{W}_B \Diamond \mathbb{K}_B + \mathbb{W}_S * \mathbb{K}_S, \mathbb{P}) \\ &= Conv(\mathbf{K_B} + \mathbf{K_S}, \mathbb{P}) \\ &= Conv(\mathbf{K_{BS}}, \mathbb{P}) \end{aligned} \tag{5}$$

其中，$m(\mathbb{K})$ 是 $\mathbb{K}$（除以 $K_h \times K_w$ 维数）上的均值函数，$\mathbf{K_{BS}} = \mathbf{K_B} + \mathbf{K_S}$，$\mathbf{K_{BS}} \in R^{K_h \times K_w \times C_{in} \times C_{out}}$。

事实上，公式（2）和公式（5）在数学上等价，即：

$$\begin{aligned} \mathbb{F} &= ShapeConv(\mathbb{K}, \mathbb{W}_B, \mathbb{W}_S, \mathbb{P}) \\ &= Conv(\mathbb{K}, \mathbf{P_{BS}}) \\ &= Conv(\mathbf{K_{BS}}, \mathbb{P}) \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbb{F}_{c_{out}} &= \sum_i^{K_h \times K_w \times C_{in}} (\mathbb{K}_{i,c_{out}} \times \mathbf{P_{BS}}_i) \\ &= \sum_i^{K_h \times K_w \times C_{in}} (\mathbf{K_{BS}}_{i,c_{out}} \times \mathbb{P}_i) \end{aligned} \tag{7}$$

在图 2（b）和（c）所示的实现中利用等式 5 中的形状转换。

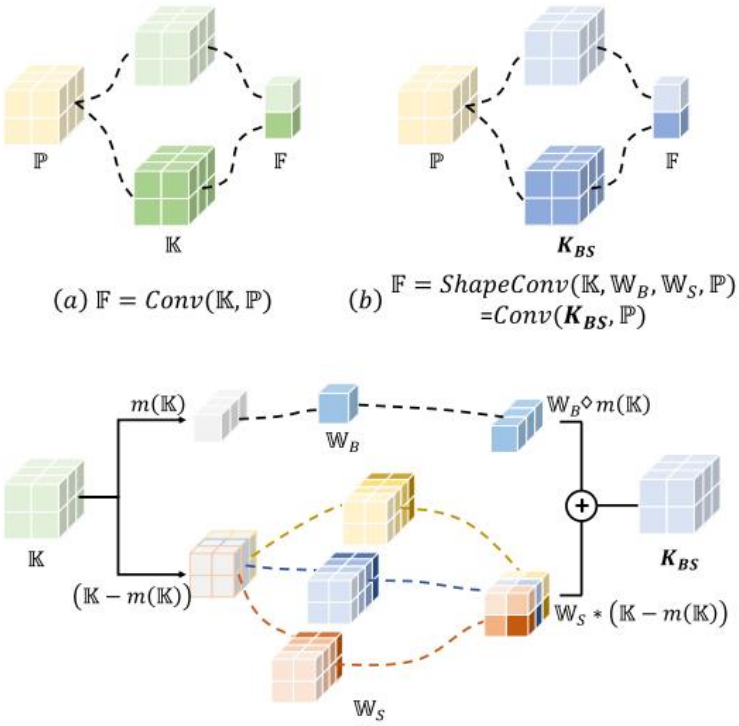

Figure 2. Comparison of vanilla convolution and ShapeConv within a patch $\mathbb{P}$. In this figure, $K_h = K_w = 2$, $C_{in} = 3$, and $C_{out} = 2$, "+" denotes element-wise addition. (a) Vanilla convolution with kernel $\mathbb{K}$; (b) ShapeConv with folding the $\mathbb{W}_B$ and $\mathbb{W}_S$ into $\mathbf{K_{BS}}$; (c) The computation of $\mathbf{K_{BS}}$ from $\mathbb{K}$, $\mathbb{W}_B$ and $\mathbb{W}_S$.

图 2:

2.2.2 推理阶段

由于两个附加权重 $\mathbb{W}_B$ 和 $\mathbb{W}_S$ 固定，因此可以将它融合为 $\mathbf{K_{BS}}$:

$$\mathbf{K_{BS}} = \mathbb{W}_B \Diamond \mathbb{K}_B + \mathbb{W}_S * \mathbb{K}_S \tag{8}$$

$\mathbf{K_{BS}}$ 与等式 1 中的 $\mathbb{K}$ 具有相同的张量大小，因此，ShapeConv 实际上与图 2（a）中的普通卷积层相同。

2.3 ShapeConv 增强型网络架构

为了在语义分割中利用高级模型架构，首先需要通过 RGB 和 D 信息的级联将输入特征从 RGB 图像转换为 RGB-D 数据。在实际中，D 可以是深度值或 HHA 图像 (HHA 即将深度图像转换为三种不同的通道 (水平差异，对地高度以及表面法向量的角度))，然后在，模型主干和分割阶段都使用 ShapeConv 替换普通卷积。

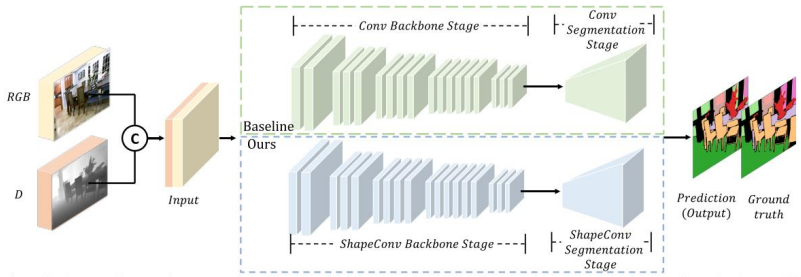

Figure 3. The overall semantic segmentation network architecture. In this figure, yellow and orange cube denote the RGB and D inputs; "C" denotes channel-wise concatenation; Green and blue boxes denote architectures consisting of vanilla convolutional layers and ShapeConv layers, respectively.

图 3:

Architecture	Back bone	Setting	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
Deeplabv3+ [4]	Res Net 101	Baseline	73.4	58.9	45.9	59.7
		Ours	74.5	59.5	47.4	60.8
		+	1.1	0.6	1.5	1.1
	Res Net 50	Baseline	73.1	57.7	45.6	59.2
		Ours	74.1	59.1	47.3	60.5
		+	1.0	1.4	1.7	1.3
Deeplabv3 [3]	Res Net 101	Baseline	73.3	57.3	45.1	59.2
		Ours	73.6	58.5	46.4	59.7
		+	0.3	1.2	1.3	0.5
	Res Net 50	Baseline	71.6	55.5	43.2	57.2
		Ours	72.8	56.6	44.9	58.5
		+	1.2	1.1	1.7	1.3
UNet [23]	Res Net 101	Baseline	70.9	54.7	42.1	57.7
		Ours	72.3	56.5	43.9	58.8
		+	1.4	1.8	1.8	1.1
	Res Net 50	Baseline	70.0	51.7	39.7	55.5
		Ours	70.8	54.1	42.0	56.9
		+	0.8	2.4	2.3	1.4
PSPNet [33]	Res Net 101	Baseline	72.8	56.8	44.2	58.9
		Ours	73.3	59.2	46.3	59.6
		+	0.5	2.4	2.1	0.7
	Res Net 50	Baseline	71.1	53.6	42.0	56.7
		Ours	72.0	56.2	44.0	57.7
		+	0.9	2.6	2.0	1.0
FPN [18]	Res Net 101	Baseline	72.8	57.3	44.7	59.1
		Ours	73.6	58.4	45.9	60.0
		+	0.8	1.1	1.2	0.9
	Res Net 50	Baseline	70.3	52.8	40.9	56.0
		Ours	71.5	54.9	42.8	57.5
		+	1.2	2.1	1.9	1.5

图 4: