基于 Transformer 的超像素变换的高效语义分割

摘要

语义分割旨在对图像中的每个像素进行分类，是机器感知中的关键任务，在机器人技术和自动驾驶等领域有许多应用。由于这项任务的高维度，大多数现有方法使用本地操作，如卷积，生成每个像素的特征。然而，由于在密集图像上操作的计算成本高，这些方法通常无法有效利用全局上下文信息。在这项工作中，我们通过利用超像素的概念（图像的过分割）并结合现代转换器框架，提出了解决这个问题的方法。具体而言，我们的模型通过一系列局部交叉注意力学习将像素空间分解为具有空间低维度的超像素空间。然后，我们对超像素应用多头自注意力，以全局上下文丰富超像素特征，然后直接为每个超像素生成类别预测。最后，我们通过超像素与图像像素特征之间的关联，将超像素类别预测直接投影回像素空间。在超像素空间进行推理使得我们的方法在与基于卷积的解码器方法相比具有更高的计算效率。然而，由于全局自注意机制生成的丰富超像素特征，我们的方法在语义分割方面达到了最先进的性能。我们在 Cityscapes 和 ADE20K 上的实验证明了我们的方法在准确性方面与当前技术水平相匹配，同时在模型参数和延迟方面表现更出色。

关键词：超像素；语义分割；Transformer

1 引言

语义分割问题，即对图像中的每个像素进行分类，在许多机器人应用中越来越普遍。在杂乱的环境中进行导航，尤其是对于像自动驾驶这样深度依赖场景理解的应用，对世界进行密集、细粒度的理解是必要的，而这在安全关键的情况下显得尤为重要。另一方面，许多机器人系统是高度复杂和专业化系统的组合，而延迟对于实时操作而言则是一个时刻存在的问题。

对于现代机器人系统而言，安全性、性能和延迟之间的平衡至关重要。虽然在语义分割领域的最新技术在诸如平均交并比（mIoU）等指标方面能够取得强大的性能，但许多方法仍然依赖于产生场景中每个像素预测的密集解码器。因此，这些方法往往相对昂贵，并且可以说对于通常非常相似的附近像素产生了大量冗余计算。

为了解决这个问题，我们的目标是将围绕超像素的经典思想引入到现代深度学习中。使用超像素的前提是将图像分解并过分割成一系列不规则的补丁。通过将相似的像素分组成超像素，然后在超像素级别进行操作，可以显著降低密集预测任务（如语义分割）的计算成本。然而，经典的超像素算法，如 SLIC [1]，依赖于每个图像像素与超像素之间的硬关联。这使得将超像素表示嵌入到神经网络架构 [24] 中变得困难，因为这种关联在反向传播过程中不可微分。最近的研究，如超像素采样网络（SSN）[20]，通过将硬关联转变为软关联来解决了这个问题。虽然这是将超像素纳入神经网络的一步，但它们的分割质量仍然落后于采用每像素或每掩模表示的其他模型。

在这项工作中，我们提出了一种旨在在现代转换器框架 [34] 中恢复可微分超像素生成流水线的新型架构。我们提出通过在一组学习到的超像素查询和像素特征之间开发一系列局部交叉注意力，来学习超像素表示，而不是使用 SLIC [1] 和 SSN [20] 中使用的迭代聚类算法。交叉注意力模块的输出充当超像素特征，直接用于语义分割预测。因此，提出的转换器解码器有效地将对象查询转换为超像素特征，使模型能够端到端地学习超像素表示。

操作在超像素级别上提供了许多显著的优势。传统的基于像素的方法受到像素空间高维度的限制，使得全局自注意在计算上变得棘手。许多方法，如轴向注意力 [36] 或窗口注意力 [26]，已经发展出来以解决这些问题，通过将全局关注放宽到局部关注。通过将图像过分割成一小组超像素，我们能够在超像素上高效应用全局自注意，为超像素特征提供完整的全局上下文，即使在处理高分辨率图像时也是如此。尽管在我们的模型中应用了全局自注意（相对于传统的卷积神经网络），但由于超像素空间的低维度，我们的方法比现有方法更有效。最后，我们直接为超像素特征生成语义类别，然后使用超像素-像素关联将预测的类别反投影到图像空间。

我们在 Cityscapes [12] 和 ADE20K [49] 数据集上进行了广泛的评估，在这里我们的方法达到了最先进的性能，但计算成本显著较低。

总的来说，本工作的主要贡献如下：

- 在现代转换器框架中复兴超像素表示的第一个工作，其中使用对象查询来学习超像素特征。
- 一种新颖的网络架构，利用局部交叉注意力将像素特征的空间维度显著降低到一小组超像素特征，实现了它们之间的全局上下文学习和每个超像素的直接分类。
- 一个超像素关联和展开方案，将每个超像素类别预测投影回到密集像素分割，丢弃卷积神经网络像素解码器。
- 在 Cityscapes 和 ADE20k 数据集上的实验证明，我们的方法优于当前的技术水平。

2 相关工作

超像素用于分割在深度学习时代之前，超像素表示与图形模型结合是图像分割的主要范式。超像素方法 [28]， [31]， [11]， [1] 通常在预处理步骤中用于降低计算成本。浅层分类器，如 SVM [13]，预测每个超像素的语义标签 [15]，并汇总手工制作的特征。然后使用图形模型，特别是条件随机场 [23]，来细化分割结果 [19]， [22]， [21]。

卷积神经网络用于分割以像素为单位进行语义分割的卷积神经网络（ConvNets） [24] 以全卷积的方式部署 [29]。典型的基于 ConvNet 的方法包括 DeepLab 系列 [4]， [5]， [7]，PSPNet [47]，UPerNet [39] 和 OCRNet [44]。此外，还有一些工作 [16]， [20] 使用超像素来聚合由 ConvNets 提取的特征，并取得了令人满意的结果。

变换器用于分割变换器 [34] 及其视觉变体 [14] 已作为图像分割的主干编码器被采用 [6]， [48]， [3]。变换器编码器可以通过使用自注意模块 [37]， [36] 来实例化为增强 ConvNets。当作为独立的主干 [30]， [14]， [26]， [40] 时，它们与先前的 ConvNet 基线相比也表现出强大的性能。变换器还用于图像分割的解码器 [2]。一个常见的设计是通过对象查询生成掩模嵌入向量，然后与像素特征相乘以生成掩模 [33]， [38]。例如，MaX-DeepLab [35] 提出了一种直接预测带有类别标签的对象掩模的端到端掩模变换器框架。Segmenter [32] 和 MaskFormer

[10] 从掩模分类的角度解决语义分割问题。K-Net [45] 通过一组可学习的核生成分割掩模。受到掩模变换器和聚类算法之间相似性的启发 [28]，提出了基于聚类的掩模变换器来进行图像分割 [42]，[41]，[43]。可变形变换器 [50] 也用于改进图像分割，如 Panoptic SegFormer [25] 和 Mask2Former [9]。

与本文类似，Region Proxy [46]（RegProxy）也将超像素的思想融入深度分割网络中，通过使用 CNN 解码器学习每个像素与超像素之间的关联。然而，RegProxy 使用常规像素网格上的特征来表示每个超像素，并在其上应用自注意力。相比之下，我们应用一组学到的权重，这些权重对应于每个超像素，并使用像素特征进行交叉注意力计算像素-超像素关联。我们的实验证明，我们的方法提供了显著的性能改进。

总的来说，将变换器应用于分割的先前工作在某种程度上都依赖于密集的 CNN 解码器生成最终的密集特征，然后将这些特征与注意机制结合以提高性能。相比之下，我们的方法使用交叉注意力将图像减少到一小组超像素，并且在解码器中仅在这个超像素空间中应用自注意力。这使得我们的方法在更低维度的空间上操作（通常是图像分辨率的 32 * 32 小），同时利用全局自注意力的优势，实现了最先进的性能。。

3 本文方法

3.1 本文方法概述

此部分对本文将要复现的工作进行概述：

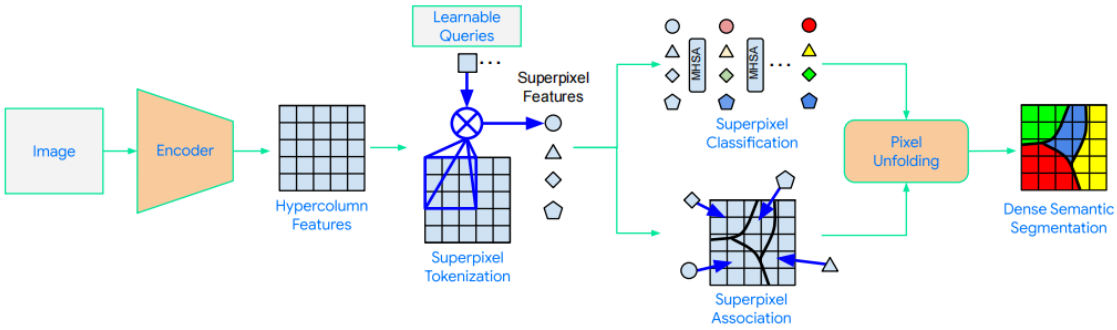

图 1. 方法示意图

我们提出的 Superpixel Transformer 架构总结如图 1 所示，由四个主要组件组成：

1) 像素特征提取：一个卷积编码器骨干网络，用于生成超列特征。

2) 超像素提取：一系列局部双通道交叉注意力，介于一组学习到的查询和像素特征之间，用于生成一组超像素特征。

3) 超像素分类：一系列多头自注意力层，用于优化超像素特征并为每个超像素生成语义类别。

4) 超像素关联：将预测的超像素类别与像素特征关联起来，得到最终的密集语义分割。

我们将在以下各小节详细介绍每个组件。

3.2 特征提取模块

典型的卷积神经网络，如 ResNet [18] 和 ConvNeXt [27]，被用作编码器的骨干网络。在编码器输出的基础上，我们应用一个多层感知机（MLP），并对第 1 阶段（步幅 2），第 3 阶段（步幅 8）和第 5 阶段（步幅 32）之后的特征进行双线性调整。通过加法组合多尺度特征以形成超列特征 [17]。每个像素特征由其相应的超列特征表示，然后输入到下面的超像素提取模块。

$$Q_{pi} = \text{softmax}_{i \in \mathcal{N}(p)}(I_p^{t_n} \cdot S_i^{t_n}) \quad (5)$$

$$Y_p = \sum_{i \in \mathcal{N}(p)} Q_{pi} \cdot C_i \quad (6)$$

3.3 超像素提取

在介绍我们提出的超像素提取模块之前，我们简要回顾了先前关于可微分 SLIC（Differentiable SLIC）的工作 [1]， [20]，并将其与 Transformer 框架相结合进行了现代化改进。

初步了解：可微分 SLIC 简单线性迭代聚类（SLIC） [1] 采用了经典的迭代 k 均值算法 [28]，通过根据像素的特征（例如颜色相似性和位置接近度）对其进行聚类，从而生成超像素。给定一组像素特征 I_p 和 S_i^0 在迭代 0 时初始化的超像素特征，该算法在第 t 次迭代中在两个步骤之间进行迭代：

1):（硬）分配：计算每个像素特征 I_p 和超像素特征 S_i^t 之间的相似度 $Q_{p,i}^t$, 根据最大相似度将每个像素分配给一个超像素。

2): 更新：基于分配给它的像素特征，更新超像素特征 S_i^t。

$$Q_{pi}^t = e^{-\|I_p - S_i^{t-1}\|^2} \quad (1)$$

$$S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^{n} Q_{pi}^t I_p \quad (2)$$

其中 $Z_i^t = \sum_p Q_{pi}^t$ 是归一化常数。在实践中，当 $t = 0$ 时，超像素特征被初始化为图像中均匀分布的一组矩形块内的均值特征。为了减少计算复杂性并应用空间局部性约束，距离计算 $\|I_p - S_i^{t-1}\|^2$ 仅限于每个像素周围的局部 3×3 超像素邻域，尽管更大的窗口尺寸是可能的。

超像素标记化（Superpixel Tokenization）：我们提出展开 SSN 迭代并用一组局部交叉注意力（local cross-attentions）替换 k-均值聚类步骤。我们使用一组随机初始化的可学习查询 S_i^0 初始化超像素特征，这些超像素特征在图像中以规则网格分布（见图 3），然后使用跨超像素特征和像素特征之间的交叉注意力进行超像素更新步骤，这里我们采用了双路径交叉注意力 [35] 的适应形式。

$$S_i^t = S_i^{t-1} + \sum_{p \in \mathcal{N}(i)} \text{softmax}_p(q_{S_i^{t-1}} \cdot k_{I_p^{t-1}}) v_{I_p^{t-1}} \quad (3)$$

$$I_p^t = I_p^{t-1} + \sum_{i \in \mathcal{N}(p)} \text{softmax}_i(q_{I_p^{t-1}} \cdot k_{S_i^{t-1}}) v_{S_i^{t-1}} \quad (4)$$

其中 $N(x)$ 表示 x 的邻域，q、k 和 v 分别是应用于每个相应特征的 MLP 生成的查询、键和值，加上可学习的位置嵌入。对于与超像素对应的每个超像素邻域，我们共享相同的位置嵌入集。对于每个超像素 S_i，有 $9 \cdot h \cdot w$ 个像素邻居，其中 $[h, w]$ 是一个超像素覆盖的补

这种局部双路径交叉注意力有三个目的：

- 与完全的交叉注意力相比，降低复杂性。
- 稳定训练，因为最终的 softmax 只在 9 个超像素特征或 9 · h · w 个像素特征之间进行。
- 促进超像素的空间局部性，迫使它们专注于一致的、局部的过分分割。

3.4 超像素分类

在超像素提取模块中，考虑到更新后的超像素特征，我们通过一系列自注意力机制直接为每个超像素预测一个类别。具体而言，我们应用 k 个多头自注意力（MHSA）层 [34] 来学习超像素之间的全局上下文信息，生成输出 F i 。在超像素特征上执行 MHSA 相比于在像素特征上执行要高效得多，因为超像素的数量要小得多。在我们的实验中，通常使用的超像素分辨率比输入分辨率小 32 * 32 倍。最后，我们应用一个线性层作为分类器，为每个精炼的超像素特征生成语义类别预测，即 C i。与其他方法中使用的 CNN 像素解码器 [8]、[10] 不同，我们的超像素类别预测 C i 可以直接投影回最终的像素级语义分割输出，无需额外的层，如第 III-D 节所述 [8]。

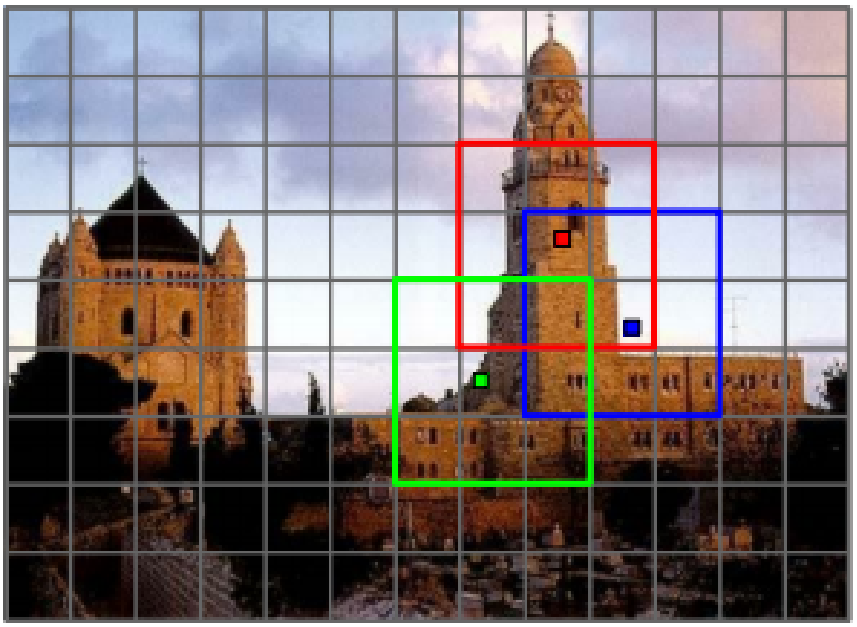

图 2. 超像素关联

3.5 超像素关联

为了将超像素类别预测投影回像素空间，我们使用超像素提取模块的输出，即 $I_p^{t_n}$ 和 $S_i^{t_n}$，计算每个像素与其 9 个相邻超像素之间的关联：

$$Q_{pi} = \text{softmax}_{i \in \mathcal{N}(p)}(I_p^{t_n} \cdot S_i^{t_n}) \quad (5)$$

最终的密集语义分割 Y 然后在每个像素 p 处计算，作为来自超像素分类模块的每个预测超像素类别 C i 的组合，由上述关联加权得到：

$$Y_p = \sum_{i \in \mathcal{N}(p)} Q_{pi} \cdot C_i \quad (6)$$

在训练过程中，密集语义分割 Y 受到语义分割的地面实况（ground truth）的监督。

4 复现细节

4.1 与已有开源代码对比

该论文未公开源码，在复现的过程中，也并没有引用他人的代码，所有代码为本人原创，在无源码复现的过程中，我完成了代码的的编写并且更换了实验的数据集以及修改了部分模型（例如将特征提取的框架改成了 ResNet18，能够更好地适应小型数据集，提高分割的效率，并且依据自身的实验设备减少了网络的深度）。

4.2 实验环境搭建

本报告旨在详细描述实验所需的环境搭建过程，确保实验可以在一个稳定、一致的环境中进行。

- 硬件环境

在进行实验之前，首先需要明确实验所需的硬件环境。以下是我们实验所采用的硬件环境：

处理器：（型号）13th Gen Intel(R) Core(TM) i9-13900K

内存：（容量）24GB

显卡：（型号）NVIDIA GeForce RTX 3090 Ti

存储：（硬盘容量）1T

- 软件环境

实验所需的软件环境包括操作系统、编程语言、库和框架等。以下是我们实验所采用的软件环境：

操作系统：Ubuntu 22.04

编程语言：Python 3.8

深度学习框架：PyTorch 1.8

4.3 项目搭建和代码分析

根据论文指示，代码将分成 4 个模块，第一个模块是像素特征的提取，第二个模块为超像素的形成，第三个模块为超像素的分类结果，第四个模块为超像素的映射。本文将把带代码分割成 3 个模块，其中第三和第四个模块将合并在一起。

- 特征提取模块

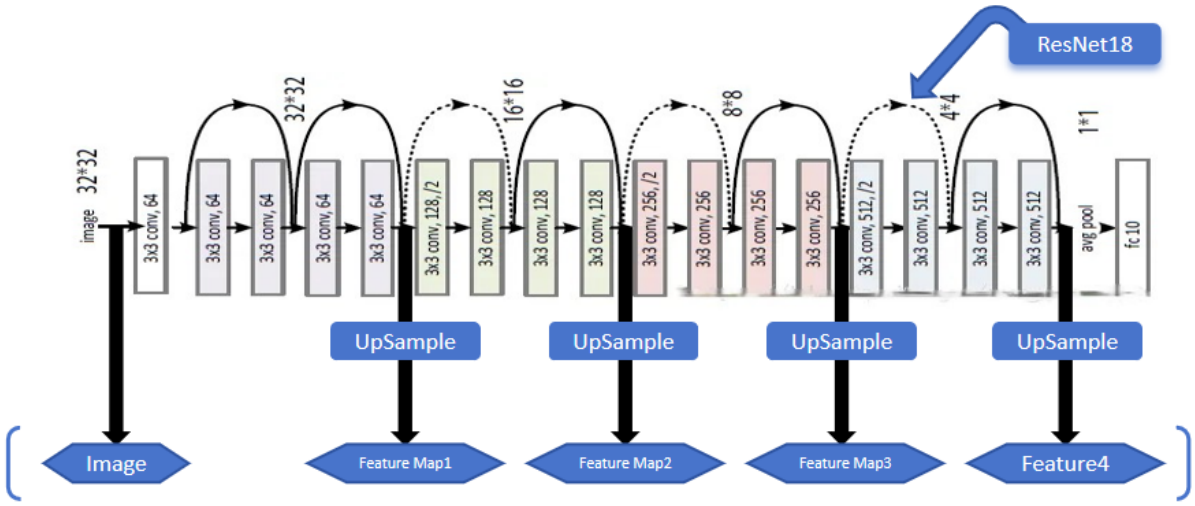

图 3. 解码器示意

```python
# 提取特征
self.feature_map0 = nn.Conv2d(3, target_num_features, kernel_size=1)

self.layer1 = nn.Sequential(*list(self.resnet.children())[:3],
      nn.Conv2d(64,target_num_features,kernel_size=1),
      nn.BatchNorm2d(target_num_features),
      nn.ReLU(),
      nn.ConvTranspose2d(target_num_features,target_num_features, kernel_size=4, stride=2, padding=1))

self.layer2 = nn.Sequential(*list(self.resnet.children())[:6],
      nn.Conv2d(512,target_num_features,kernel_size=1),
      nn.BatchNorm2d(target_num_features),
      nn.ReLU(),
      nn.ConvTranspose2d(target_num_features,target_num_features, kernel_size=16, stride=8, padding=4))

self.layer3 = nn.Sequential(*list(self.resnet.children())[:8],
      nn.Conv2d(2048,target_num_features,kernel_size=1),
      nn.BatchNorm2d(target_num_features),
      nn.ReLU(),
      nn.ConvTranspose2d(target_num_features,target_num_features, kernel_size=64, stride=32, padding=16))
```

图 4. 解码器代码示意

上图展示了解码器的结构及其代码实现。我们复现的解码器采用了 ResNet18 的骨架。当图像经过 ResNet18 残差网络时，能够从不同层中提取图像的表面特征和深度特征，实现多尺度的信息获取，从而有效地进行图像解码。解码器进一步处理特征图，通过上采样将其恢复为图像的原始尺寸，并进行下一步的特征拼接。这个过程有助于保留图像中的关键信息，为后续步骤提供更丰富的特征表示。

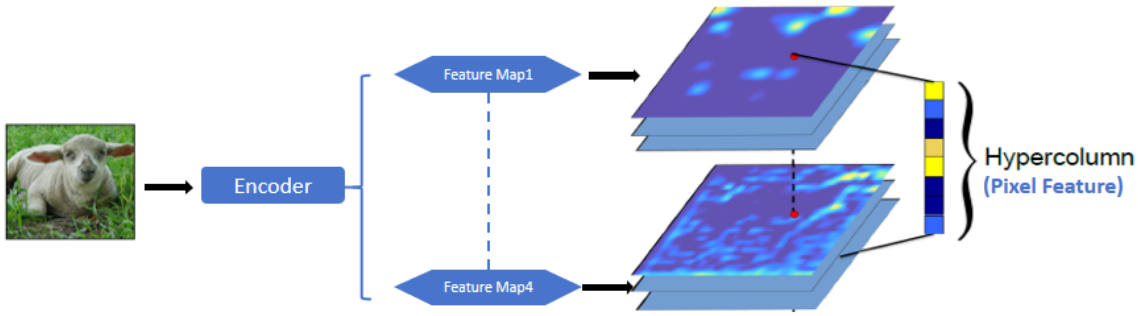

图 5. 特征拼接模块示意

```
feature0 = self.feature_map0(x)
upsampled_feature1 = F.interpolate(self.layer1(x), size=(height, width), mode='bilinear', align_corners=False)
upsampled_feature3 = F.interpolate(self.layer2(x), size=(height, width), mode='bilinear', align_corners=False)
upsampled_feature5 = F.interpolate(self.layer3(x), size=(height, width), mode='bilinear', align_corners=False)
# 将不同尺度的特征连接在一起
hypercolumn = torch.cat([feature0, upsampled_feature1, upsampled_feature3, upsampled_feature5], dim=1)
```

图 6. 特征拼接代码示意

上图展示了特征拼接的流程和代码实现过程。在生成一系列特征图之后，这些特征图被拼接在一起，随后提取每个像素的特征值形成特征向量。这些特征向量进一步组合成超列向量，为我们在后续图像聚类过程中提供更丰富的信息。这个步骤的执行有助于综合利用不同尺度和特征的信息，从而提高对图像内容的全面理解。

- 超像素聚类模块

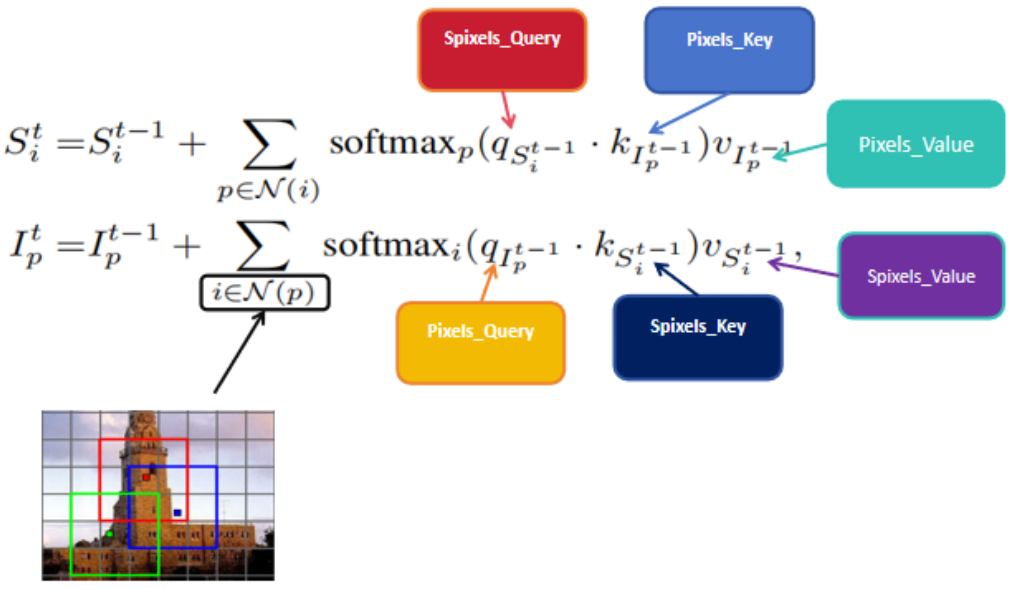

图 7. 超像素聚类示意

```
# 3更新Spixels和Pixels ()
for _ in range(n_iter):
    # 计算得分Score_Superpixel, Score_pixel
    Score_Superpixel = torch.matmul(spixels_Q, pixels_K.permute(0, 2, 1)) / (spixels_Q.size(-1) ** 0.5)
    Score_pixel = torch.matmul(pixels_Q, spixels_K.permute(0, 2, 1)) / (pixels_Q.size(-1) ** 0.5)

    # 屏蔽不在范围内的值
    Score_Superpixel = Score_Superpixel * mask_array
    Score_pixel = Score_pixel * mask_array.T

    # 0 值换成 最大值
    max_value = 1e16
    Score_Superpixel = torch.where(Score_Superpixel == 0, max_value, Score_Superpixel)
    Score_pixel = torch.where(Score_pixel == 0, max_value, Score_pixel)

    # softmax化
    Score_Superpixel = (-Score_Superpixel).softmax(2)
    Score_pixel = (-Score_pixel).softmax(2)

    Add_spixel_features = torch.bmm(Score_Superpixel, pixels_V)
    Add_pixel_features = torch.bmm(Score_pixel, spixels_V)

    # 更新:  I(i+1) = I(i) + softmax (q_I(i) * k_S(i)) * V_S(i)
    spixel_features = spixel_features.permute(0, 2, 1)
    spixel_features = spixel_features + Add_spixel_features
    spixel_features = spixel_features.permute(0, 2, 1)

    permute_pixel_features = permute_pixel_features.permute(0, 2, 1)
    permute_pixel_features = permute_pixel_features + Add_pixel_features
    permute_pixel_features = permute_pixel_features.permute(0, 2, 1)

    pixels_Q, pixels_K, pixels_V = self.SelfAttenion(permute_pixel_features)
    spixels_Q, spixels_K, spixels_V = self.SelfAttenion(spixel_features)

# 最终权重矩阵Q的返回:
Score = torch.bmm(spixel_features.permute(0,2,1), permute_pixel_features) / (spixel_features.size(1) ** 0.5)
Score = Score_Superpixel = Score_Superpixel * mask_array
max_value = 1e16
Score = torch.where(Score_Superpixel == 0, max_value, Score_Superpixel)
# Score:[batch_size, Spixel_num, Pixel_num]
Score = (-Score).softmax(2)

return spixel_features, Score
```

图 8. 超像素聚类代码示意

上图说明了超列向量是如何通过聚类迭代形成超像素的过程。在初始化超像素中心后，将超列向量分解成 Q、K、V 矩阵，通过注意力得分计算它们之间的相似度，并持续更新超像素中心。最终得到一个权值矩阵，表示超像素与像素之间的软链接以及超像素向量。这个过程利用自注意力机制有效地捕捉了图像中的关联信息，为后续的超像素生成提供了准确而丰富的表达。

- 超像素的分类和映射

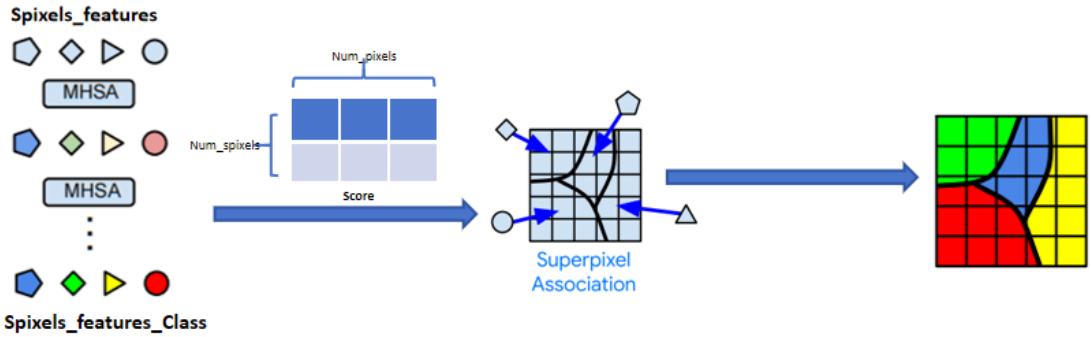

图 9. 分类和映射过程示意

```
# 调整形状以满足 MultiheadAttention 的要求
hypercolumn_superpixel = hypercolumn_superpixel.view(hypercolumn_superpixel.size(0), hypercolumn_superpixel.size(1), -1)
hypercolumn_superpixel = hypercolumn_superpixel.permute(2, 0, 1)  # 将特征维度置于前面
#多头注意力
attention_output, _ = self.attention(hypercolumn_superpixel, hypercolumn_superpixel, hypercolumn_superpixel)
#恢复形状
attention_output = attention_output.permute(1, 2, 0)
attention_output = attention_output.reshape(attention_output.size(0),attention_output.size(1),height,-1)
# FC分类层
hypercolumn_superpixel_class = self.classifier(attention_output.reshape(batchsize,target_num_features,int(super_height),int(super_width)).cuda())
#映射层
output = torch.bmm(hypercolumn_superpixel_class.reshape(batchsize,21,int(super_height * super_width)), Score.cuda()).reshape(batchsize,21,height,width)

return output
```

图 10. 分类映射代码示意

上图展示了超像素进行语义分割的过程。超像素经过两个自注意力层后，再通过全连接网络进行处理，得到每一个超像素的分类结果。接着，通过先前生成的软链接权值矩阵，将分类结果映射到各个像素，最终实现像素级别的语义分割。这一流程充分利用了自注意力机制和全连接网络，为语义分割提供了高效而精确的特征表达。

5 实验结果分析

- 运行结果分析

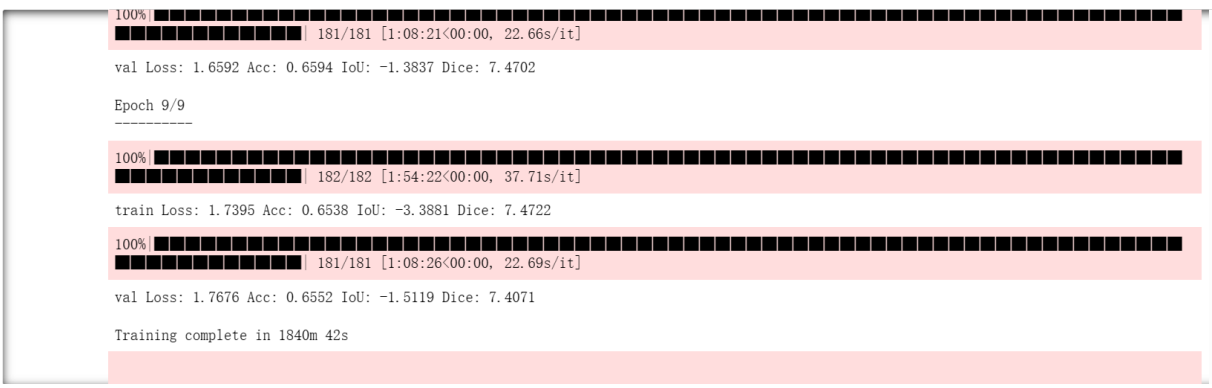

图 11. 训练结果示意

如图所示，训练只设置了 10 个 epoch，运行的时间为 1840 分钟，其中正确率只有百分之 65。对此结果进行分析，首先运行速度慢，是因为在进行聚类迭代形成超像素的时候，只能在 CPU 上进行运行，使得无法并行，大大减小了训练速度，其次，迭代生成的过程速度本来就十分缓慢，是因为迭代的代码没有使用到 C++，而是用了 Python，而且在迭代的数据结构设计上，也并没有有效地优化。因此训练的速度十分缓慢。

- 语义分割结果分析

图 12. 分割结果 1

分类的结果表示网络的具有一定的表达能力，由图所示，图中表示的是模型对背景中的人进行语义分割，其中第一行的图片是原图像。第三行的图片是标签，而第二行是网络进行语义分割的结果。可以看出，在单一类别占比比较大时，网络往往可以较好地对结果进行分类，这表明我们编写的超像素分割模块是正确，网络具有一定的表达能力。

图 13. 分割结果 2

由图所示，图中表示的是模型对背景中的桌子进行语义分割，其中第一行的图片是原图像。第三行的图片是标签，而第二行是网络进行语义分割的结果。可以看出，在单一类别占比较小时，网络无法对该类别进行判别，这证明网络的表达能力有限，网络内部的权值失衡，基于这个现象，我认为有可能是模型的分类层数有限，缺乏表达能力；也有可能是数据集类别的比例不对；或者是模型进行搭建的时候，归一化层数设置得不够多。

6 总结与展望

在这次的论文复现实验中，我完成了无源码项目的编写，根据论文提供的方法逐步构建了网络。这次实践大大提高了我的工程代码能力，深入了解了超像素分割的相关知识，熟悉了一系列超像素分割的算法，为今后的科研打下了坚实的基础。尽管在超像素分割领域，目前的相关工作相对较多，但近几年仍然缺乏突破性的进展，主要依赖聚类算法生成超像素。探索如何通过像素特征直接生成超像素仍然具有重要意义。此外，将超像素分割扩展到三维空间，尤其是关于超体素分割的研究相对较少。然而，通过过分割操作形成超体素或者超像素

可以显著提高运算速度，具有实用的潜在价值。因此，过分割问题仍然有很大的探索空间。希望每一位与这一领域相关的研究者都能早日取得突破性的成果，推动该领域实现巨大的进步。

至此，我的论文复现报告到此结束。

参考文献

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[3] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv:2102.04306*, 2021.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. In *arXiv:1706.05587*, 2017.

[6] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[8] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020.

[9] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022.

[10] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.

[11] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[15] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[16] Raghudeep Gadde, Varun Jampani, Martin Kiefel, Daniel Kappler, and Peter V Gehler. Superpixel convolutional networks using bilateral inceptions. In *ECCV*, 2016.

[17] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] Xuming He, Richard S Zemel, and Miguel A Carreira-Perpinán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[20] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *ECCV*, pages 352–368, 2018.

[21] Philipp Kr¨ahenb¨uhl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NeurIPS*, 2011.

[22] L' ubor Ladick'y, Chris Russell, Pushmeet Kohli, and Philip HS Torr. Associative hierarchical crfs for object class image segmentation. In *ICCV*, 2009.

[23] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[24] Yann LeCun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Ping Luo, and Tong Lu. Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In *CVPR*, 2022.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[27] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

[28] Stuart Lloyd. Least squares quantization in pcm. volume 28, pages 129–137, 1982.

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[30] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[31] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 888–905, 2000.

[32] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

[33] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

[35] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021.

[36] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020.

[37] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[38] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *NeurIPS*, 2020.

[39] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[40] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021.

[41] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022.

[42] Qihang Yu, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In *CVPR*, 2022.

[43] Qihang Yu, Huiyu Wang, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. k-means mask transformer. In *ECCV*, 2022.

[44] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020.

[45] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. In *NeurIPS*, 2021.

[46] Yifan Zhang, Bo Pang, and Cewu Lu. Semantic segmentation by early region proxy. In *CVPR*, 2022.

[47] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[48] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, and et al. Philip HS Torr. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.

[49] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[50] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020.