Adaptive Prototype Learning and Allocation for Few-Shot Segmentation

Gen Li[1,3], $VarunJampani^2$, $LauraSevilla-Lara^1$, $DeqingSun^2$, $JonghyunKim^3$, $JoongkyuKim^{3*}$

[1] $UniversityofEdinburgh$, [2] $GoogleResearch$, [3] $SungkyunkwanUniversity$

摘要

原型学习被广泛应用于少样本分割。通常通过对全局目标信息进行平均，从支持特征中获得单个原型。然而，使用一个原型来表示所有的信息可能会导致歧义。在本文中，我们提出了两个新的模块，即超像素引导聚类（SGC）和引导原型分配（GPA），用于多种原型的提取和分配。具体来说，SGC 是一种无参数、无训练的方法，它通过聚合相似的特征向量来提取更有代表性的原型，而 GPA 能够选择匹配的原型来提供更准确的指导。通过将 SGC 和 GPA 集成在一起，我们提出了自适应超像素引导网络（ASGNet），这是一个轻量级的模型，可以适应物体的规模和形状变化，并且可以很容易地推广到 k-shot 分割，有实质性的改进还没有额外的计算成本。

关键词：少样本分割；超像素；原型学习；自适应；原型分配

1 引言

人们只通过少数样本就可以学习如何识别新物体。目前而言，尽管基于深度学习的计算机视觉系统取得巨大的进步，但在很大程度上依赖于大规模的训练集。此外，深度网络大多使用预定义的类来工作，并且不能推广到新的类。

在这项工作中，我们解决了少样本分割问题，其中目标是在只有少数具有真实分割掩膜的支持图像可用的情况下，学习分割给定查询图像中的对象。这是一个具有挑战性的问题，因为测试数据是训练集中不存在的新类别，并且支持图像和查询图像之间的外观和形状通常存在很大差异。

目前的少样本分割网络通常从查询图像和支持图像中提取特征，然后提出不同的特征匹配方法和从支持图像的目标掩膜迁移方法。这种特征匹配和掩膜转移通常通过原型特征学习或亲和学习进行。原型学习技术将支持图像中的掩膜对象特征压缩为单个或几个原型特征向量，然后在查询图像中找到相似特征的像素位置来分割所需的对象。原型学习的一个关键优点是原型特征比像素特征对噪声具有鲁棒性。然而，原型特征不可避免地会丢失空间信息，这在支持图像和查询图像之间的对象外观变化很大时是很重要的。此外，大多数的原型学习网络仅仅通过掩膜平均池生成一个单一的原型，如图 1(a) 所示，从而丢失了信息和可辨别性。

在这项工作中，我们提出了一种新的原型学习技术，它解决了现有的原型学习技术的一些主要缺点。特别是我们希望根据图像内容自适应地改变原型的数量及其空间范围，使原型的内容具有自适应性和空间感知能力。这种自适应的多原型策略对于处理不同图像之间的物体尺度和形状的巨大变化非常重要。直观地说，当一个物体占据了图像的大部分时，它携带了更多的信息，因此需要更多的原型来表示所有必要的信息。相反，如果对象相当小，而背景的比例很大，那么单个或几个原型就足够了。此外，我们希望每个原型的支持区域（空间范围）能够适应于支持图像中存在的对象信息，能够根据

特征的相似性将支持特征划分为几个具有代表性的区域。我们还希望自适应地选择更重要的原型，同时在查询图像中寻找相似的特征。由于在不同的图像区域和不同的查询图像中对不同的对象部分可见，我们希望在查询图像中动态分配不同的原型进行特征匹配。

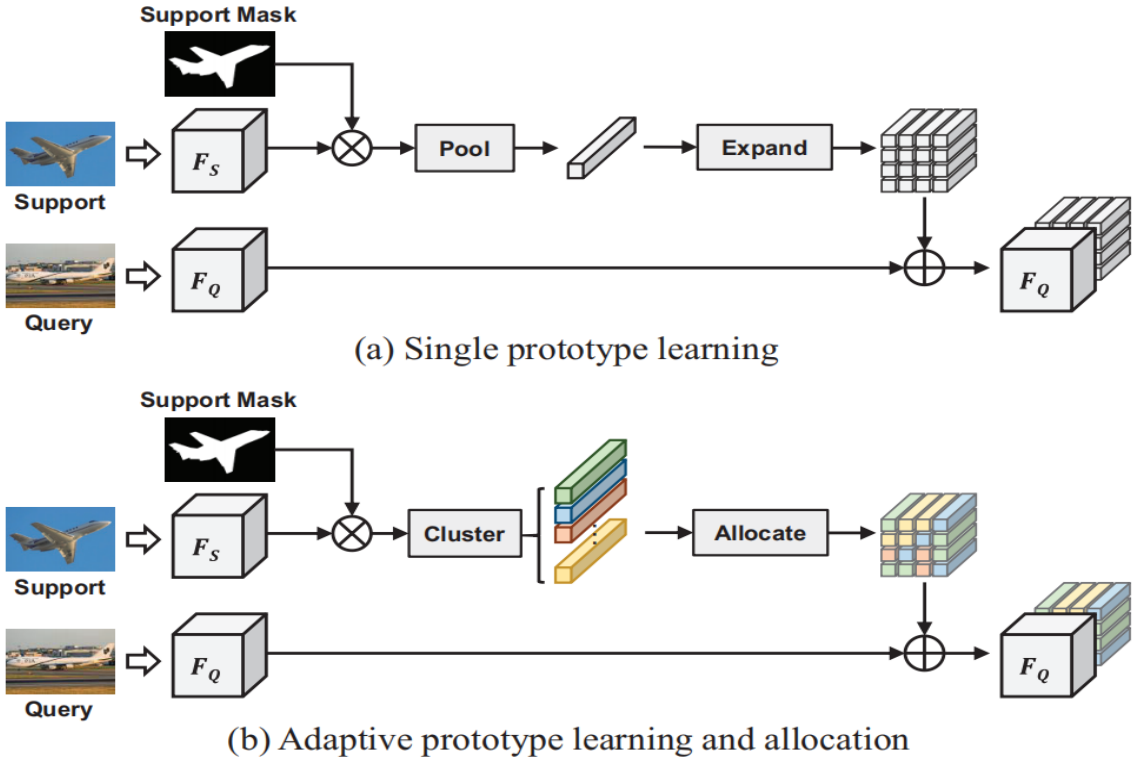

图 1: (a) 单原型学习和 (b) 提出了自适应原型学习和分配

我们通过自适应超像素引导网络（ASGNet）实现了这种自适应的多原型学习和分配，该网络利用超像素来适应原型的数量和支持区域。如图 1(b) 所示。特别地，我们提出了两个模块，超像素引导聚类（SGC）和引导原型分配（GPA），它们构成了 ASGNet 的核心。SGC 模块对支持图像进行快速的基于特征的超像素提取，所得到的超像素质心被视为原型特征。由于超像素的形状和数字是自适应图像内容的，因此产生的原型也成为自适应的。GPA 模块使用一种类似于注意力的机制，将最相关的支持原型特征分配给查询图像中的每个像素。总之，SGC 模块在原型的数量和空间范围方面提供了自适应的原型学习，而 GPA 模块在处理查询特征时提供了学习到的原型的自适应分配。这两个模块使 ASGNet 具有高度的灵活性和自适应不同的对象形状和大小，使它能够更好地推广到不可见的对象类别。本文做出了以下贡献：

- 提出了一种灵活的少样本分割的原型学习方法，能够适应不同的物体尺度、形状和遮挡。
- 引入了两个新的模块，即超像素引导聚类（SGC）和引导原型分配（GPA），分别用于自适应原型的提取和分配。它们可以作为功能匹配上的有效插件。
- ASGNet 以更少的参数和更少的计算量获得了性能最好的结果。

2 相关工作

2.1 语义分割

现有的语义分割方法大多是基于全卷积网络（FCNs）[1]，用全卷积层取代完全连接的层进行像素级预测。近年来，在语义分割方面的突破主要来自于多尺度特征聚合或注意机制。这些方法通常使用

扩展的卷积核，并在保持特征分辨率的同时，建立一个编解码器结构来获得一个较大的感受野。虽然这些方法取得了巨大的成功，但它们需要较长的训练时间和大量的像素级标记来充分监督网络。此外，在推理阶段训练模型不能识别训练集中不存在的新类。

2.2 少样本学习

少样本学习关注模型的泛化能力，因此在只给出一些带注释的例子的情况下也可以学习预测新的类。现有的方法主要集中在度量学习和元学习上。度量学习的核心思想是距离测量，它通常被表述为对图像或区域之间的距离/相似性的优化。在元学习方法中，主要思想是定义特定的优化或损失函数，以实现快速学习能力。在这些方法中，原型网络的概念已被广泛采用在少样本分割中，这在保持高性能的同时大大减少了计算预算。大多数方法集中在图像分类，而最近少样本分割受到越来越多的关注。

2.3 少样本分割

少样本分割是少样本分类的扩展，它解决了一个更具挑战性的任务，即预测每个像素上的一个标签，而不是预测整个图像的单个标签。这个研究问题是由 Shaban 等人[2]提出的，他提出了一个经典的双分支网络。后来，PL[3]介绍了使用原型的想法。在这项工作中，每个预测都是通过测量查询图像中的原型和像素之间的相似性来生成的。SGOne[4]提出掩膜平均池来获得对象相关的原型，在后续工作中得到了广泛的应用。PANet[5]引入了一种新的原型对齐正则化方法，以充分利用支持知识，在仅测量余弦距离的情况下进行最终预测。基于掩膜平均池化，CANet[6]将原型扩展到相同的查询特性的大小，并将它们连接在一起。本工作还使用了一个迭代优化模块，对分割结果进行了细化。PGNet[7]、BriNet[8]和 DAN[9]在支持和查询特征之间引入了密集的像素到像素的连接，以维护空间信息。最近，pmm[10]利用期望最大化（EM）算法来生成多个原型。所有的原型在这个模型中都具有相同的相关性，这可能对不匹配的原型很敏感，而我们利用每个原型和查询特征之间的相似性来在每个像素位置选择最相关的原型。

2.4 超像素分割

超像素被定义为一组具有相似特征（颜色、纹理、类别）的像素。超像素在许多计算机视觉任务中都很有效，并被用作图像分割任务的基本单元。超像素比像素携带更多的信息，可以为下游的视觉任务提供更紧凑和方便的图像表示。

我们的研究是受到了 maskSLIC[11]和超像素采样网络（SSN）[12]的启发。MaskSLIC 将 SLIC[13]适应于一个定义的感兴趣区域（RoI），其主要贡献是在 RoI 中放置种子点。SSN 通过使 SLIC 算法可微，提出了第一个端到端可训练的超像素算法。受这两种技术的启发，我们提出了特征空间中的掩膜超像素聚类，它可以将相似的特征聚集在一起，生成超像素质心作为原型。超像素符号表示整个对象的信息，而不是具有相似特征的对象部分。

3 本文方法

在本节中，首先介绍了所提出的原型生成和匹配的两个模块，即超像素引导聚类（SGC）和引导原型分配（GPA）。然后，讨论了这两个模块的自适应能力。之后介绍了整体的网络架构，即自适应超像素引导网络（ASGNet），它将 SGC 和 GPA 模块集成在一个模型中。整体结构如图 7 所示。最后，在 ASGNet 中详细说明了 k-shot 的设置。

3.1 超像素引导聚类 (SGC)

SGC 的核心思想是受到超像素采样网络 (SSN)[12]和 MaskSLIC[11]的启发，将超像素质心作为原型，但是不计算图像空间中的超像素质心，而是进行估计它们通过在特征空间中聚类相似的特征向量。在算法 1 中描述了整个 SGC 过程。

Algorithm 1 Superpixel-guided Clustering (SGC)

Input: Support feature F_s, support mask M_s, and initial superpixel seeds S^0

concatenate the absolute coordinates to F_s
extract masked features $F_s^{'}$ via support mask M_s
for each iteration t **do**
Compute association between each pixel p and superpixel i, $Q_{pi}^t = e^{-\left\| F_p^{'} - S_i^{t-1} \right\|^2}$
Update superpixel centroids, $S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^{N_m} Q_{pi}^t F_p^{'}$; $Z_i^t = \sum_p Q_{pi}^t$
end for
remove coordinates information
return final superpixel centroids S ▷ (N_{sp}, C)

图 2: 超像素引导聚类（SGC）算法

给定支持特征 $F_s \in \mathbb{R}^{c \times h \times w}$，支持掩膜 $M_s \in \mathbb{R}^{h \times w}$ 和初始超像素种子 $S^0 \in \mathbb{R}^{c \times N_{sp}}$（$N_{sp}$ 是超像素的数量），目标是获得作为多个紧凑原型的动态超像素质心。首先，将每个像素的坐标与支持特征映射连接起来以引入位置信息。然后，在 SLIC 之后定义距离函数 D，它由特征和空间距离组成：

$$D = \sqrt{(d_f)^2 + (d_s/r)^2} \tag{1}$$

其中，d_f、d_s 为特征和坐标值的欧氏距离，r 为加权因子。我们使用支持掩膜过滤掉背景信息，只保留掩膜特征，将特征映射从 $F_s \in \mathbb{R}^{c \times h \times w}$ 压缩到 $F_s' \in \mathbb{R}^{c \times N_m}$，其中 N_m 是支持掩膜内的像素数。

然后以迭代的方式计算基于超像素的原型。对于每次迭代 t，首先根据距离函数 D，计算每个像素 p 与所有超像素之间的关联映射 Q^t:

$$Q_{pi}^t = e^{-D\left(F_p', S_i^{t-1}\right)} = e^{-\left\| F_p' - S_i^{t-1} \right\|^2} \tag{2}$$

然后，将新的超像素质心更新为掩膜特征的加权和：

$$S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^{N_m} Q_{pi}^t F_p' \tag{3}$$

其中 $Z_i^t = \sum_p Q_{pi}^t$ 是一个归一化的常数。SGC 算法可视化过程如图 3 所示。

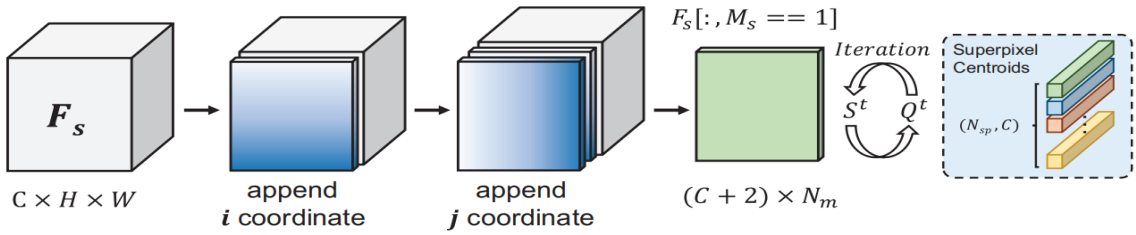

图 3: 超像素引导聚类过程

接着详细阐述初始种子的选择。在超像素算法中，一般将 $H \times W$ 图像均匀划分为大小为 $h \times w$ 的规则网格单元，并将每个网格单元视为初始种子（即超像素）。然而，这种初始化并不适用于所提出的方法，因为支持图像给出了一个前景掩膜，而我们只需要初始化这个前景区域内的种子。为了在掩膜区域中统一初始化种子，引用 MaskSLIC[11]来迭代地放置每个初始种子，初始种子初始管道图如图 4 所示。通过距离变换选择最大点，该点设置为 false 后重新选择最大点，不断迭代直到获取所有初始种子的坐标，这种种子初始化帮助了超像素引导聚类的快速收敛。

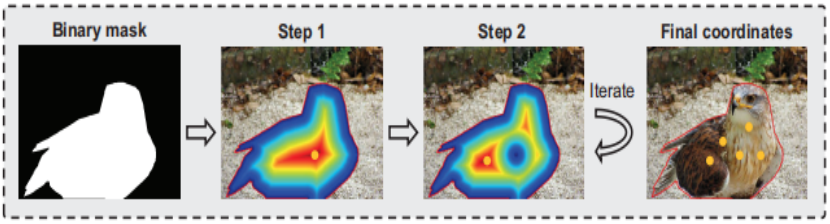

图 4: 初始种子流程

3.2 引导原型分配 (GPA)

在提取原型之后，以前的方法大多遵循 CANet[6]的设计，将单个原型扩展到查询特征的相同大小，并将它们连接在一起。然而，该操作会导致对查询功能中的每个位置的引导是相等的。为了更好地匹配更适合查询图像内容的原型，提出引导原型分配 (GPA) 算法，如图 5 所示。我们首先计算余弦距离来度量每个原型和每个查询特征元素之间的相似性。

$$\mathrm{C}_i^{x,y} C_i^{x,y} = \frac{S_i \cdot F_q^{x,y}}{\|S_i\| \cdot \|F_q^{x,y}\|} \quad i \in \{1, 2, \ldots, N_{sp}\} \tag{4}$$

其中，$S_i \in \mathbb{R}^{c \times 1}$ 是第 i 个超像素质心（原型），$F_q^{x,y} \in \mathbb{R}^{c \times 1}$ 是查询特征位置 (x，y) 处的特征向量。使用相似性信息作为一个双分支结构的输入。第一个分支计算在每个像素位置上最相似的原型如下：

$$\mathrm{G}^{x,y} = \underset{i \in \{0,\ldots,N_{sp}\}}{\arg\max}\ C_i^{x,y} \tag{5}$$

再采用 argmax 算子得到 $G^{x,y}$，即是表示特定原型的单个索引值。将所有索引值放在一起，我们得到一个引导图 $G \in \mathbb{R}^{h \times w}$。然后，通过在每个引导图的位置，我们得到引导特征 $F_G \in \mathbb{R}^{c \times h \times w}$，实现逐像素引导。在另一个分支，所有相似性信息 C 相加所有超像素得到概率图 P。最后，将概率图和引导功能与原始查询功能 F_Q 提供引导信息，从而得到细化查询特征 F'_Q：

$$\mathrm{F}'_Q = f\left(F_Q \oplus F_G \oplus P\right) \tag{6}$$

其中，$\oplus$ 表示沿信道维数的连接操作，$f(\cdot)$ 为 1×1 的卷积。

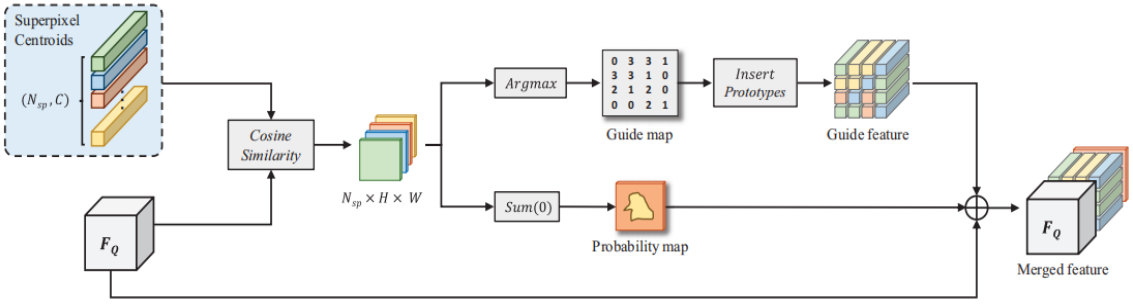

图 5: 引导原型分配（GPA）模块

3.3 自适应性

ASGNet 网络的关键属性之一是其对少样本语义分割的自适应能力。在图 6 中，提供了一些例子来说明 SGC 和 GPA 的自适应能力。在 SGC 中，为了使其适应目标尺度，我们定义了一个调节超像素质心数量的准则为：

$$N_{sp} = \min\left(\left\lfloor \frac{N_m}{S_{sp}} \right\rfloor, N_{\max}\right) \tag{7}$$

其中 Nm 是支持掩膜中的像素数；Ssp 是分配给每个初始超像素种子的平均面积，根据经验设置为 100。当前景相当小，即 $Nsp = 0$ 或 1 时，该方法会退化为一般的掩膜平均池，如图 6(a) 所示。此外，为了减少计算负担，设置了一个超参数 Nmax 来约束原型的最大数量。在 GPA 中，可以观察到它对物体形状的适应性。换句话说，它对咬合很有弹性。当查询图像存在严重遮挡时，例如图 6(b)，GPA 可以为每个查询特征位置选择最佳匹配的原型。

(a) SGC is adaptive to object scale variation　　(b) GPA is adaptive to object shape variation

图 6: 模型在原型学习和分配中的适应性说明

3.4 自适应超像素引导网络 ASGNet

基于 SGC 和 GPA 模块，提出了自适应超像素引导网络（ASGNet）进行少样本语义分割，如图 7 所示。首先，支持和查询图像被输入一个共享的 CNN（在 ImageNet[14]上预先训练）来提取特征。然后，通过将支持特征通过带有支持掩膜的 SGC 进行传递，得到了作为原型的超像素质心。之后，为了获得更准确的像素级引导，我们采用 GPA 模块将原型与查询特征进行匹配。最后，利用特征富集模块[15]，建立了一个类似 FPNs[16]自顶向下结构来引入多尺度信息，从细到粗促进了特征交互。最后，将所有不同的尺度连接起来，每个尺度产生一个计算损失的分割结果。

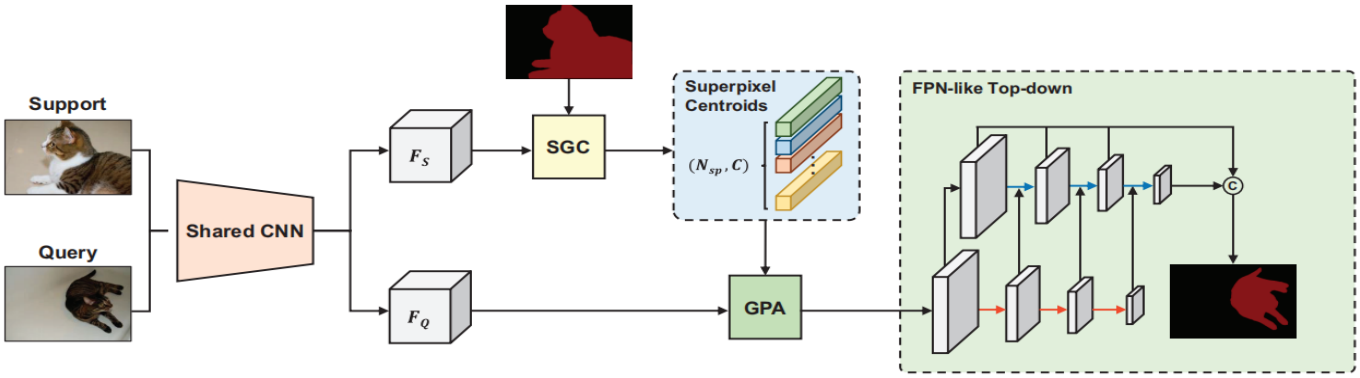

图 7: 自适应超像素引导网络图 (ASGNet)

4 复现细节

4.1 注意力优化模块 AMM

当使用模型识别图像时，一般是通过卷积核去提取图像的局部信息，然而，每个局部信息对图像能否被正确识别的影响力是不同的，为了让模型知道图像中不同局部信息的重要性，因此引入了注意力机制。设计了一个注意调制模块（AMM），通过按顺序在通道和空间维度上进行特征调制。利用一个调制功能来重新安排特征的分布，它试图强调次要特征。通道-空间顺序方式有助于明确地建模每一层局部接受域内的通道级相互依赖和空间编码，以自适应地调节面向分割的特征响应。关键就是学出一个权重分布，然后作用在特征上，在这里我们作用在空间尺度上，也作用于通道尺度上。AMM 框架如图 8 所示。

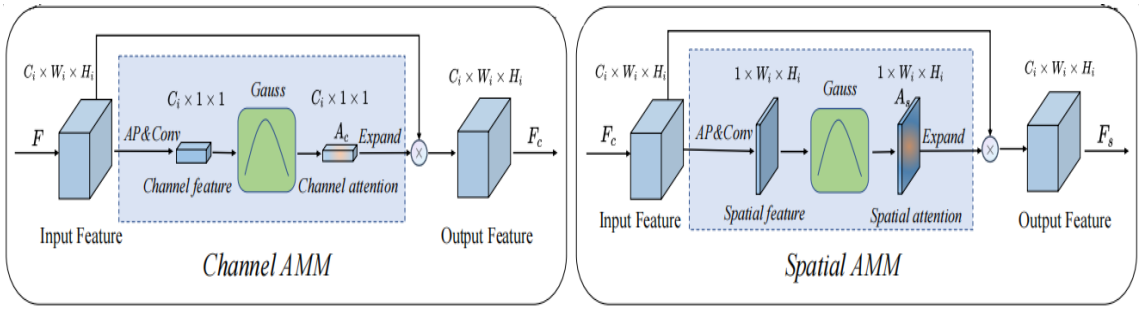

图 8: AMM 框架图

AMM 模块由通道注意调制和空间注意调制组成。我们首先将特征 F(I) 馈送给通道 AMM。信道的相互依赖由平均池化和卷积层明确地建模，这反映了对信息特征的敏感性。受 Jiang[17]的启发，最敏感的特征对应于区分区域，次要特征表示重要但容易被忽略的区域，而平淡的特征可能表示背景概念。因此，我们利用一个调制函数来增强次要特征，并抑制最敏感和最不敏感的特征。以上操作均可记号为：

$$A_c = \mathcal{G}\left(H\left(P_s(F(I))\right)\right) \tag{8}$$

其中 A_c 是通道注意图。我们表示 P_s 为空间平均池化函数，H 为卷积层。然后利用调制函数 $\mathcal{G}$ 重新分配特征的分布，以突出信道维度中的次要特征。然后，我们在信道注意图和输入特征图之间进行元素级乘法，生成重新分布的特征，定义为：

$$F_c(I) = \tilde{A}_c \odot F(I) \tag{9}$$

其中，$\tilde{A}_c$ 表示扩展到特征图维度的通道注意图。$F_c(I)$ 表示输出的特征映射。为了进一步模拟空

间维度上的空间间关系，我们还引入了一个空间 AMM，在通道 AMM 之后进行级联。具体来说，我们首先在信道维数上 $F_c(I)$ 上使用一个信道平均池化 P_c，然后对它们应用一个卷积操作 H。输出的特征图说明了这些特征在空间维度中的重要性。然后，我们对输出特征映射执行一个调制函数，以增加次要的激活。实施过程可制定为：

$$A_s = \mathcal{G}\left(H\left(P_c\left(F_c(I)\right)\right)\right) \tag{10}$$

$\tilde{A}_s$ 中的高激活值反映了容易被忽略的区域。然后，我们在空间注意图和特征图之间进行元素级乘法, 其中 $\tilde{A}_s$ 表示扩展到特征图维度的空间注意图：

$$F_s(I) = \tilde{A}_s \odot F_c(I) \tag{11}$$

下面为相关注意力机制引入的代码，旨在以通道-空间顺序的方式调节特征的激活图。利用空间层面和通道层面的关注来突出通道和空间维度中的重要线索。通道注意力层面，通过空间平均池化函数以及卷积层，再利用调制函数重新分配特征的分布，以突出信道维度中的次要特征，然后，我们在通道注意图和输入特征图之间进行两个矩阵对应位置元素进行乘积，生成重新分布的特征。

```
class ChannelGate(nn.Module):
    def __init__(self, gate_channels, reduction_ratio=16, pool_types=['avg']):
        super(ChannelGate, self).__init__()
        self.gate_channels = gate_channels
        self.mlp = nn.Sequential(
            Flatten(),
            nn.Linear(gate_channels, gate_channels // reduction_ratio),
            nn.ReLU(),
            nn.Linear(gate_channels // reduction_ratio, gate_channels)
            )
        self.pool_types = pool_types

    def forward(self, x):
        channel_att_sum = None
        for pool_type in self.pool_types:
            if pool_type=='avg':
                avg_pool = F.avg_pool2d( x, (x.size(2), x.size(3)), stride=(x.size(2), x.size(3)))
                channel_att_raw = self.mlp( avg_pool )
            elif pool_type=='max':
                max_pool = F.max_pool2d( x, (x.size(2), x.size(3)), stride=(x.size(2), x.size(3)))
                channel_att_raw = self.mlp( max_pool )

            if channel_att_sum is None:
                channel_att_sum = channel_att_raw
            else:
                channel_att_sum = channel_att_sum + channel_att_raw

        # Gauss modulation
        mean = torch.mean(channel_att_sum).detach()
        std = torch.std(channel_att_sum).detach()
        scale = GaussProjection(channel_att_sum, mean, std).unsqueeze(2).unsqueeze(3).expand_as(x)

        #scale = scale / torch.max(scale)
        return x * scale

class ChannelPool(nn.Module):

    def forward(self, x):
        return torch.cat( (torch.max(x,1)[0].unsqueeze(1), torch.mean(x,1).unsqueeze(1)), dim=1 )
```

图 9: AMM 模块代码 1

紧接着为了进一步模拟空间维度上的空间间关系，还引入了一个空间注意力模块，在通道注意力模块之后进行级联。首先在通道模块后使用通道平均池化，然后进行卷积操作。输出的特征图说明了这些特征在空间维度中的重要性。同样地在空间注意图和特征图之间进行两个矩阵对应位置元素进行乘积。

```
class SpatialGate(nn.Module):
    def __init__(self):
        super(SpatialGate, self).__init__()
        kernel_size = 7
        self.pool = ChannelPool()
        self.spatial = BasicConv(2, 1, kernel_size, stride=1, padding=(kernel_size-1) // 2, relu=False)

    def forward(self, x):
        x_pool = self.pool(x)
        x_out = self.spatial(x_pool)

        # Gauss modulation
        mean = torch.mean(x_out).detach()
        std = torch.std(x_out).detach()
        scale = GaussProjection(x_out, mean, std)

        #scale = scale / torch.max(scale)
        return x * scale
```

图 10: AMM 模块代码 2

最后基于实现好的通道注意力模块和空间注意力模块进行级联完成 AMM 模块，按照顺序在通道和空间维度上进行特征调制从而重新安排特征的分布，使得模型分割性能得到提升。

```
class AMM(nn.Module):
    def __init__(self, gate_channels, reduction_ratio=16, pool_types=['avg']):
        super(AMM, self).__init__()
        self.ChannelAMM = ChannelGate(gate_channels, reduction_ratio, pool_types)
        self.SpatialAMM = SpatialGate()

    def forward(self, x):
        x_out = self.ChannelAMM(x)
        x_out = self.SpatialAMM(x_out)
        return x_out
```

图 11: AMM 模块代码 3

但是经过实验发现引入注意力机制所产生的优效果化不理想，因此我们调整优化方向利用先验掩膜模块进行实验，取得了不错的优化效果。

4.2 运用先验掩膜优化 ASGNet

现有的少样本分割方法普遍存在的问题包括高级特征的误用导致的泛化损失以及查询样本和支持样本之间的空间不一致。如在 CANet[6]的实验中发现了高层特征会导致最终表现下降。原因在于包含在高级特征中的语义信息比包含在中级特征中的语义信息更特定于某个类，这表明高级特征更有可能对模型对不可见类的泛化能力产生负面影响。但是这一矛盾促使我们寻找其他的方式利用高级信息的方法，以提高在少样本分割中的准确率。因此提出了一种先验掩膜优化的方法。先验生成方法通过利用对预先训练好的高级特征的余弦相似度计算来提高性能。先验掩膜鼓励模型更好地对查询目标进行定位，而不会失去泛化能力。

先验掩膜代表像素属于目标类的概率，作为一个指标，突出了未知类别的感兴趣的区域。具体来

说，揭示查询特征和支持特征之间像素级的对应关系。掩膜上的一个高值像素表明对应的查询像素与支持特征中的至少一个像素具有高对应关系，所以此像素很可能处于查询图像的目标区域。

正确使用先验知识能够使模型更好地识别医学图像中所通常具有特定的形状和特征，进而帮助模型更好地泛化到新数据，模型也会从先验知识中学到有用的信息进而减少所需要的训练数据量，很好的应对医学图像中数据集少、标注费时且困难的场景，再对信息的充分利用使得模型能够在更短的时间内达到较高的训练准确度。

X_Q 和 X_S 分别代表高层查询图像特征和高层支持图像特征，假定作为 X_Q 先验掩膜的 Y_Q 能够揭示 X_Q 和 X_S 之间的像素对应关系，X_Q 中具有高值的像素意味着这个像素在支持特征 X_S 中至少有一个像素与之具有高对应性。因此它很可能在查询图像的目标区域，且通过将支持特征的背景设置为零，查询特征的像素与支持特征的背景没有对应关系，它们只与前景目标区域相关。首先计算 X_Q 和 X_S 每个像素间的余弦相似度 $\cos(x_q, x_S)$：

$$\cos(x_q, x_s) = \frac{x_q^T x_x}{\|x_q\| \|x_s\|} q, s \in \{1, 2, \ldots, hw\} \tag{12}$$

再将所有支持像素间的最大相似度作为响应值为 c_q：

$$c_q = \max_{s \in \{1,2,\ldots,hw\}} (\cos(x_q, x_s)) \tag{13}$$

$$C_Q = [c_1, c_2, \ldots, c_{hw}] \in R^{hw \times 1} \tag{14}$$

将 C_Q 转换为 h×w 的形式，并对其做归一化处理即得到先验掩膜 Y_Q，这一方法的关键在于利用固定的高级特征生成先验掩膜，从大小为 hw × hw 的相似矩阵中取最大值。

$$Y_Q = \frac{Y_Q - \min(Y_Q)}{\max(Y_Q) - \min(Y_Q) + \varepsilon} \tag{15}$$

对计算出来的先验掩膜特征拼接，把三类特征拼接并且卷积处理后得到各尺度的查询特征, 再进行多尺度特征融合, 得到最终的分割图，加入先验掩膜模块后的 ASGNet 网络图如图 12 所示。

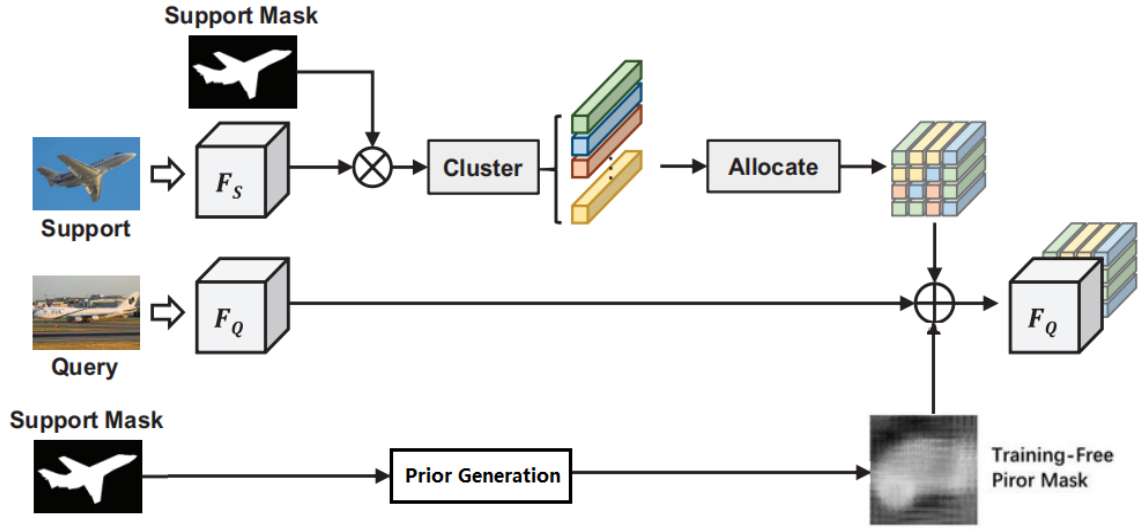

图 12: 先验掩膜改良后的 ASGNet

ASGNet 结合先验掩膜的优化代码如图 13 与图 14 所示，计算出先验掩膜后，与 ASGNet 生成的引导特征和查询特征拼接并且卷积处理后得到各尺度的查询特征, 再进行多尺度特征融合，最终得到分割结果。

```
# 生成先验掩膜
corr_query_mask_list = []
cosine_eps = 1e-7
for i, tmp_supp_feat in enumerate(final_supp_list):
    resize_size = tmp_supp_feat.size(2)
    tmp_mask = F.interpolate(mask_list[i], size=(resize_size, resize_size), mode='bilinear', align_corners=True)

    tmp_supp_feat_4 = tmp_supp_feat * tmp_mask
    q = query_feat_4
    s = tmp_supp_feat_4
    bsize, ch_sz, sp_sz, _ = q.size()[:]

    tmp_query = q
    tmp_query = tmp_query.contiguous().view(bsize, ch_sz, -1)
    tmp_query_norm = torch.norm(tmp_query, 2, 1, True)

    tmp_supp = s
    tmp_supp = tmp_supp.contiguous().view(bsize, ch_sz, -1)
    tmp_supp = tmp_supp.contiguous().permute(0, 2, 1)
    tmp_supp_norm = torch.norm(tmp_supp, 2, 2, True)

    similarity = torch.bmm(tmp_supp, tmp_query) / (torch.bmm(tmp_supp_norm, tmp_query_norm) + cosine_eps)
    similarity = similarity.max(1)[0].view(bsize, sp_sz * sp_sz)
    similarity = (similarity - similarity.min(1)[0].unsqueeze(1)) / (
                similarity.max(1)[0].unsqueeze(1) - similarity.min(1)[0].unsqueeze(1) + cosine_eps)
    corr_query = similarity.view(bsize, 1, sp_sz, sp_sz)
    corr_query = F.interpolate(corr_query, size=(query_feat_3.size()[2], query_feat_3.size()[3]),
                               mode='bilinear', align_corners=True)
    corr_query_mask_list.append(corr_query)
corr_query_mask = torch.cat(corr_query_mask_list, 1).mean(1).unsqueeze(1)
corr_query_mask = F.interpolate(corr_query_mask, size=(query_feat.size(2), query_feat.size(3)), mode='bilinear',
                                align_corners=True)
```

图 13: 先验掩膜生成代码

```
corr_query_mask_list = []
cosine_eps = 1e-7
for i, tmp_supp_feat in enumerate(final_supp_list):
    resize_size = tmp_supp_feat.size(2)
    tmp_mask = F.interpolate(mask_list[i], size=(resize_size, resize_size), mode='bilinear', align_corners=True)

    tmp_supp_feat_4 = tmp_supp_feat * tmp_mask
    q = query_feat_4
    s = tmp_supp_feat_4
    bsize, ch_sz, sp_sz, _ = q.size()[:]

    tmp_query = q
    tmp_query = tmp_query.contiguous().view(bsize, ch_sz, -1)
    tmp_query_norm = torch.norm(tmp_query, 2, 1, True)

    tmp_supp = s
    tmp_supp = tmp_supp.contiguous().view(bsize, ch_sz, -1)
    tmp_supp = tmp_supp.contiguous().permute(0, 2, 1)
    tmp_supp_norm = torch.norm(tmp_supp, 2, 2, True)

    similarity = torch.bmm(tmp_supp, tmp_query) / (torch.bmm(tmp_supp_norm, tmp_query_norm) + cosine_eps)
    similarity = similarity.max(1)[0].view(bsize, sp_sz * sp_sz)
    similarity = (similarity - similarity.min(1)[0].unsqueeze(1)) / (
                similarity.max(1)[0].unsqueeze(1) - similarity.min(1)[0].unsqueeze(1) + cosine_eps)
    corr_query = similarity.view(bsize, 1, sp_sz, sp_sz)
    corr_query = F.interpolate(corr_query, size=(query_feat_3.size()[2], query_feat_3.size()[3]),
                               mode='bilinear', align_corners=True)
    corr_query_mask_list.append(corr_query)
corr_query_mask = torch.cat(corr_query_mask_list, 1).mean(1).unsqueeze(1)
corr_query_mask = F.interpolate(corr_query_mask, size=(query_feat.size(2), query_feat.size(3)), mode='bilinear',
                                align_corners=True)
```

图 14: ASGNet 结合先验掩膜优化代码

4.3 实验环境搭建

4.3.1 文件夹概述

data/ ：包括配置文件和训练/验证列表文件

model/ ：包括相关的模型和模块

tool/ ：包括训练和测试脚本

util/ ：包括数据处理、种子初始化

4.3.2 使用方法

4.3.2.1 依赖

python==3.7, torch==1.6, scipy, opencv-python, tensorboardX

4.3.2.2 数据集

Pascal-5i[2] (VOC 2012, SBD)：包括来自 PASCAL VOC 2012[18]的图像和来自 SBD[19]的额外注释。总共 20 个类别平均分为四个部分。本实验数据集中带分割标注图 12031 张，训练集总数 5953 张，测试集总数 1449 张。

4.3.2.3 预训练模型

预训练的主干网络和模型可以在谷歌 Driver 中找到，下载主干网络并将 pth 文件放在 initmodel/文件夹下。下载链接：https://drive.google.com/drive/folders/1dEJL_KSkZZ0nIEy6zwqqb93L4zBDvCV-。

4.3.2.4 测试和训练

(1) 在数据/配置文件中指定数据集和预训练模型的路径。

(2) 使用 sh tool/test.sh|train.sh data model split_backbone 命令。例如在 PASCAL-5i 的 0split 上用 ResNet50 测试 ASGNet 的命令为 sh tool/test.sh pascal asgnet split0_resnet50。

4.4 创新点

针对大多数的原型学习网络仅仅通过掩膜平均池生成一个单一的原型，从而丢失了信息和可辨别性的问题，提出了一种灵活的少样本分割的原型学习方法，能够适应不同的物体尺度、形状和遮挡。同时还引入了两个新的模块，分别是超像素引导聚类（SGC）和引导原型分配（GPA）模型，分别用于自适应原型的提取和分配，并且 ASGNet 能以更少的计算量得出性能更佳的结果。

因为在网络当中合理地加入先验知识能使模型获得更好的分割效果，所以在原来的 ASGNet 模型中引入了先验掩膜生成模块（Prior Generation）。该模块利用查询和支持图像的高层特征通过余弦相似度计算来生成模型的先验掩膜，所用的高层特征是通过预先训练得到的，所以生成先验掩膜的过程中并没有增加额外的训练过程，而且生成的模型不会失去对未见类的泛化能力，通过先验掩膜有助于模型更好地识别查询图像和对查询目标进行定位，极大地提高了预测精度，保持了高泛化性。

此外，实验结果表明，加入先验掩膜生成模块后得到的分割效果要比之前好很多，而且评估指标 mIou 与 FBIou 都有所提升，整体上来说较原来的 ASGNet 模型的效果要优一些。

5 实验结果分析

在原 ASGNet 与经过先验掩膜优化后的 ASGNet 中利用 pascal 数据集进行实验，得出实验结果如表 1 所示。表中列出了相同数据集在 PFENet[15]、原 ASGNet 及我的复现与改进后的分割效果，并对这些实验数据进行对比分析。在复现实验过程中发现对 ASGNet 进行的复现结果与原论文的结果不一致。通过分析得知，所采用的 pascal 数据集存在少部分标注图片缺失，导致实验结果与原来论文中的有些许差距，但是由于改进过程仍然使用的是相同的数据集，所以复现结果与改进结果仍具有可比性。

通过实验结果的对比分析得知，在 1shot 的情况下，通过先验掩膜优化后的 ASGNet，在四个 split 的结果中，平均 mIoU 比原 ASGNet 有了较好的优化效果 (53.50% vs 52.54%), 在 FB-IoU 同样也有不错的优化效果 (67.38% vs 68.37%)。在跑 5shot 的实验时，因为受实验室显卡性能限制，且训练速度过慢，因此只有 0-split 的对比结果，但通过对比实验结果发现，5shot 的情况下，经过先验掩膜优化后的 ASGNet 仍然比原 ASGNet 分割性能高，因此整体来说，对模型的改进是成功的，加入先验掩膜模块的加入确实使得 ASGNet 网络分割性能提升了。

表 1: 实验结果统计

复现对比	1shot						5shot	
	0-split	1-split	2-split	3-split	mean		0-split	
	mIoU	mIoU	mIoU	mIoU	mIoU	FB-IoU	mIoU	FB-IoU
PFENet	61.70	69.50	55.40	56.30	60.80	73.3	63.10	73.9
ASGNet	58.85	67.86	56.79	53.66	59.29	69.2	63.66	74.2
复现	53.09	63.26	44.33	49.49	52.54	67.38	61.58	78.15
改进后	54.95	64.24	44.61	50.58	53.50	68.37	63.55	79.11

ASGNet 分割实例与经过先验掩膜优化后的 ASGNet 分割实例对比如图 15 所示。通过鸟、自行车的分割效果对比分析可看出，经过先验掩膜优化后，对所需要分割的物品边缘把握更细致，并且由轮船的分割效果对比来看，经过先验掩膜优化的交并比 (IoU) 比原 ASGNet 要高，充分证明了加入先验掩膜优化之后 ASGNet 的分割效果要优于原来的 ASGNet 网络。

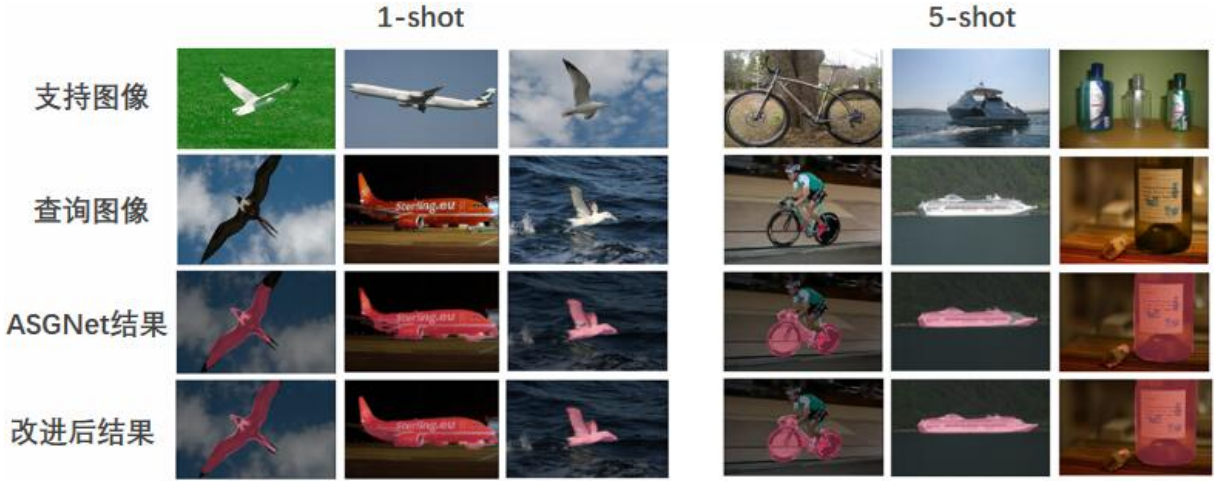

图 15: 原 ASGNet 分割实例与改进后的 ASGNet 分割实例对比

6 总结与展望

近年来小样本学习已经得到深入研究，并且取得了一定进展，在“Adaptive Prototype Learning and Allocation for Few-Shot Segmentation”论文中提到了一种新的少样本图像分割新方法，该论文针对当前原型学习中原型特征不可避免地会丢失空间信息，以及大多数原型学习网络仅通过掩膜平均池生成单个原型，失去了信息的辨别能力的两个问题提出了两个模块，即超像素引导聚类 (SGC) 和引导原型分配模块 (GPA)，用于多种原型的提取和分配，从而使 ASGNet 具有高度的灵活性和自适应不同的对象形状和大小，更好地推广到不可见的对象类别。

为此，基于作者的研究目的，对 ASGNet 当中的两个核心模块 SGC 和 GPC 进行理解和复现，正确选择作者的 python 版本及相关依赖，在下载好指定的主干网络，使用 Pascal-5i (VOC 2012, SBD) 数据集对 ASGNet 进行复现，该数据集包含 20 个对象类，是一个广泛使用的图像分割数据集。再遵循了论文中的实验设置，训练出自己的 ASGNet 模型，使用验证集对训练出来的模型进行验证，以 mIoU 和 FB-IoU 来评估复现 ASGNet 的性能。但是根据复现结果来看性能略于原论文结果，通过分析得知，我们所采用的 pascal 数据集存在部分标注图片缺失，导致分割性能有所下降。并且由于改进花了较长时间探索和实现，以及实验室的显卡配置不足，导致了 5shot 结果没有完全训练和验证完。这也是本次实现过程中的两个不足之处，但实验结果仍具有可比性。

但目前少样本分割面临着一些挑战，如样本数量很少时，模型很难从数据中学习到有用的特征，因此我们可以引入先验知识去提供一些额外的信息，帮助模型更好地理解数据以及更准确的预测，并且基于先验掩膜的约束能使得模型能快速地收敛。根据这一思路，我们基于原 ASGNet 提出了自己的先验掩膜优化策略，在 ASGNet 网络中加入先验掩膜生成模块用来构造先验掩膜，与 ASGNet 生成的引导特征和查询特征拼接并且卷积处理后得到各尺度的查询特征, 再进行多尺度特征融合，最终得到分割结果。经过实验和结果对比，先验掩膜优化后的 ASGNet 在 mIoU 和 FB-IoU 都有了大约 1% 的提升，充分验证了本次改良的有效性。

在未来小样本学习的发展方向应当也是合理利用先验知识训练模型，或者更好地利用无标注数据。为了使小样本学习的概念更靠近真实，可以不依赖模型预训练，也能够取得很好的效果。并且尝试不同小样本学习方法的融合，现有小样本学习模型都是单一使用数据增强或迁移学习的方法，今后可以尝试将二者进行结合，从数据和模型两个层面同时进行改进，以达到更好的效果。

参考文献

[1] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[J]., 2015: 3431-3440.

[2] SHABAN A, BANSAL S, LIU Z, et al. One-shot learning for semantic segmentation[J]. arXiv preprint arXiv:1709.03410, 2017.

[3] DONG N, XING E P. Few-shot semantic segmentation with prototype learning.[J]., 2018, 3(4).

[4] ZHANG X, WEI Y, YANG Y, et al. Sg-one: Similarity guidance network for one-shot semantic segmentation[J]. IEEE transactions on cybernetics, 2020, 50(9): 3855-3865.

[5] WANG K, LIEW J H, ZOU Y, et al. Panet: Few-shot image semantic segmentation with prototype alignment[J]., 2019: 9197-9206.

[6] ZHANG C, LIN G, LIU F, et al. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning[J]., 2019: 5217-5226.

[7] ZHANG C, LIN G, LIU F, et al. Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation[J]., 2019: 9587-9595.

[8] YANG X, WANG B, CHEN K, et al. Brinet: Towards bridging the intra-class and inter-class gaps in one-shot segmentation[J]. arXiv preprint arXiv:2008.06226, 2020.

[9] WANG H, ZHANG X, HU Y, et al. Few-shot semantic segmentation with democratic attention networks [J]., 2020: 730-746.

[10] YANG B, LIU C, LI B, et al. Prototype mixture models for few-shot semantic segmentation[J]., 2020: 763-778.

[11] IRVING B. maskSLIC: regional superpixel generation with application to local pathology characterisation in medical images[J]. arXiv preprint arXiv:1606.09518, 2016.

[12] JAMPANI V, SUN D, LIU M Y, et al. Superpixel sampling networks[J]., 2018: 352-368.

[13] ACHANTA R, SHAJI A, SMITH K, et al. SLIC superpixels compared to state-of-the-art superpixel methods[J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 34(11): 2274-2282.

[14] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[15] TIAN Z, ZHAO H, SHU M, et al. Prior guided feature enrichment network for few-shot segmentation [J]. IEEE transactions on pattern analysis and machine intelligence, 2020.

[16] LIN T Y, DOLLÁR P, GIRSHICK R, et al. Feature pyramid networks for object detection[J]., 2017: 2117-2125.

[17] JIANG P T, HOU Q, CAO Y, et al. Integral object mining via online attention accumulation[J]., 2019: 2070-2079.

[18] EVERINGHAM M, VAN GOOL L, WILLIAMS C K, et al. The pascal visual object classes (voc) challenge[J]. International journal of computer vision, 2010, 88(2): 303-338.

[19] HARIHARAN B, ARBELÁEZ P, BOURDEV L, et al. Semantic contours from inverse detectors[J]., 2011: 991-998.