

目录	1
----	---

目录

1 引言	2
2 相关工作	3
2.1 语义分割	3
2.2 目标检测	3
2.3 伪装目标分析	4
3 方法	5
3.1 基于条件随机场 (CRF) 的伪装目标分割	5
3.1.1 基于数据集的发现	5
3.1.2 一元势能提取	6
3.1.3 基于全连接条件随机场的后处理逻辑	7
3.2 基于结构支持向量机 (SSVM) 的伪装目标分割	10
3.2.1 模型	10
3.2.2 训练	12
3.2.3 推理	12
3.2.4 整数线性规划	13
3.3 基于 AdaBoost 的伪装目标检测	14
3.3.1 基于 Faster R-CNN 的定位器与样本生成器	15
3.3.2 基于 AdaBoost 的分类器	17
4 实验	17
4.1 数据集	17
4.2 伪装目标分割实验结果分析	18
4.2.1 模型训练分析	18
4.2.2 模型性能对比	19
4.2.3 不同 CRF 参数下的性能指标对比	20
4.2.4 可视化分析	21
5 代码	22

1 引言

伪装目标分析 (camouflaged object detection) 是对一张图片中的伪装目标进行检测或者分割，而伪装目标则是指所要分析的目标与背景之间存在颜色、纹理、形状等方面的高度相似性，这种相似性常见于被捕食者改变它们自身的某些特征，使这些特征与周围环境的特征高度相似，从而躲避捕食者的追击。

而与伪装目标分析相对则是显著目标分析 (salient object detection)，显著目标分析所要分析的目标与周围环境在颜色、纹理等特征上存在较大的差异，能够被视觉系统快速、精准的捕捉。

本报告将围绕伪装目标分割与检测展开，主要使用如下三种方法¹：

- **基于条件随机场 (CRF) 的伪装目标分割：**条件随机场模型具有天然的对数据结构建模的能力，所以该方法结合深度全卷积网络来提取每个像素的一元势能，然后通过条件随机场模型对像素间的结构关系进行建模，从而提升分割结果的平滑性和连续性
- **基于结构支持向量机 (SSVM) 的伪装目标分割：**结构支持向量机是一种用于解决结构化预测问题的机器学习方法。本任务使用结构支持向量机来进行伪装目标分割。具体来说，使用深度全卷积网络和结构支持向量机对相邻像素间的关系进行建模。随后通过优化结构化铰链损失函数来学习像素之间的关系，从而提高分割的准确性。
- **基于 AdaBoost 的伪装目标检测：**AdaBoost 算法是一种集成学习方法，通过结合多个弱学习器来构建一个强学习器。在本任务中，伪装目标检测被分为定位器和分类器两个模块。定位器使用 Faster R-CNN 模型来生成区域特征向量以便后续训练 AdaBoost 分类器，从而实现对伪装目标的检测

¹代码实现已经放在了 [github](#) 仓库

2 相关工作

2.1 语义分割

语义分割是许多视觉理解系统中的核心组成部分，其目的是将图像划分为多个区域。一些早期的方法包括阈值分割 [9]、基于直方图的分组、区域增长 [8]、 k 均值聚类 [2] 和分水岭算法 [7] 等。这些方法为语义分割奠定了基础，但在复杂场景中往往存在性能限制。近年来，随着深度学习技术的快速发展，语义分割领域迎来了新的变革。深度学习模型通过学习高维特征表示和端到端的优化策略，大幅提升了分割模型的性能。在多个基准数据集上的实验表明，基于深度学习的语义分割方法通常能够取得最高的精度。

DeepLab [1] 系列是深度学习与传统机器学习相结合用于语义分割的一个典型例子。DeepLab 模型通过引入空洞卷积和空洞空间金字塔池化技术，有效地提升了分割模型的性能。此外，DeepLab 还引入了条件随机场 (CRF) 模型，用于对分割结果进行后处理，提升分割结果的平滑性和连续性。在 PASCAL VOC 2012 数据集上，DeepLab 模型取得了当时最好的分割性能。

2.2 目标检测

候选区域生成 (Region Proposal) 是目标检测中的关键环节，其核心在于高效生成可能包含目标的区域，同时确保高召回率和低计算成本。传统方法如 Selective Search 和 EdgeBoxes 主要依赖低级特征，通过启发式策略生成候选区域，虽然具有一定的准确性，但在处理复杂场景时速度较慢，成为检测流程的瓶颈。随着深度学习的发展，候选区域生成逐渐与卷积神经网络 (CNN) 深度结合，为该领域带来了革命性变化。

R-CNN 系列是候选区域生成与深度学习结合的开端，其中 R-CNN [5] 通过传统算法生成候选区域，再利用 CNN 对区域进行分类和边界框回归。尽管这种方法显著提升了检测性能，但因缺乏共享特征导致计算效率低下。Fast R-CNN [4] 通过引入 RoI 池化层，从共享的卷积特征中提取候选区域特征，实现了检测流程的加速。然而，候选区域生成仍然依赖外部算法，限

制了整体效率。

Faster R-CNN [10] 中的区域提议网络 (Region Proposal Network, RPN) 突破了这一限制，首次将候选区域生成与目标检测网络完全整合。RPN 通过滑动窗口在共享特征图上生成候选区域，利用多尺度 Anchor 框设计，直接回归预测候选区域的坐标及其得分。这种方法不仅避免了传统多尺度图像金字塔的高计算开销，还能在保持高召回率的同时简化计算流程。RPN 通过端到端的训练流程，与检测网络共享特征，在大幅提升效率的同时，显著提高了候选区域生成的质量，使 Faster R-CNN 成为准实时目标检测的典范。

2.3 伪装目标分析

伪装目标分析 (Camouflaged Object Analysis, COA) 的核心在于开发能够有效检测和分割高度隐藏于背景中的目标的算法，其挑战性主要来源于目标与背景之间的高相似性，以及伪装目标在形状、纹理、颜色等特征上的复杂性。

在检测方法方面，SINet 框架 [3] (Search and Identification Network) 提出了一个两阶段的解决方案，其设计灵感来自捕食者的视觉行为。第一阶段的搜索模块通过多尺度特征提取机制识别可能的伪装区域，第二阶段的识别模块通过进一步细化处理来确定目标的边界和结构。SINet 充分利用了低层特征的空间信息和高层特征的语义信息，通过特征融合和注意力机制实现了在复杂场景中的高效伪装目标检测。

在分割方法上，Anabranach 网络 [6] (ANet) 通过双流设计将分类任务与分割任务有机结合，显著提高了分割精度。分类流的主要作用是检测图像中是否存在伪装目标，并生成全局语义信息；分割流则专注于生成像素级的伪装目标分割图。两条流的输出通过特定融合机制进行整合，从而有效提升了对伪装目标的定位和分割能力。值得注意的是，ANet 通过采用卷积神经网络 (CNN) 进行特征提取，并结合全卷积网络 (FCN) 实现像素级预测，其整体架构设计既灵活又高效。此外，ANet 通过显著性检测模型与分割模型的耦合，借鉴显著性目标检测的成功经验，进一步提升了对伪装目标边界的处理能力，特别是在背景复杂、目标边界模糊的情况下表现出色。

3 方法

3.1 基于条件随机场 (CRF) 的伪装目标分割

对于任务一，我将采用如下的流程来完成伪装目标分割任务：

1. **深度全卷积网络来获取每个像素的初始一元势能 (Unary Potential)**：受到 DeepLab [1] 等相关工作的启发，可以利用深度全卷积网络 (DCNN) 本身对于视觉信息的强大处理能力，直接使用经过预处理的原始图像作为模型输入，来进行一元势能的提取，为后续条件随机场提供一个较好的迭代初始点，详细模型请见 3.1.2。
2. **基于条件随机场 (CRF) 的后处理逻辑**：直接使用 DCNN 的输出作为最终分割掩码可能会导致分割边界粗糙、过多的错误分割区域，分割区域不连续等问题，这些问题出现的根本原因是全卷积网络对于输出像素间的结构关系预测能力较差，而这正是条件随机场模型 (CRF) 所擅长建模的。条件随机场模型可以基于一些衡量图像像素特征间特定关系的核函数，来对像素类别的可能取值进行更新与约束，从而使预测分割掩码具有一定的结构性，关于条件随机场模型的更多细节请见 3.1.3。

为了方便起见，我将称该模型为 **DLabCRF**。接下来，我将首先介绍一些观察数据集所得到的发现，随后引出基于这些发现所使用的模型，最后介绍条件随机场模型的后处理逻辑。

3.1.1 基于数据集的发现

从图 1 可以看出，小伪装目标图片占数据集的绝大部分，这种数据的分布会导致以下问题：

1. **难以分割出较小伪装目标**：由于伪装目标占整张图片较小的一部分区域，如图 2，且其与周围环境存在颜色与纹理上的高度相似性，所以模型会非常容易将伪装目标误分类成背景。所以对于小尺度上的特征提取就非常重要。DLabCRF 利用 DeepLab 模型所提出的多尺度图像处理以及空洞金字塔池化 (ASPP) 技巧来缓解该问题所带来的影响。

MO	496	5	326	124	361	260	133
BO	5	30	0	29	9	12	10
SO	326	0	3435	351	1623	1342	1034
OV	124	29	351	898	455	486	329
OC	361	9	1623	455	2315	1290	673
SC	260	12	1342	486	1290	2269	656
IB	133	10	1034	329	673	656	1534
MO	496	5	326	124	361	260	133

图 1: 样本属性分布图 其中每个单元格表示同时拥有 X 和 Y 属性的样本数量, SO 表示小伪装目标样本

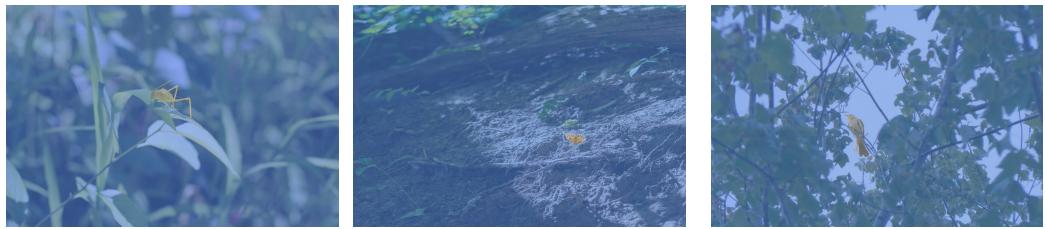


图 2: 小伪装目标图片示例

2. 背景部分的预测会主导模型的训练过程: 对于在包含较小伪装目标的图片上的训练来说, 由于伪装类别占整张图片的较小部分, 所以用于指导模型参数梯度下降的损失值在平均意义上更偏向于使背景图片分类正确, 这样在训练损失值上的体现就是在初期会有一个较大的下降, 而在后期保持几乎不变。模型的输出也仅有很小一部分像素, 或者没有任何像素, 预测为伪装目标。对于该问题的处理, 我采用的是基于类别的加权损失计算, 即针对伪装目标类别给予较高的权重 (0.9), 而对于背景类别则是较低的权重 (0.1)。

3.1.2 一元势能提取

在本任务中, 使用基于 Resnet-101 的全卷积神经网络 [1] 作为提取一元势能的模块, 具体设计细节如下:

空洞卷积的使用: 在深度卷积神经网络 (DCNN) 中, 标准的下采样操作 [11] (例如最大池化或步幅卷积) 会导致特征图的空间分辨率显著下降。

这种降维虽然增强了不变性，但同时降低了空间定位精度，尤其在语义分割任务中会影响对边界和小目标的捕获。为了解决这一问题，DeepLab 模型通过在卷积核之间插入“空洞”（即零值）形成空洞卷积来扩大感受野，同时保持特征图的空间分辨率。这种方法无需增加参数量或计算开销，即可在网络中保留更高的分辨率特征。具体实现中，DeepLab 将 ResNet-101 的最后几层池化操作的步幅设为 1，并使用空洞卷积以确保特征响应的密集性，最后使用双边插值将特征图恢复到原始图像分辨率。

多尺度分割的处理： DeepLab 模型通过两种方式来处理多尺度分割：

1. **多尺度输入：**首先通过将输入图像调整为不同的尺寸（如 $0.5x$ 、 $0.75x$ 、 $1x$ ），然后分别输入到共享参数的 DCNN 中进行特征提取。最后，通过插值将特征图恢复到原始尺寸并采用最大值融合的方式进行多尺度融合。
2. **空洞空间金字塔池化 (ASPP)：**该模块包含多个并行的分支，每个分支使用不同的空洞率 r 和固定的卷积核（如 3×3 ）。具体设置包括：总共有四个不同的空洞率，分别是 6, 12, 18, 24，小空洞率用于捕获细粒度的特征，大空洞率用于捕获全局上下文信息。

在训练阶段，模型分别输出图像在三个尺度下的响应，以及最大值融合响应应用于指导训练过程。在推理阶段，模型只输出最大值融合响应作为预测结果，该响应即为条件随机场的一元势能。

3.1.3 基于全连接条件随机场的后处理逻辑

在语义分割中，DCNN 生成的像素级分类得分图通常是平滑的，但在物体边界处可能会失去细节或出现模糊。为了解决这一问题，可以使用全连接 CRF 模型建模像素间的二元结构关系，增强边界定位并消除伪预测。

能量函数定义 全连接 CRF 使用以下能量函数 $E(x)$ 对像素标签分配进行建模：

$$E(x) = \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j)$$

- **一元势能** $\theta_i(x_i)$ ：由上述 DeepLab 的最大值融合响应给出
- **二元势能** $\theta_{ij}(x_i, x_j)$ ：定义像素 i 和 j 之间的相互作用，旨在通过像素的空间和颜色相似性对分类进行约束。

二元势能的建模 二元势能采用了两种高斯核函数建模像素间关系：

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp \left(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2} \right) + w_2 \exp \left(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2} \right) \right]$$

- 第一项是**双边核 (bilateral kernel)**，依赖于像素位置 p 和颜色 I ，强制空间位置相近且颜色相似的像素拥有相同的标签。
- 第二项是**空间核 (spatial kernel)**，仅基于像素位置，鼓励空间邻近的像素具有相同标签。
- $\mu(x_i, x_j)$ 是 **Potts 模型**，只有当像素 $x_i \neq x_j$ 时， $\mu(x_i, x_j) = 1$ ，以确保标签不一致时施加惩罚。

推理 推理过程采用均值场近似以实现高效的推理，更新算法见 1，更新函数如下：

$$Q_i(Y_i = l) = \frac{1}{Z_i} \exp \left(-\phi_{u,i}(Y_i) - \sum_{l'=1}^L \mu(l, l') \sum_{k=1}^K w_k \sum_{j \neq i} g_k(f_i, f_j) Q_j(l') \right)$$

Algorithm 1 均值场近似

- 1: $Q_i(Y_i) \leftarrow \frac{1}{Z_i} \exp(-\phi_{u,i}(Y_i))$ {初始化}
 - 2: **while** 未收敛 **do**
 - 3: $\hat{Q}_i^k(Y_i = l) \leftarrow \sum_{j \neq i} g_k(f_i, f_j) Q_j(Y_j = l)$ {消息传递}
 - 4: $\tilde{Q}_i(Y_i = l) \leftarrow \sum_{l' \in \mathcal{L}(l, l')} \sum_{k=1}^K w_k \hat{Q}_i^k(l')$ {兼容性变换}
 - 5: $Q_i(Y_i) \leftarrow \exp(-\phi_{u,i} - \tilde{Q}_i(Y_i))$ {局部更新}
 - 6: $Q_i(Y_i) \leftarrow \frac{1}{Z_i} Q_i(Y_i)$ {归一化}
 - 7: **end while**
-

基于贝叶斯优化的参数搜索 由于 CRF 在推理过程中，使用参数高斯核函数来建模不同像素特征间的结构关系，所以高斯核函数中参数的取值对最终的分割效果有着较大的影响。故在本节中，讨论基于贝叶斯优化算法的参数搜索策略，以期望找到相对较优的参数组合，从而提高模型的分割性能

首先对条件随机场模型的参数进行定义：

参数	定义
ITER	迭代数
POS_W	空间核函数权重
POS_XY_STD	空间核函数标准差
BI_W	双边核函数权重
BI_XY_STD	双边核函数位置标准差
BI_RGB_STD	双边核函数颜色标准差

表 1: 条件随机场模型参数定义

搜索策略叙述如下：

1. 使用一组预设的高斯核函数参数作为初始参数，对迭代数进行实验，迭代数的取值范围是从 1 到 10，观察其对分割性能的影响
2. 选择最优迭代数固定其不变，对高斯核函数参数进行实验，观察其对分割性能的影响。高斯核函数参数的搜索空间定义如下：

参数	最小值	最大值
POS_W	1	4
POS_XY_STD	1	10
BI_W	1	10
BI_XY_STD	40	60
BI_RGB_STD	1	5

表 2: 高斯核函数参数搜索空间

具体实验结果可以见 4.2.3

3.2 基于结构支持向量机 (SSVM) 的伪装目标分割

结构支持向量机在引入了结构化损失函数之后，可以被用于解决结构化预测问题，即其输出具有某种特定的结构。在任务二中，我将使用基于结构支持向量机的模型来进行伪装目标分割

3.2.1 模型

从任务一的结果中可以看到，DLabCRF 模型的输出在整体上能够很好的捕捉到伪装目标的整体形状，但是存在以下的问题：

1. **错分类区域：** 即不是伪装目标的区域被分类成伪装目标。这种现象出现的原因是伪装目标与周围环境存在颜色纹理上的高度相似性，导致模型的输出呈现出分割区域分散的问题
2. **分割边界模糊且不平滑：** 这是因为 DLabCRF 底层模型 DeepLab 本身分割能力的限制以及在最后阶段使用双边插值将分割掩码恢复到真值掩码的大小所导致的

故在任务二中尝试使用如图 3 的结构，即使用 DLabSSVM 模型，来尝试解决以上问题，具体细节如下：

第一阶段 该阶段的主要目的是特征提取，使用的仍是 DeepLab 模型，模型初始化为任务一中预训练的权重，且在之后的训练中不对 DeepLab 模型的权重进行进一步调整。

DeepLab 模型的输出经过 softmax 函数变为像素点属于某一类别概率的得分图，其通道数为类别数，在伪装类别分析中类别数是 2。通过将该得分图与图像的颜色特征以及位置特征进行通道维度上的拼接形成新的特征作为第二阶段结构支持向量机的输入，即每一像素点的特征向量是 $(R, G, B, POS_X, POS_Y, S_1, S_2)$

第二阶段 该阶段主要目的是使用结构支持向量机对图像像素间的结构关系进行建模。

采用全连接线性层对结构支持向量机进行设计，模型结构可见图 4。具体来说，结构支持向量机有三个分支，每个分支模型都采用全连接线性层设计：

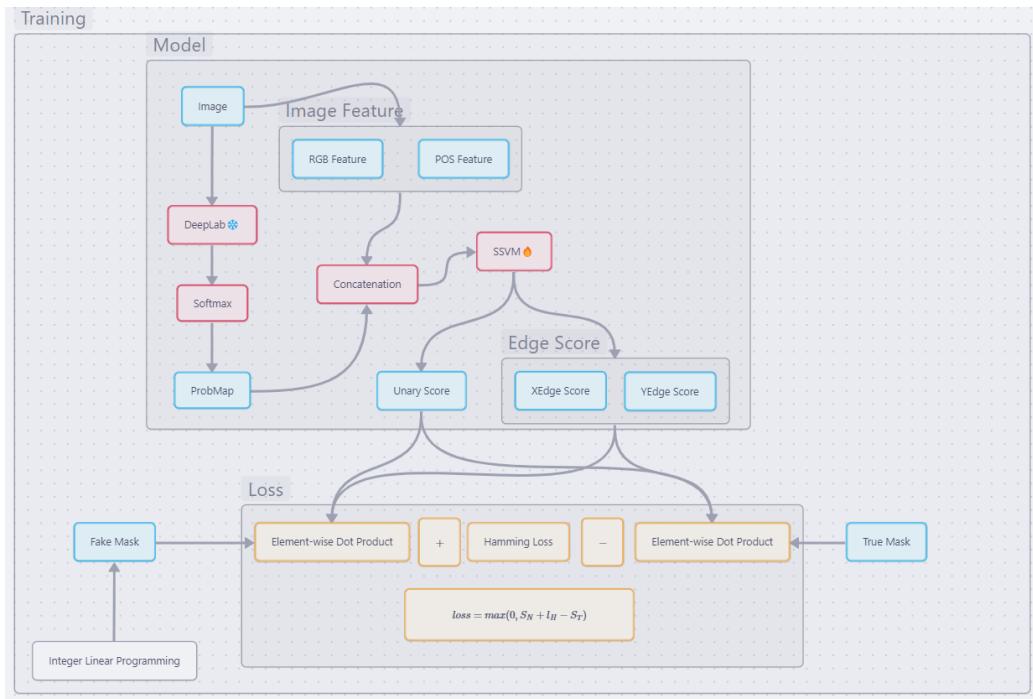


图 3: DLabSSVM 模型结构

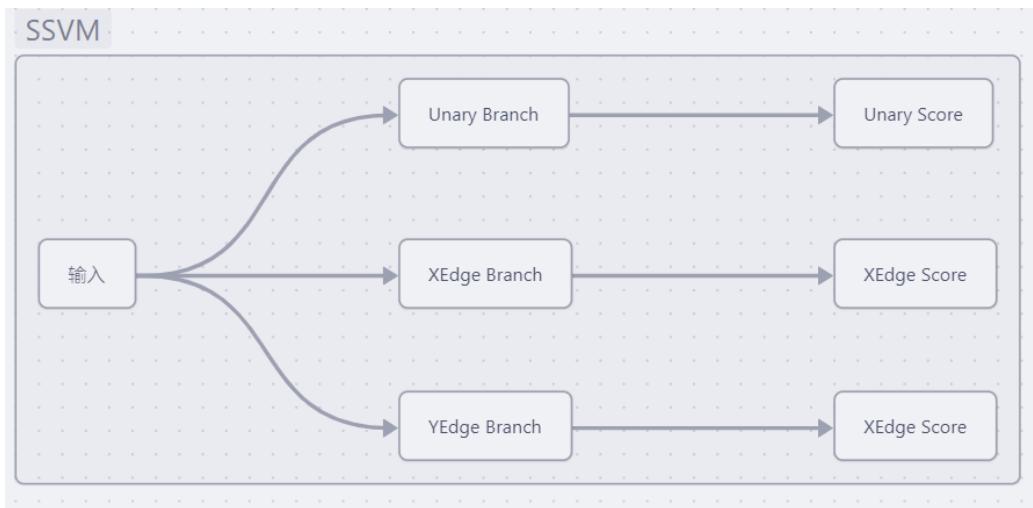


图 4: SSVM 模型结构

1. **Unary Branch :** 该分支负责输出像素点一元得分图，即该分支只考虑像素点自身的特征，输出的得分图中每一像素是该像素点属于每一类别的得分。

2. **XEdge Branch** : 该分支接受当前像素点与其右侧的第一个像素点拼接形成的向量作为输入，输出是在水平 X 方向上像素间的二元得分图
3. **YEdge Branch** : 该分支接受当前像素点与其下侧的第一个像素点拼接形成的向量作为输入，输出是在垂直 Y 方向上像素间的二元得分图

3.2.2 训练

DLabSSVM 模型所要训练的只有结构支持向量机模块。训练流程包括以下两个部分：

1. **伪掩码的产生**：这里的伪掩码指的是得分高且与真值掩码差异较大的分割掩码，该掩码用整数线性规划求解，具体细节可见 3.2.4。
2. **结构化铰链损失 (Structural Hinge Loss)** : 利用如下函数计算伪掩码和真值掩码的得分：

$$Score(\text{Mask}, \text{SMap}) = \sum_i^3 Dot(\text{Mask}_i, \text{SMap}_i)$$

即计算掩码与三个得分图的点积并相加。分别记伪掩码得分为 S_N ，真值掩码得分为 S_T ，代入如下的结构化铰链损失函数：

$$L_{\text{structural}}(x, \hat{y}, y) = \max(0, l_h(y, \hat{y}) - S_T + S_N)$$

其中 l_h 为汉明损失。随后可用得到的损失值进行梯度计算，从而实现模型训练

3.2.3 推理

模型推理流程可见图 5。通过在预测的得分图上进行整数线性规划，从而得到约束条件下的最佳分割掩码。

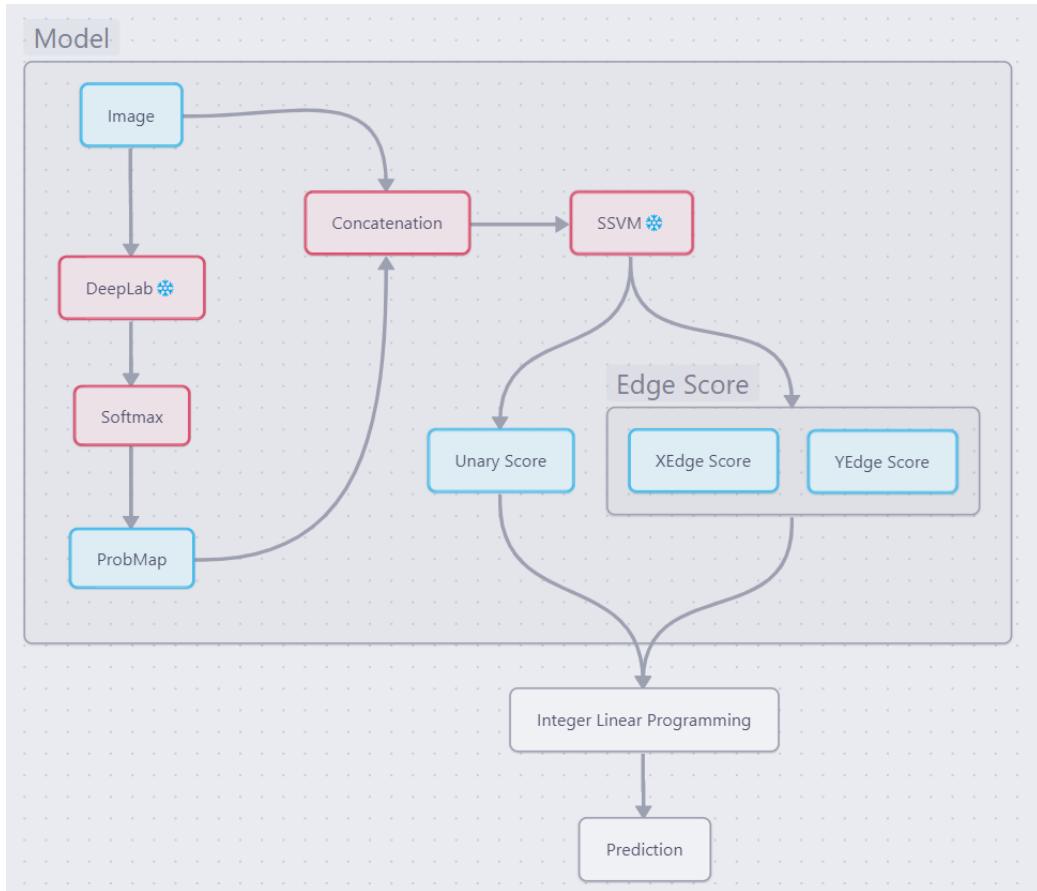


图 5: 模型推理结构

3.2.4 整数线性规划

在 DLabSSVM 中，整数线性规划 (Integer Linear Programming) 被分别用在训练阶段中的伪掩码的产生以及推理阶段预测掩码的产生，两个过程的本质都是求解如下的整数线性规划问题：

$$\begin{cases} x_{ijc} = 0 \text{ or } 1, & i \in \{1, \dots, H\}, j \in \{1, \dots, W\}, c = 0 \text{ or } 1 \\ e_{ijc}^1 = 0 \text{ or } 1, & i \in \{1, \dots, H\}, j \in \{1, \dots, W-1\}, c = 0 \text{ or } 1 \\ e_{ijc}^2 = 0 \text{ or } 1, & i \in \{1, \dots, H-1\}, j \in \{1, \dots, W\}, c = 0 \text{ or } 1 \end{cases}$$

约束条件:

$$\text{s.t. } \begin{cases} \sum_{k=0}^c X_{ijc} = 1 \\ e_{ijc}^1 \leq X_{ijc} \\ e_{ijc}^1 \leq X_{i(j+1)c} \\ e_{ijc}^2 \leq X_{ijc} \\ e_{ijc}^2 \leq X_{(i+1)jc} \end{cases}$$

在训练阶段使用损失增强的最大后验推理 (loss augmented MAP inference)，即最大化如下目标函数:

$$\text{obj} = \sum_{i=1}^3 \text{Dot}(Mask_i, SMap_i) + \sum l(y, X)$$

其中 $Mask_1 = x, Mask_2 = e^1, Mask_3 = e^2$, $l(\cdot)$ 为汉明损失，该目标函数保证所选择的是模型最容易混淆的掩码，即与真值掩码差异最大 (最大化汉明损失)，且得分最高的掩码。

在推理测段只需要保证预测掩码得分最高即可，即最大化如下目标函数:

$$\text{obj} = \sum_{i=1}^3 \text{Dot}(Mask_i, SMap_i)$$

3.3 基于 AdaBoost 的伪装目标检测

AdaBoost 算法是集成学习的一种，其主要思想是通过结合多个“好而不同”的弱学习器来实现一个强学习器。而“好而不同”这个特性的实现则是通过对数据集中的样本分配权重，使弱学习器的每一次迭代学习，所学到的侧重点都不一样，从而弱学习器之间相互“取长补短”，达到强学习器的效果。

在本次任务中，我将伪装目标检测任务分成两个相互联系的模块。

1. **基于 Faster R-CNN 的定位器与样本生成器:** 该模块主要用来实现对伪装目标的定位，以及用于生成训练 AdaBoost 分类器的正负样本特征。

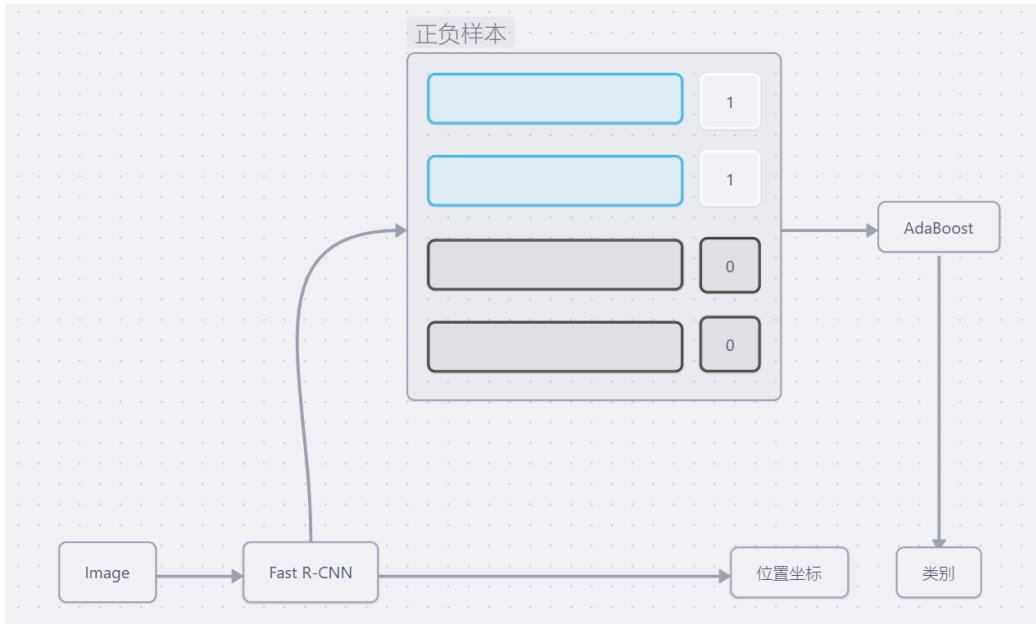


图 6: RegionBoost 模型结构

2. 基于 AdaBoost 的分类器：该模块对 Faster R-CNN 提供的候选框特征进行分类。输出类别 1 对应着伪装目标，输出类别 0 对应着背景。

为了方便起见，称该任务所使用的模型为 **RegionBoost**。接下来将对各个部分进行介绍，模型结构可见 6。

3.3.1 基于 Faster R-CNN 的定位器与样本生成器

首先在伪装目标数据集上对 Faster R-CNN 进行训练，使其能够较好的理解伪装数据集的分布。之后冻结 Faster R-CNN 的权重。Faster R-CNN 在这里充当以下角色：

定位器 在模型推理过程，Faster R-CNN 接收图像作为输入，通过区域提议网络 (RPN) 产生候选框，然后通过 RoI 池化层提取特征用于之后的分类和边界框回归。经过训练的 Faster R-CNN 可以较好的对伪装目标进行边界框定位。

正负样本生成器 在本任务所使用的数据集 (COD10K+CAMO) 中，存在以下问题：

1. **缺少负样本：** 负样本指的是类别为背景的边界框。AdaBoost 分类器的训练需要大量的正负样本来使模型能够在迭代学习中产生对背景和伪装目标的分辨能力，只具有正样本是不够的。
2. **正样本数量较少：** 大部分图片中只包含一个伪装目标，这会造成 AdaBoost 分类器无法充分学习到伪装目标的特征，从而导致分类性能较差

为解决上述问题，RegionBoost 使用 Faster R-CNN 作为正负样本生成器，具体算法如下：

Algorithm 2 生成正负样本的算法

Require: 原始图像及其对应的真值边界框，区域提议网络 (RPN) 输出的候选框集合 $\hat{B} = \{\hat{b}_i\}$

- 1: 从 RPN 中生成的候选框集合 \hat{B} 中获取候选框，其坐标位于原始图像坐标系中。
- 2: **for** 每个真值边界框 b **do**
- 3: 计算 b 与所有候选框 $\hat{b}_i \in \hat{B}$ 的 IoU。
- 4: 将 IoU 大于 0.8 的候选框标记为潜在正样本，类别设为 1。
- 5: 将 IoU 小于 0.5 的候选框标记为潜在负样本，类别设为 0。
- 6: **end for**
- 7: 从潜在正样本和负样本中分别选取预设数量 n_p 和 n_n 个样本，保证正负样本数量比例均衡。
- 8: **for** 每个选定的正样本和负样本 **do**
- 9: 将样本的坐标从原始图像坐标系映射到候选区域坐标系。
- 10: 使用 ROI 池化操作提取样本的特征向量。
- 11: **end for**
- 12: 将生成的样本特征向量及对应类别保存为 “.npz” 格式，以便后续用于 AdaBoost 分类器的训练

3.3.2 基于 AdaBoost 的分类器

AdaBoost 的实现主要在于基学习器的设计。在 RegionBoost 中使用浅层全连接网络作为 AdaBoost 的基学习器，这种设计主要出于以下考虑：

1. **高维输入：** AdaBoost 分类器的输入是从 Faster R-CNN 中提取的经过扁平化处理后的特征图，而特征图的尺寸通常在 $1024 \times 7 \times 7$ ，也就是说每个样本特征的维度是 50176。传统弱学习器如决策树可能会面临计算复杂度过高或者过拟合于某些特征子空间而限制泛化能力的问题
2. **全连接网络强大的泛化能力：** 即使是浅层全连接网络依然可以产生对数据不错的拟合能力和泛化能力

故综上考虑，使用浅层全连接网络作为基学习器。

4 实验

4.1 数据集

CAMO [6] : 该数据集中总共有 1250 张，取该数据集的前 1055 个样本作为训练数据 CAMO_Train，之后的 195 个样本按照 1:1 的比例分为验证集 CAMO_VAL 和测试集 CAMO_TEST

COD10k [3]: 该数据集包含 5066 张伪装目标图片，3000 张背景图片，1934 张非伪装目标图片。类别划分为 5 个超类，78 个子类，并且对于每一张伪装目标图片都标注有关键特征（如多目标、小目标、有遮盖，形状复杂等）。对前 55 个子类样本组成的数据集进行抽样，所形成的新的数据集作为训练数据 COD10K_Train，共 2900 张图片。对后 23 个子类进行同样的处理，所形成的数据集有 1100 图片，按照 1:1 比例分为验证集 COD10K_VAL 和测试集 COD10K_TEST。

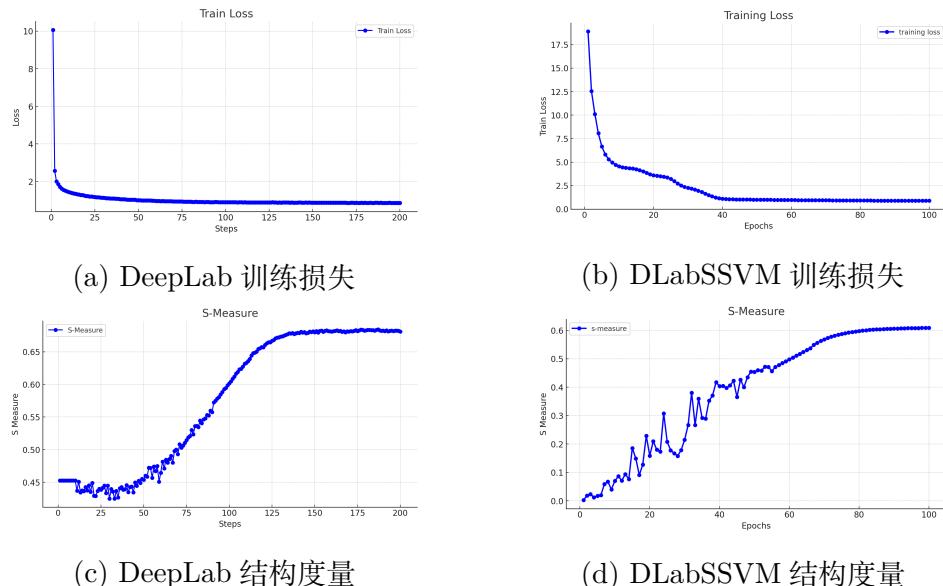
数据划分如下：

划分	数据集
训练集	CAMO_Train + COD10K_Train
验证集	CAMO_VAL + COD10K_VAL
测试集	CAMO_TEST COD10K_TEST

表 3: 数据集划分

4.2 伪装目标分割实验结果分析

4.2.1 模型训练分析



在伪装目标切割任务中，需要训练的模块有两个，一个是用于特征提取的 DeepLab 模型，另一个是 DLabSSVM 中用于结构化预测的 SSVM 模型。现在从训练过程中的指标变化来深入分析这两个模型的表现：

训练损失 如 7a 和 7b 所示，在训练初期，DeepLab 模型的损失值呈现出急剧下降的趋势，尤其是在训练的前几步，损失从接近 10 迅速降到接近 2，但 DLabSSVM 模型相较于 DeepLab 模型下降较缓，在 30 个 epochs 左右降至 5，并在后期趋于平稳。

结构度量 从 7c 和 7d 中可以看出, 与 DLabSSVM 模型的初期结构度量相比, DeepLab 模型在前期的结构度量值较高 (约为 0.45), 这是因为 DeepLab 模型使用了在 COCO 和 PASCAL VOC 数据集上预训练的权重, 导致其在最开始阶段有一个相当不错的准确率, 但 DLabSSVM 中的 SSVM 模块是随机初始化权重, 所以在刚开始训练阶段的结构度量并没有 DLabCRF 出色。随着训练的推进, 两者的结构度量值都在稳步上升, 最终 DeepLab 稳定在 0.68 左右, 而 DLabSSVM 则稳定在 0.61 左右。

4.2.2 模型性能对比

Model	CAMO Test		COD10K Test	
	MAE	S-Measure	MAE	S-Measure
Deeplab	0.134	0.688	0.064	0.693
DLabCRF	0.133	0.685	0.057	0.706
DLabSSVM	0.191	0.607	0.124	0.637

表 4: 伪装目标分割模型性能对比

从在两个测试集上的性能来看, DLabCRF 模型在整体上优于 DeepLab 模型, 尤其是在 COD10K Test 数据集上。这可能是因为 CRF 通过均值场近似算法将 DeepLab 模型预测与像素点的颜色和位置信息进行了整合从而使得原先的离散的分割区域得到了合并, 边界变得更加平滑, 从而提高了分割的准确性。

而 SSVM 模型的性能相对较差, 可能有以下几个原因:

- SSVM 结构过于简单:** SSVM 结构设计采用深度较浅的全连接线性层, 无法有效的捕捉到像素间的结构关系, 存在结构信息丢失的问题
- 推理算法的选择:** 因为 DLabSSVM 使用整数线性规划 (ILP) 来进行推理, 故为了保证训练过程的可行性与高效性, 即在训练时能够在有限时间内找到约束条件下的最具代表性的伪掩码进行指导训练, 我将得分图的分辨率进行降低, 并在最后使用插值的方式将分割掩码恢复到原始分辨率, 这样的处理方式可能会导致分割掩码的边界模糊, 离散伪分割区域较多等问题, 这一点在 4.2.4 中也可以看出来。

3. 像素间结构关系的建模过于简单：我只考虑了像素与其右侧和下侧的第一个像素间的关系，这种建模方式易于实现，但是对于像素间的结构关系捕捉不够充分，导致 DLabSSVM 模型的性能在 DeepLab 模型之下。

4.2.3 不同 CRF 参数下的性能指标对比

Iteration	0	1	2	3	4	5	6	7	8
S-Measure	0.689	0.695	0.693	0.693	0.692	0.691	0.689	0.688	0.685

从上表中可以得到 CRF 均值随机场迭代次数越少，结构度量值 S-Measure 越高。但同时从迭代次数为 0 和非 0 时的对比中，即有无 CRF 后处理逻辑，可以证明 CRF 模块在提升分割精度上的有效性

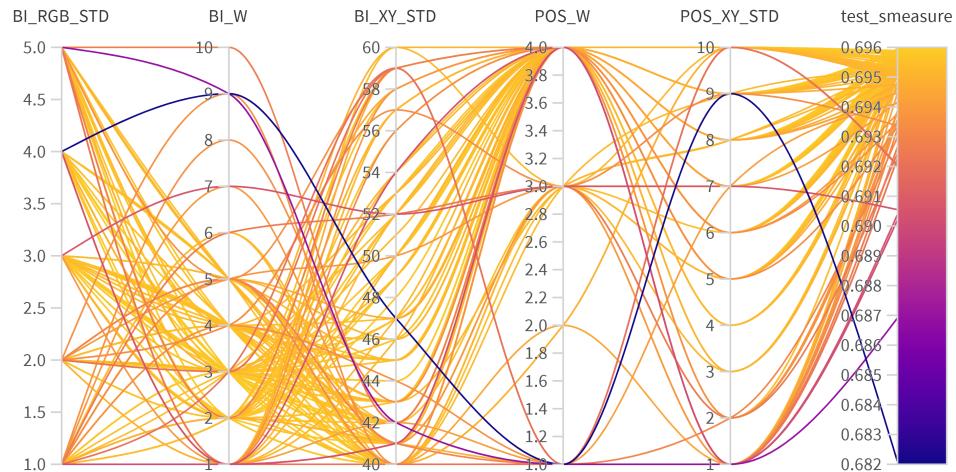


图 8: CRF 参数搜索

固定 CRF 迭代次数为 1，对高斯核函数的参数进行搜索，得到如图 8 所示的平行坐标图。其中当参数取以下值时，结构度量值得到最大 0.696，故选取该组参数作为后续实验的 CRF 参数。

Iter	POS_W	POS_XY_STD	BI_W	BI_XY_STD	BI_RGB_STD
1	4	7	3	40	4

4.2.4 可视化分析

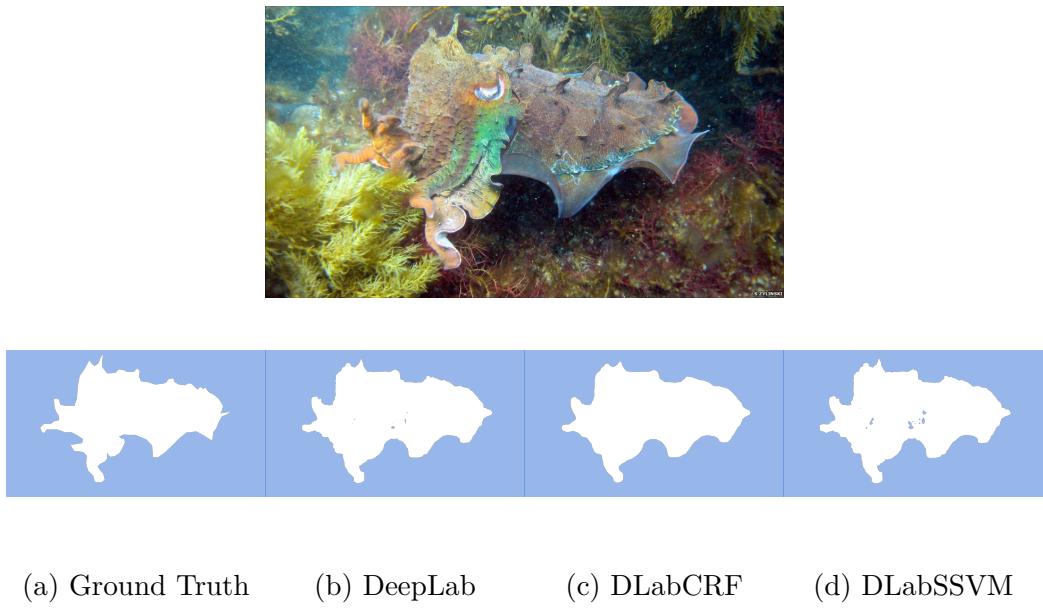


图 9: CAMO 可视化

分别从 CAMO Test 和 COD10K Test 数据集中选取了一张图片进行可视化分析，从中可以看出分割效果最好的是 DLabCRF 模型，分割边界最平滑，分割区域集中，没有出现明显的误分类区域。但其也没有完全能够恢复伪装目标的全部细节，比如图 10 中的猫头鹰的耳朵部分就没有被分割出来。

DeepLab 模型与 DLabSSVM 模型在可视化上可以看出有明显的误分割区域，但 DeepLab 模型的分割边界更加平滑，DLabSSVM 模型的分割边界更加锯齿化，且 DLabSSVM 的误分割区域更多。

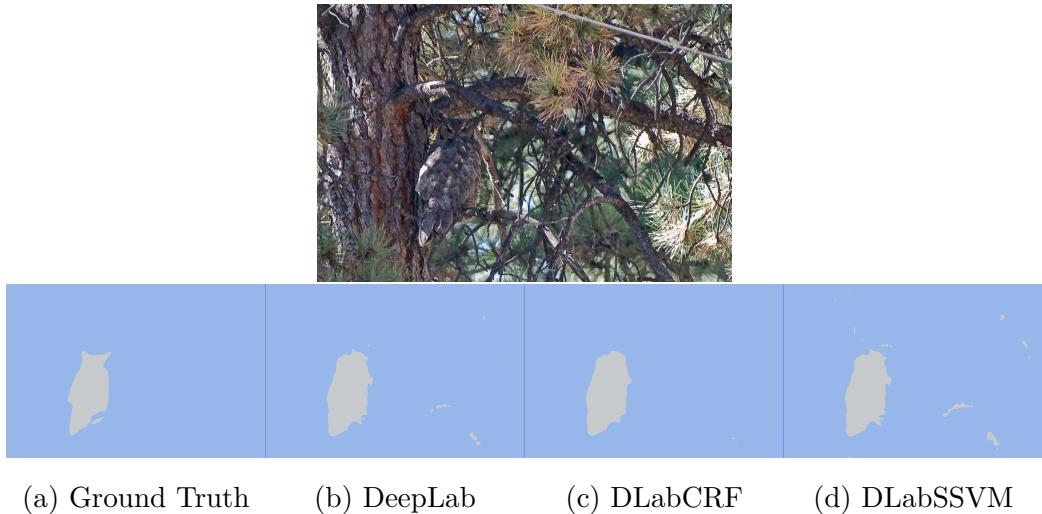


图 10: COD10K 可视化

5 代码

以下展示部分关键代码，完整代码已经放在 github 仓库

```

1
2     class CamouflageDataset(_BaseDataset):
3         def __init__(self, split_ratio, dataset="none", **kargs):
4             self.split_ratio = split_ratio
5
6             assert dataset in ["COD10K", "CAMO", "CAMO+
7                 COD10K", "COD10K+CAMO"], "Dataset not found"
8             self.dataset = dataset
9             self.image_list = []
10            self.mask_list = []
11            super().__init__(**kargs)
12
13        def _load_data(self, index):
14            # Image preprocessing:
```

```
14     image = np.asarray(Image.open(self.image_list[
15         index]))
16
17     mask = np.asarray(Image.open(self.mask_list[
18         index]))
19
20     print("====")
21     print(f"Processing image: {self.image_list[
22         index]}")
23     # print(f"Processing mask: {self.mask_list[
24         index]}")
25
26     return image, mask
27
28
29 def _set_files(self):
30     self.split_dir = os.path.join(self.root, self.
31         split)
32
33     imageP = Path(os.path.join(self.split_dir, "image"))
34     maskP = Path(os.path.join(self.split_dir, "GT"))
35
36     if self.split == "test":
37         for d in self.dataset.split("+"):
38             selected_images, selected_masks = self.
39                 .split_dataset(imageP, maskP, "camouflage" if d == "CAMO" else d)
40             self.image_list.extend(selected_images)
41             self.mask_list.extend(selected_masks)
42
43     else:
```

```
35     self.image_list = sorted([img for img in
36                               imageP.iterdir()])
37     self.mask_list = sorted([mask for mask in
38                               maskP.iterdir()])
39
40     return
```

```
1
2     class DenseCRF(object):
3         def __init__(self, iter_max, pos_w, pos_xy_std,
4                      bi_w, bi_xy_std, bi_rgb_std):
5             self.iter_max = iter_max
6             self.pos_w = pos_w
7             self.pos_xy_std = pos_xy_std
8             self.bi_w = bi_w
9             self.bi_xy_std = bi_xy_std
10            self.bi_rgb_std = bi_rgb_std
11
12        def __call__(self, image, probmap):
13            C, H, W = probmap.shape
14
15            U = utils.unary_from_softmax(probmap)
16            U = np.ascontiguousarray(U)
17
18            image = np.ascontiguousarray(image)
19
20            d = dcrf.DenseCRF2D(W, H, C)
21            d.setUnaryEnergy(U)
22            d.addPairwiseGaussian(sxy=self.pos_xy_std,
23                                  compat=self.pos_w)
24            d.addPairwiseBilateral(
25                sxy=self.bi_xy_std, srgb=self.bi_rgb_std,
```

```
        rgbim=image, compat=self.bi_w
    )

25
26     Q = d.inference(self.iter_max)
27     Q = np.array(Q).reshape((C, H, W))

28
29     return Q
```

```
1
2     class SegmentationMetric:
3         """
4
5         Metric类用来跟踪MAE与S-Measure指标
6
7         要度量的性能指标如下：
8         1. S-Measure: "Structure-measure: A New Way to
9             Evaluate Foreground Maps" 论文中提出
10
11         2. Mean Absolute Error(MAE)
12         """
13
14
15     @staticmethod
16     def get_mae(mae_list, reduce="average"):
17         """
18
19         根据reduce返回平均绝对误差的均值、加和或者MAE
20         列表
21         """
22
23         assert reduce in ["average", "sum", "none"], f
24             "Got unexpected reduce method:{reduce}"
25         mae = torch.stack(mae_list, 0)
26
27
28         if reduce == "average":
29             return mae.mean()
30         elif reduce == "sum":
```

```
22         return mae.sum()
23     else:
24         return mae
25
26     @staticmethod
27     def get_smeasure(smeasure_list, reduce="average"):
28         """
29             根据reduce返回结构度量的均值、加和或者S-
30             Measure列表
31         """
32
33         assert reduce in ["average", "sum", "none"], f
34             "Got unexpected reduce method: {reduce}"
35         sm = torch.stack(smeasure_list, 0)
36
37         if reduce == "average":
38             return sm.mean()
39         elif reduce == "sum":
40             return sm.sum()
41         else:
42             return sm
43
44     @staticmethod
45     def calculate_mae(pred, gt):
46         if pred.ndim == 3:
47             assert pred.shape[0] == 1, "只能是单张图像"
48             pred = pred.squeeze(0)
49         if gt.ndim == 3:
50             assert gt.shape[0] == 1, "只能是单张图像"
51             gt = gt.squeeze(0)
52         assert gt.shape == pred.shape, "预测图像与真值"
```

```
50
51     return torch.abs(pred - gt).mean()
52
53     @staticmethod
54     def calculate_smeasure(pred, gt, alpha=0.5):
55         assert gt.shape[1:] == pred.shape[1:], "预测图
56             像与真值图像的尺寸必须一致"
57
58         if pred.ndim == 3:
59             assert pred.shape[0] == 1, "只能是单张图像
60                 "
61
62             pred = pred.squeeze(0)
63
64         if gt.ndim == 3:
65             assert gt.shape[0] == 1, "只能是单张图像"
66
67             gt = gt.squeeze(0)
68
69
70         y = gt.mean()
71         if y == 0:
72             x = pred.mean()
73             Q = 1.0 - x
74
75         elif y == 1:
76             x = pred.mean()
77             Q = x
78
79         else:
80             gt[gt>=0.5] = 1
81             gt[gt<0.5] = 0
82             Q = alpha * SegmentationMetric._S_object(
83                 pred, gt) + (1-alpha) *
84                 SegmentationMetric._S_region(pred, gt)
85
86         if Q.item() < 0:
```

```
76         Q = torch.FloatTensor([0.0])
77     if isinstance(Q, float):
78         raise ValueError("Q is a float")
79     return Q
```

```
1
2     class DLabSSVM(nn.Module):
3
4         def __init__(self, num_classes, num_features,
5             num_iterations=8):
6
7             super(DLabSSVM, self).__init__()
8
9             self.num_classes = num_classes
10            self.num_features = num_features
11            self.num_iterations = num_iterations
12
13
14            self.ssvm = SSVM(num_classes, num_features)
15
16
17            def forward(self, x, y):
18
19                """
20
21                Args:
22                    x: torch.Tensor, shape=(N, C, H, W)
23                    y: torch.Tensor, shape=(N, H, W)
24
25                """
26
27                N, C, H, W = x.shape
28                x = x.view(N, C, -1).permute(0, 2, 1).
29                    contiguous()
30                y = y.view(N, -1)
31
32
33                # 训练SSVM
34                self.ssvm.train(x, y, self.num_iterations)
35
36
37                # 推理
38                y_pred = self.ssvm.inference(x)
```

```
26
27     # CRF后处理
28     y_pred = y_pred.view(N, H, W)
29     x = x.permute(0, 2, 1).view(N, H, W, C).
30         permute(0, 3, 1, 2)
31     y_pred = self.crf(x, y_pred)
32
33
34     return y_pred
```

```
1
2     class RGBandPOSFeature(nn.Module):
3         def __init__(self):
4             super().__init__()
5
6         def forward(self, x):
7             """
8                 Expect x to be an image of size (bs, 3, height
9                     , width)
10                Output should be the feature vector of image x
11                    with size (bs, f, height, width)
12
13                feature_vector = None
14                cf = self.color_feature(x)
15                pf= self.position_feature(x).unsqueeze(0)
16                feature_vector = torch.concat([cf, pf], dim=1)
17
18
19                return feature_vector
20
21
22                def color_feature(self, x):
23                    return x
24                def position_feature(self, img):
25                    _, _, height, width = img.shape
```

```
23     xs = torch.arange(0, width)
24     ys = torch.arange(0, height)
25     return torch.stack(torch.meshgrid([ys, xs],
26                               indexing="ij"))
```

参考文献

- [1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin P. Murphy, and Alan Loddon Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:834–848, 2016.
- [2] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.
- [3] Deng-Ping Fan, Ge-Peng Ji, Guolei Sun, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Camouflaged object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2774–2784, 2020.
- [4] Ross B. Girshick. Fast r-cnn. 2015.
- [5] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2013.
- [6] Trung-Nghia Le, Tam V. Nguyen, Zhongliang Nie, Minh-Triet Tran, and Akihiro Sugimoto. Anabanch network for camouflaged object segmentation. *Comput. Vis. Image Underst.*, 184:45–56, 2019.

- [7] Laurent Najman and Michel Schmitt. Watershed of a continuous function. *Signal Process.*, 38:99–112, 1994.
- [8] Richard Nock and Frank Nielsen. Statistical region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1452–1458, 2004.
- [9] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.*, 9:62–66, 1979.
- [10] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.
- [11] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2014.