### 1课题介绍

#### 1.1 背景简介

在北京旷视科技有限公司实习期间，根据公司的安排，笔者参加NeurIPS2022 Workshop 的 Visual Domain Adaptation Challenge 比赛。

按照比赛的要求，笔者从 0 到 1 搭建了一个深度学习项目，该项目使用语义分割的方法进行垃圾分类。本报告中，将对该项目的相关算法进行详细的阐述。

#### 1.2 任务介绍

该比赛要求使用语义分割的方法进行垃圾分类，即判断每个像素属于背景还是 4 种垃圾中的一种（分别对应序号 0，1-4）。同时，该比赛要求选手提出适当的无监督域适应策略。下面进行具体阐述。

在使用深度学习语义分割进行工业废物分类时，我们无法收集到完全代表期望数据分布的标准数据集，这是由于工业废物流的视觉外观和成分随时间、位置、季节、使用的机器而不断变化。这些因素都会导致训练数据的分布和检测时数据的分布之间存在一定的 gap，即域迁移。针对这种域迁移，我们需要提出合适的域适应策略，以使模型能够克服训练数据和测试数据之间的 gap，从而得到更好的测试性能。

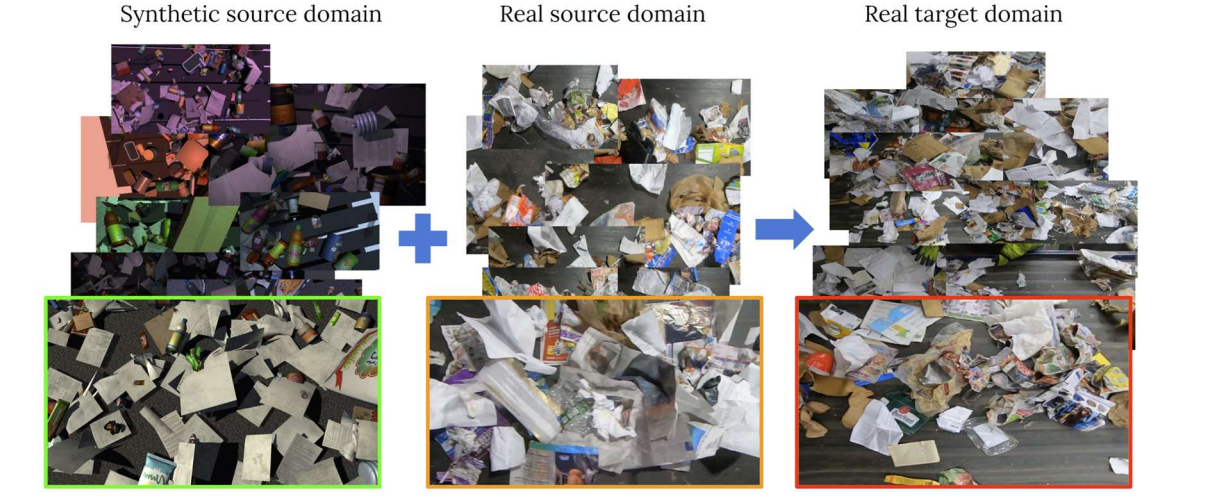
在实际的应用场景中，我们在训练时使用的数据是有标注的，他们来自某种数据分布，记为 A 分布；测试时的数据是无标注的，他们来自另一种数据分布，记为 B 分布。A 分布与 B 分布之间有一定的差异，我们的目标就是提出合适的域适应策略，使得模型能够在 B 分布的数据上取得较好的测试性能。由于测试数据是无标注的，因此我们需要一种无监督/自监督的域适应方法。

#### 1.3 需求分析与问题建模

如前所述，我们有来自 source domain 的带有逐像素标注的训练数据，和来自target domain 的无标注的测试数据。我们的需求是，通过某种无监督或自监督的域适应策略，合理使用 source domain 和 target domain 的数据对模型进行训练，从而使得模型能够在测试数据上取得尽量优秀的性能。

在本次比赛中，由于采集大量真实分布的工业废弃物图片并进行逐像素标注是一件很困难的事，主办方将 source domain 的训练数据划分为两部分：一部分是带有标注的真实照片，由于标注需要花费较多人力，这部分数据比较少；另一部分是使用 unity 生成的仿真工业废弃物图片，这部分数据可以直接生成逐像素标注，无需人工标注，因而数量很大。

综上，本次比赛的任务是，使用深度学习语义分割进行工业垃圾分类，从仿真数据和部分真实数据构成的 source domain 较好地迁移到测试时的真实数据target domain 上。如下图所示：



需要特别说明的是，source domain 中的真实照片与 target domain 的真实照片拍摄于不同地点不同时间，因此二者之间也存在 domain gap，而不是独立同分布的。

#### 1.4 数据整理

实际数据集构成如下图所示：



其中：

zerowaste-f 为真实数据，有标注，但数量较少；

synthwaste\_splits 为仿真数据，有标注，数量充足；

synthwaste\_aug 为加入较强数据增强的仿真数据，有标注，数量充足；

zerowaste-v2-trainval 为真实数据，无标注。

本次比赛的任务是，合理使用这 4 部分数据训练模型，以使模型zerowaste-v2 数据集上的测试性能最优。测试时，使用各类别平均交并比 mIoU作为评价指标。

### 2项目框架

本次项目相对比较复杂，笔者在项目初期进行了较多的思考和规划，力求理清逻辑框架、提升可扩展性。下面，笔者将首先从模型和训练流程两个方面简要 介绍一下本项目的基本架构。

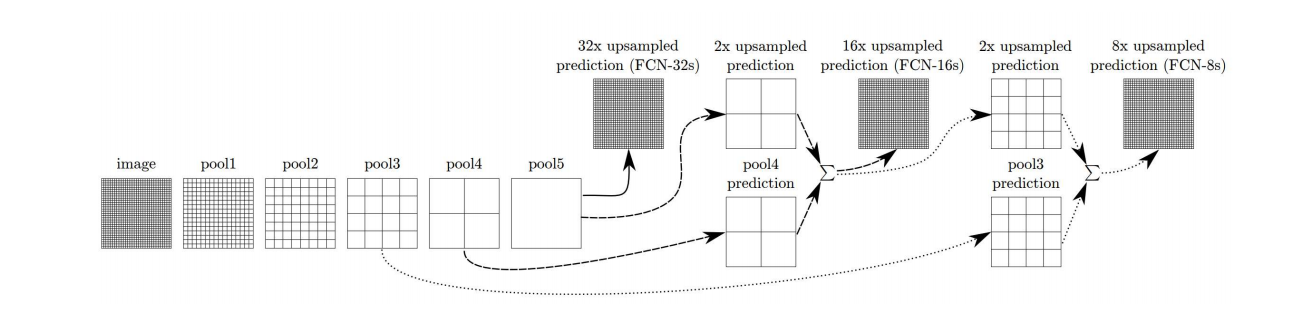
#### 2.1 模型

在项目中，笔者将多种模型统一封装为 core\_model 类，只需要调用 get()函数就可以得到对应类别的模型。目前，项目中可以使用的有 3 个模型，分别是FCN、DAFormer、CLIPSeg。下面依次进行介绍。

##### 2.1.1 FCN

Fully Convolutional Networks[1]，经典的分割任务模型，采用encoder-decoder 结构——先对输入图片进行多次卷积，下采样提取语义特征，再对特征进行多次反卷积，上采样恢复到与输入相同尺寸，从而得到逐像素预测结果。同时，模型中还有一些跳边，直接连接 encoder 和 decoder 中一些特征尺寸相同的位置。

FCN-32s、FCN-16s、FCN-8s 的模型示意图如下：

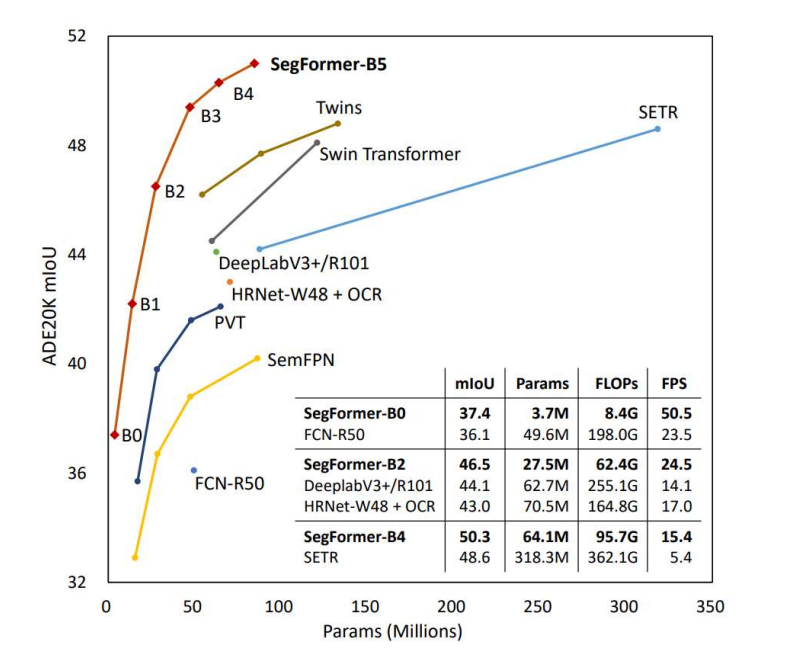


在本项目中，笔者使用的模型是在 FCN-8s 模型基础上加入更多跳边得到的，即将上图中 pool1 和 pool2 的特征也通过跳边引至 decoder 中对应位置。FCN 模型的参数量很少，其权重文件只约 70MB，是一个很轻量级的基础模型。本项目中引入 FCN 也正是这个原因，FCN 很方便本地调试，同时也节省服务器的训练资源和训练时间，能做一个不错的 baseline。

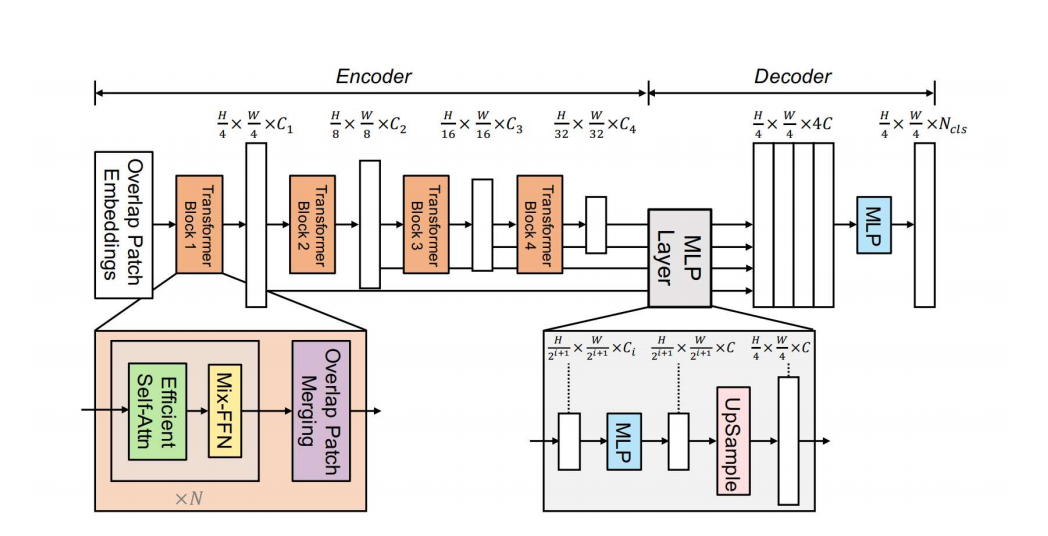
##### 2.1.2 DAFormer

DAFormer[2]是本次比赛官方给出的 baseline，它是一篇聚焦域适应语义分割的工作，文中提出了一种不错的 unsupervised domain adaptation(UDA)策略，其模型结构使用的是 SegFormer。

SegFormer[3]是一篇专注于使用 ViT 模型进行语义分割的工作，文中提出了SegFormer模型，它将transformer与轻量级多层感知器(MLP)解码器结合起来，是一个简单、高效但强大的语义分割框架。该模型在诸多数据集上都表现出了相当强大的性能，不仅参数量小、效率高，而且 mIoU 高于其他诸多模型。该模型与其他模型性能比较图如下：



SegFormer 的模型结构如下：



SegFormer 是官方的 baseline 模型，笔者将其引入项目，以便与官方的结果进行比较、排查项目中一些可能的问题。

##### 2.1.3 CLIPSeg

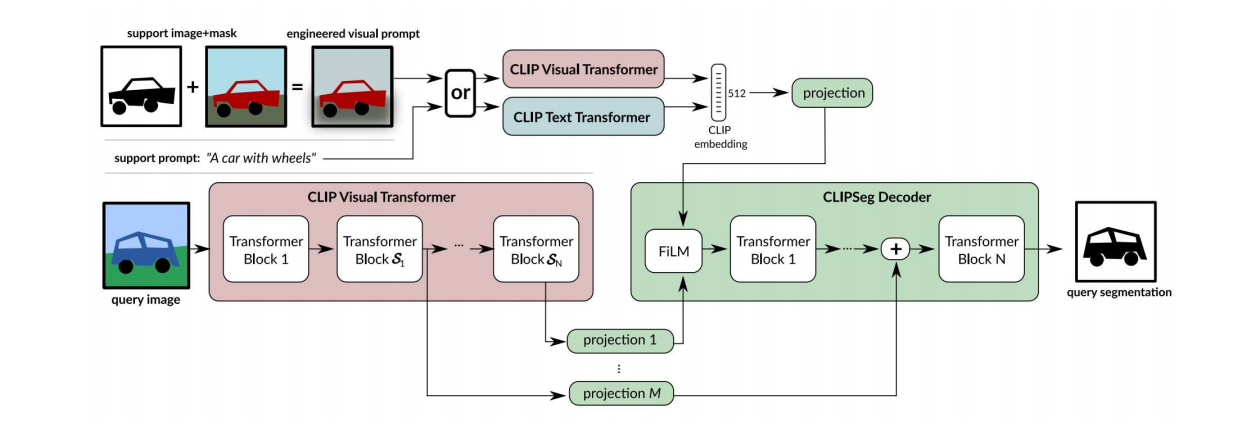
CLIP（Contrastive Language-Image Pre-training）[4]是 OpenAI 团队在多模态方向的一篇工作，也是多模态方向的一篇里程碑式的工作，对多模态领域的发展起到了强大的推进作用。CLIP 在诸多下游任务上都展现出了极其强大的性能，尤其是在 zero shot、few shot 领域。

CLIPSeg 是一篇将 CLIP 模型用于语义分割任务的工作。CLIPSeg 完全没有修改CLIP原本的模型结构，仅从CLIP的Image Encoder中取出几个activations，并额外添加了一个十分简单的语义分割结构，就能够完成逐像素的语义分割任务。在 PhraseCut 数据集上进行训练后，任意给出一张图片和一个物体名称，该模型可以给出一个二值分割图，表示每个像素是否为该物体。

站在 CLIP 的“肩膀”上，利用 CLIP 模型 Image Encoder 和 Text Encoder 对齐的语义空间，该模型能够将 referring expression segmentation、zero-shot segmentation 和 one-shot segmentation 这三种分割任务统一在一个 CLIPSeg 模型之中。本项目中，我们仅仅使用文本来指明要分割的物体，故此处不再展开。

在本项目中，笔者考虑使用 CLIPSeg 模型是因为 CLIP 确实具有十分出色的泛化能力，这对于模型克服 source domain 和 target domain 数据的差异有很大帮助。当然，CLIPSeg 的 zero shot 能力不足以应对本次比赛，我们还需要重新对模型的 decoder 进行训练，并对 CLIP 部分进行 finetune。

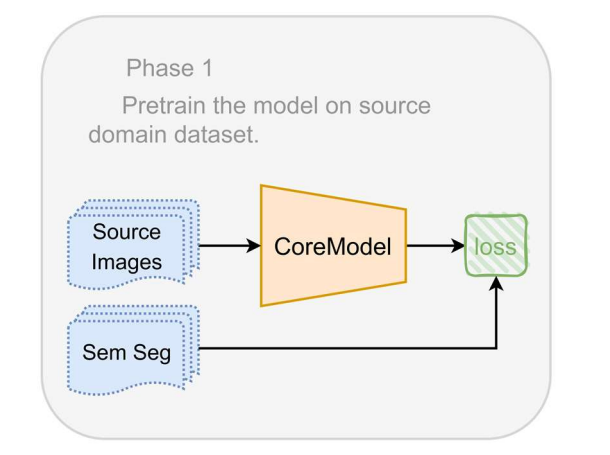
CLIPSeg 模型的结构如下图所示。可以看到，其思想十分简单，只是在 CLIP的基础上添加了一个很轻量级的 Decoder。Decoder 部分权重保存下来只有不到10MB。



#### 2.2 训练流程

##### 2.2.1 预训练

如前文所述，在本次比赛中，我们有两部分数据，一部分是来自 source domain 带有标注的数据，另一部分是来自 target domain 的无标注数据。在不考虑 UDA(unsupervised domain adaptation)策略的情况下，我们会使用 source domain 的数据对模型进行监督训练，然后依靠模型自身的泛化能力，直接得到 target domain 上的测试性能。



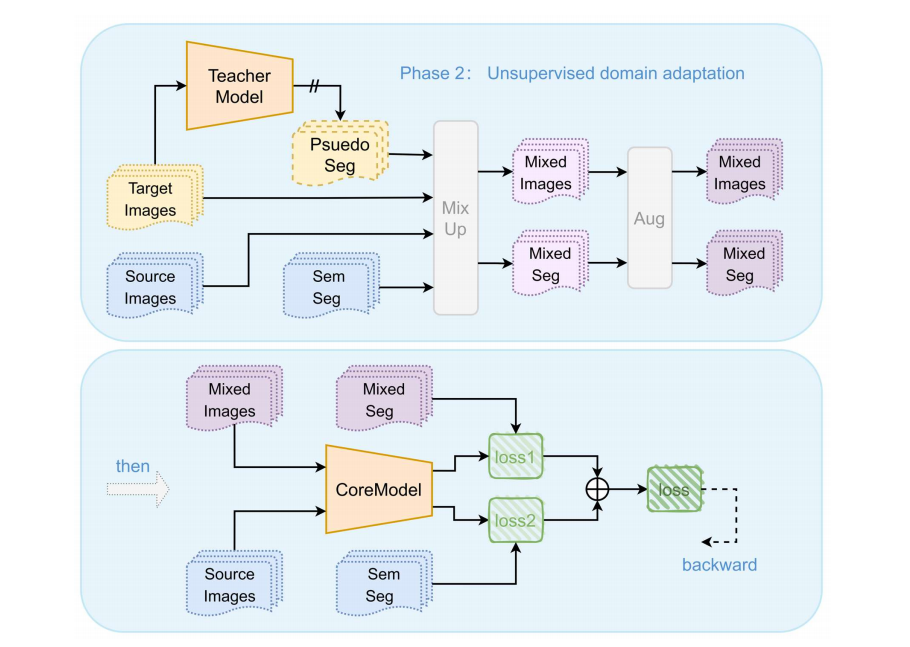
如上图所示，其中损失函数使用BCEWithLogitsLoss。

##### 2.2.2 UDA 训练

仅仅依靠模型泛化能够在 target domain 上取得不错的测试性能，但是并不能让我们满意。此时，我们在训练阶段完全没有使用到 target domain 的数据，这是一种缺失。因此，我们可以提出有效的 UDA 策略，更好地利用 target domain的无标注数据，提升模型在 target domain 上的性能。

因此，模型在训练过程中很自然地分为两个阶段：一个是前文中给出的在source domain 上进行的有监督预训练，另一个是同时使用 source 和 target 的数据，进行无监督域适应训练。

本项目中，笔者使用的 UDA 策略流程图如下。后文中将详细介绍，此处暂不再展开。



### 3基础方法

#### 3.1 改变数据尺寸

#### 3.1.1 Crop

在本项目中，DAFormer 模型和 CLIPSeg 模型均要求输入数据的长宽相等，而实际数据集中的图片的尺寸为1920\*1080，因此输入模型之前必须进行裁剪。

1. Random Crop

随机地从图片中某个可行位置裁剪出一个指定尺寸的块，同时取出这一块图片对应的标注。通常用于训练时的数据处理。

1. Cover Crop

Random Crop 解决了原始数据尺寸和模型输入尺寸要求不匹配的问题，但是，在验证或测试阶段，我们想要得到模型对整张图片所有像素的类别预测，而不是仅仅拿出图片中的某一个区域做预测。

Cover Crop 会将一张图片分割为若干相同大小的 patch，可能会有重叠，但是能够保证覆盖原图。

例如，原始图片 h 为 1080，现在要求分割的 patch 的 h 为 224，则会分割

(1080/224)向上取证，即 5 块。

进而会采取这样的策略进行分割：

第 1 块的 h 范围：0\*224 -- 1\*224-1

第 2 块的 h 范围：1\*224 -- 2\*224-1

第 3 块的 h 范围：2\*224 -- 3\*224-1

第 4 块的 h 范围：3\*224 -- 4\*224-1

第 5 块的 h 范围：1080-224 -- 1080-1

可以看到，第 4 块与第 5 块有重叠，但是能够保证这 5 块小图片完整覆盖了原始图片的 h 方向。

同时，笔者还编写了 recover\_from\_crop 函数，与 Cover Crop 的逻辑完全相同，但是是其逆过程，会将若干小块重新拼接成一个完整的图片。主要用于将个小块的预测结果拼接成全图的预测结果。

##### 3.1.2 Resize

(1) Resize

如前所述，原始图片的尺寸是 1920\*1080，而模型的输入通常是 224\*22或更大一点，二者之间存在较大差异。如果直接从原始图片中 crop 出 224\*224大小的块，则每个小块的面积只占原图的 2.42%，过于局部化，丢失了很多全局的信息，对分割任务很不利。

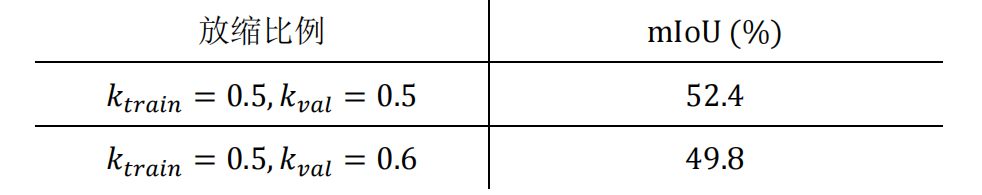
因此，在实际训练过程中，我们首先需要同步改变图片和标签的尺寸，然后再进行 crop。例如，先将图片放缩为原始尺寸的 0.5 倍，再 crop。

(2) Random Resize

在实验过程中，笔者发现，模型对于图片中物体的大小是敏感的。例如，在训练 CLIPSeg 模型时，笔者首先将图片放缩为原始尺寸的 0.5 倍，再 crop 出352\*352 大小的块，用这样的数据对模型进行训练。

验证阶段，我们可以仍然将图片放缩为 0.5 倍，再进行 crop；也可以不放缩为 0.5 倍，而是放缩为其他倍率，如 0.6 倍，再 crop 出 352\*352 大小的块。也就是说，验证时图片的放缩倍率不一定要和训练时的放缩倍率一致，放缩倍率的改变会导致 crop 出的小块中物体大小的改变。例如，放缩为 0.5 倍后 crop 出的物体比放缩为 0.6 倍后 crop 出的物体大一些。

笔者做实验验证了放缩倍率的改变对验证集上模型性能的影响：



可以看到，放缩比例的改变会明显影响模型性能，模型对于物体大小的改变缺乏鲁棒性。在一定程度上，这种鲁棒性是必要的，因为测试集中的物体大小与训练集有一定的差异。如果模型无法很好地应对物体大小的改变，则难以在测试

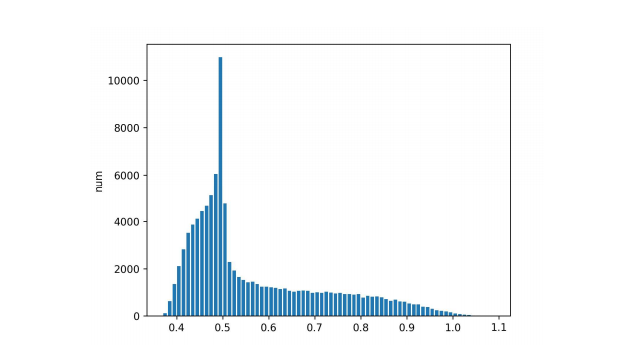
集上取得较好的性能。

为此，笔者引入了 Random Resize。笔者由标准正态分布变换出了一个分布，其特点是：

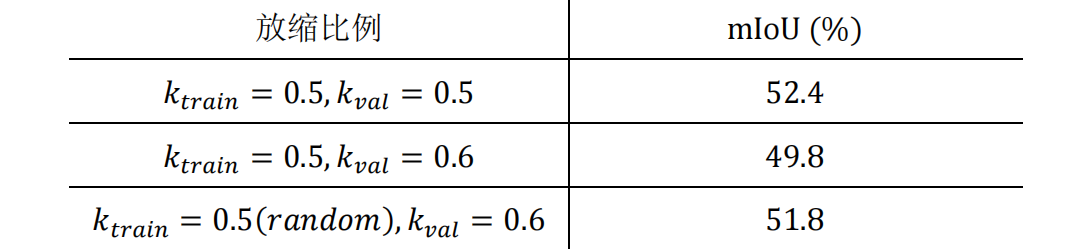
(1)有严格的数据上下限；

(2) 能够设定一个值，最终生成的数据有 50%的概率小于设定值，有 50%概率大于设定值。

设定放缩比例为 0.5 时，实际放缩比例采样 10,0000 次，得到如下频数分布图：



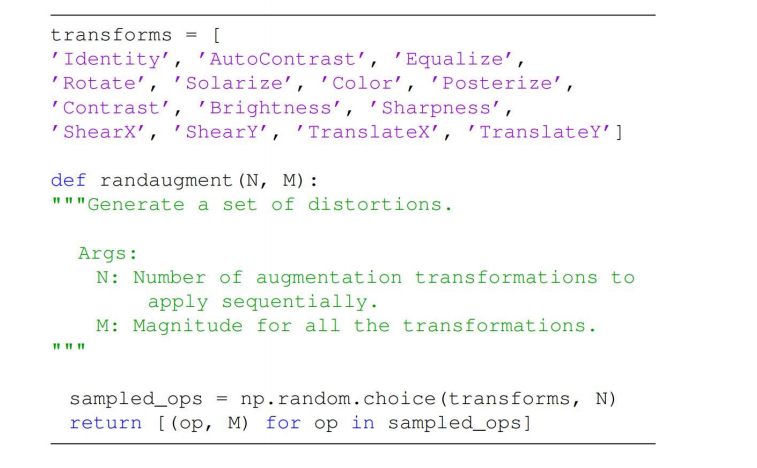
每次调用 Random Resize 时，会从该分布采样一个实际放缩系数对图片进行放缩。经过实验验证，在训练阶段加入 Random Resize 策略后，在验证集上模型对于放缩倍率改变的鲁棒性增强：



#### 3.2 数据增强

本项目中，笔者参考 Randaugment 进行了数据增强。

Randaugment[5]。是数据增强方面一篇很有影响力的工作，文中提出 了14 种数据增强的操作，并设定了两个超参数 N 和 M。其中，N 表示对于每张图片应用的增强操作数量，取值范围为 0~14；M 表示每个操作的强度，取值范围为 0~30。作者在文中对于 Randaugment 给出了一段 python 代码解释，如下图：



与分类任务不同，语义分割任务是逐像素的。在做数据增强时，有一些数据增强方法不会导致标签发生变化，有一些增强方法则需要标签同步变化。因此，在本项目中，笔者基于 RandAugment 类似的思想，针对这两种情况分别引入了RandAugment 和 SegAugment。

##### 3.2.1 不改变标签的数据增强

在本项目中，笔者将这种情况实现为 RandAugment 类。

包含以下数据增强操作：

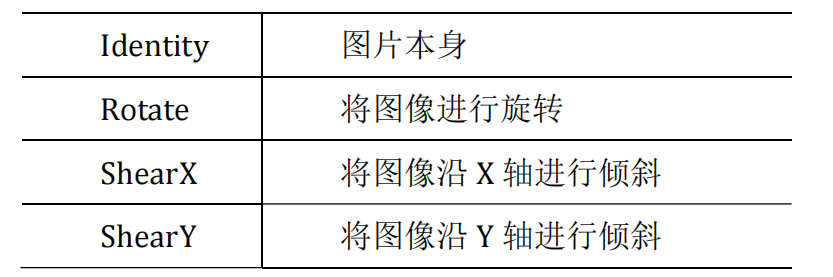


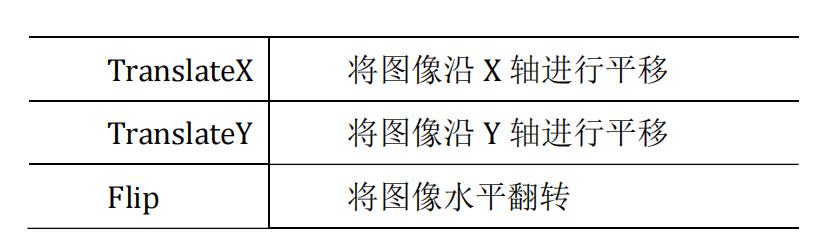
需要说明的是，原始的那篇 RandAugment 工作中，选取的 N 种增强操作可以重复，即 N 种操作里可能多次出现同一种操作，如亮度调整，最终可能导致图片亮度变化太过于极端，如接近全白或全黑，使得数据受损。因此，笔者修改了操作选取策略，保证每一次选取的 N 种增强操作无重复。

##### 3.2.2 改变标签的数据增强

在本项目中，笔者将这种情况实现为 SegAugment 类。

包含以下数据增强操作：





与之前同理，这里每一次选取的 N 种增强操作无重复。

#### 3.3 Mix Up

Mix Up 是一种非常规的数据增强方法。将两个不同数据以某种方式进行混合，得到新的混合数据，从而达到极大扩充数据分布的效果。与分类任务不同，语义分割任务中，Mix Up 需要同步处理图片和标签，以保证混合后图片和标签之间依然正确对应。

在本项目中，笔者共实现了 4 种不同的 Mix Up 模式，下面依次进行介绍。

#### 3.3.1 Hard

硬性混合。从 A 图片中某个位置取出指定大小的块，然后将这块图片直接覆盖到 B 图片的某个随机可行位置，从而得到混合图片 M。同时需要对标签做相同处理，以保证图片和标签之间的对应性。

#### 3.3.2 Soft

柔性混合。从 A 图片中某个位置取出指定大小的块，然后在 B 图片中某个随机可行位置选取相同大小的块将P\_B替换为aP\_A+(1-a)P\_B，从而得到混合图片 M。其中，a是 从0-1均匀分布中采样出的一个随机数。同时需要对标签做相同处理，以保证图片和标签之间的对应性，此时的标签不再是 one hot label。

#### 3.3.3 Object

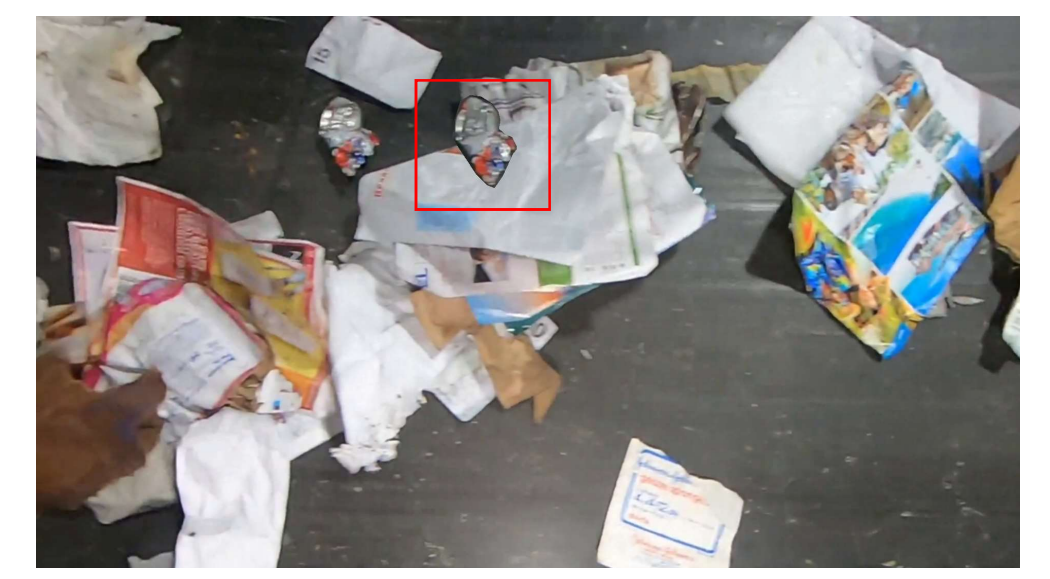
物体混合。从 A 图片中某个位置选择一个指定大小的区域，根据 A 图片的逐像素标签，我们可以取出该区域中的某 N 种物体，然后将这 N 种物体覆盖到B 图片的某个随机可行位置，从而得到混合图片 M。同时需要对标签做相同处理，以保证图片和标签之间的对应性。

在算法实现时，笔者设置了各个类别物体被选取的概率。例如，我们可以设置每种类别物体被选择的概率为：



则在从 A 图片中选取物体时，5 种类别的物体被选中的相对概率比例是[0,10, 3, 2, 2]。例如，我们设置每次从 A 图片中选取 1 种物体放入 B，并且此时A图片待选区域中包含第 0、1、2、4 类物体，这四种物体对应的选取概率比例为0、10、3、2，则实际选择每一种物体的概率为 0.0，0.67，0.2，0.13。设置每次从 A 图片中选取 2 个或多个物体放入 B 时，算法依然能保持这种预先设置的选取概率比例。

一张 Object Mix Up 示例图片如下，可以看到，红框区域的那个易拉罐是被混合进来的。



#### 3.3.4 Replace

替换混合。这种混合模式与物体混合对偶，从 A 图片中某个位置选择一个指定大小的区域，根据 A 图片的逐像素标签，我们可以取出该区域中的某 N 种物体，然后将这 N 种物体对应的像素删除，并从 B 图片中某个可行位置取出相同形状的一块填补在 A 中删除物体的位置，从而得到混合图片 M。

与之前同理，这种混合模式下可以设置每种类别物体被替换的概率。

#### 3.4 数据类别均衡

数据类别不均衡会严重影响模型的性能。例如，如果数据集中某种物体出现次数很少，模型对该类别物体的分割效果差，在该类别上的 IoU 很低，则会直接导致最终各类别交并比的平均值 mIoU 明显被拉低。因此，提升模型在稀有类上的性能是十分必要的。

在本项目中，笔者引入了数据类别均衡策略。首先，笔者统计了训练集数据中各个类别的出现频率。

统计得到数据集中各类别像素数为：

[5.21 × 10^9 , 3.92 × 10^7, 6.45 × 10^8 , 5.71 × 10^6, 3.22 × 10^8]

统计得到数据集中包含各类别物体的图片数为：

[3002, 882, 2836, 242, 2195]

可以看到，如果从像素数的角度进行统计，出现频次最高的类别对应的像素数(5.21 × 10^9)是出现频次最低的类别(5.71 × 10^6)的近 1000 倍。如果通过多次采样稀有类图片的方法来均衡数据，这就意味着包含稀有类的图片应被重复采样近 1000 次。这样会导致整个数据集几乎被几张包含稀有类的图片占据，其他图片出现的次数很少，这也是不合理的。

因此，在进行数据均衡时，笔者参考了包含各类别的图片数，而非像素数。

出现频次越高的类别应该被更少地采样，因此采样频次应为：

[1/3002, 1/882, 1/2836, 1/242,1/ 2195]

做近似化简取整，得到：

[2, 7, 2, 25, 3]

每一位数代表该类别应该采样的频次。例如，一张图片中包含 0、1、4 这三种类别的物体，此时可以读出对应的频次分别为 2、7、3，其中最大值为 7，因 此需要把这张图片重复 7 次，以得到类别较为均衡的数据集。

需要特别说明的是，笔者也尝试过使用类别加权的损失函数来应对类别不 均衡问题，但是效果不如直接对数据集做均衡化。

### 4 UDA 策略

#### 4.1 基础策略

这里主要参考了 DACS[7]这篇工作，通过跨域混合采样的方法进行域自适应训练——将来自两个域的图像与相应的标签和伪标签混合，然后同时使用source domain 的标记数据和这些混合数据训练模型。

##### 4.1.1 混合数据的产生：

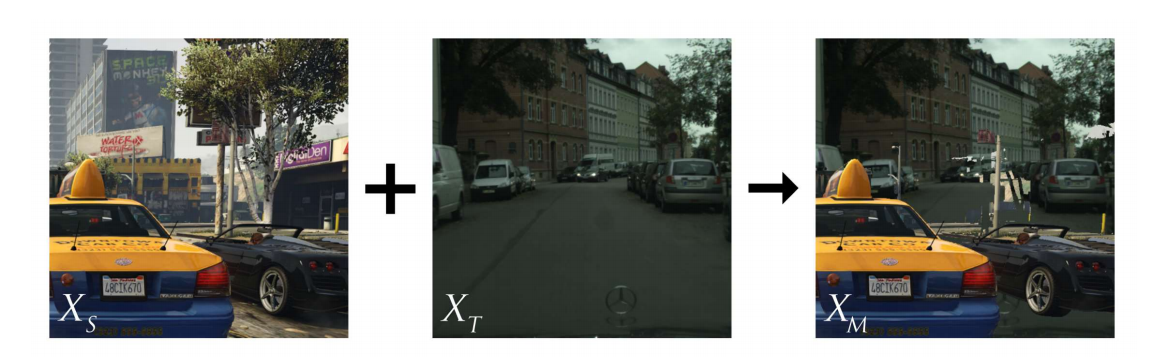
首先，我们通过 teacher model 生成 target domain 数据的伪标签。这里使用到的 teacher model 其实只是我们要训练的 core model 的指数平滑，有一种知识蒸馏和自学习的味道。需要特别指出的是，teacher model 是不进行训练的。

生成 target 数据的伪标注之后，我们将 source domain 的数据及其标注、target domain 的数据及其伪标注通过 Mix Up 的方法混合在一起，得到混合数据。最后，对混合数据做较强的数据增强操作。

下图为笔者绘制的示意图：



在 DACS 的文章中，作者也举了一个例子，如下图所示：

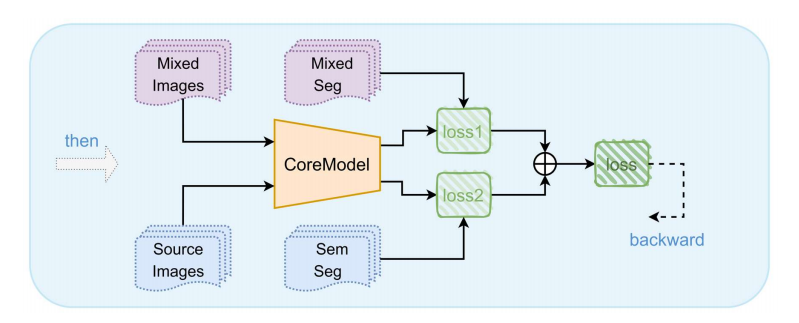


可以看到，在这个例子中，source data 中给出租车和旁边的跑车，被 mix 到了 target data 中，从而得到了混合数据。

##### 4.1.2 模型的训练：

如前文所述，混合数据同时包含来自两个域的物体。其中，来自 source 数据的部分是有标注的，来自 target 数据的部分有 teacher model 生成的伪标注。我们还对混合数据做了较强的数据增强，从而提升了自学习的效果。实验表明，使用这种混合数据对模型进行训练，能够有效地提升模型对于 target domain 的适应能力，从而取得较大的性能提升。

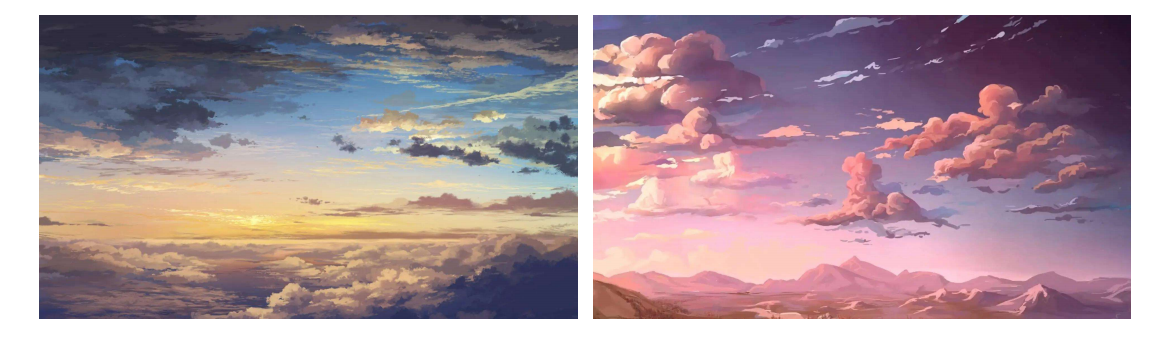
在实际训练过程中，为了防止模型遗忘 source domain 的知识、自学习“学歪了”，我们还会使用 source domain 的数据进行监督训练。因此，训练时的模式如下图所示：



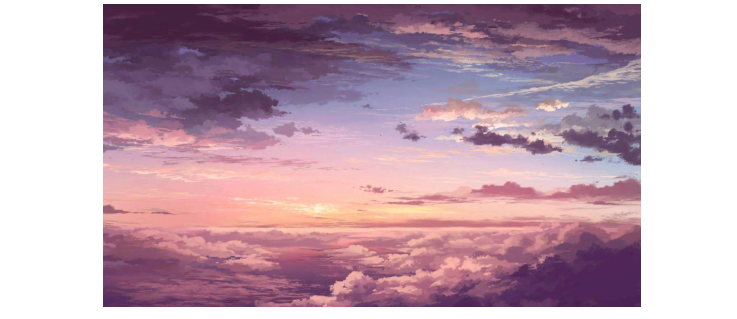
4.2 颜色匹配   
 此部分主要参考了一篇名为 Color Transfer between Images 的论文[8]。具体实现的功能是：将一张图片的颜色风格根据另一张图片进行调整，使二者的颜色风格相匹配。

根据论文，这种效果的实现也相当简单，只需要将图片由 RGB 颜色空间变换到 LAB 颜色空间，并对各通道数值进行线性映射，使得待变换的图片 LAB 各通道数值的均值、标准差与目标图片 LAB 各通道的均值、标准差相等，再变回RGB 颜色空间，就得到了颜色匹配之后的结果。

例如，要进行颜色匹配的图片和目标图片分别如下：



则通过颜色匹配算法，将左图的 LAB 均值方差调整为右图的 LAB 均值方差，最后转回 RGB 颜色空间，可以得到处理结果：



可以看到，颜色匹配效果很显著，处理结果的颜色风格完全与目标图片一致。

在本项目中，笔者考虑使用颜色匹配的方法，来提升 UDA 训练的效果。具体思路是，在进行 source data 和 target data 的混合之前，将 source data 的颜色匹配为 target data，将 target data 的颜色匹配为 source data 的颜色：

(1) 将 source data 的颜色匹配为 target data，是希望带有标注的 source domain 数据能够在色彩上长得更像 target 一些，从而在使用 source 数据进行训练时，模型能够更多地适应 target 的色彩风格；

(2) 将 target data 的颜色匹配为 source data 的颜色，则是将颜色匹配算法看作了一种较强的 augment，通过引入这种 augment 来提升自学习的效果。

实验表明，加入颜色匹配算法后，UDA 训练的效果得到较为明显的提升。

#### 4.3 过滤低置信度伪标签

在使用 teacher model 生成伪标签的时候，并不是所有像素的预测类别都是正确的，这是很正常的。因此，把这种带有部分错误的伪标注当作事实进行学习，相当于引入了许多噪声。为了减小这种错误对 UDA 训练的影响，我们希望通过某种标准选出一些预测置信度较低的像素，并将这些像素的伪标签过滤掉，即放弃在这些像素上进行训练。

##### 4.3.1 两种标准

(1) 根据可能性进行过滤

本项目中使用的 CLIPSeg 模型，本身是做二分类任务的，即预测每个像素是属于某种指定的物体还是属于背景。因此，在模型的输出中，每个数字都表示对应像素属于这种类别物体的概率。

因此，一种直接的想法是，如果某一像素属于预测类别的概率（如 0.8）超过一定的阈值（如 0.5），则认为该像素的预测类别可信；否则认为不可信。通过这种方法，过滤掉不可信的像素。

(2) 根据可能性与熵的差进行过滤

上一个判断指标具有其合理性，但是也会存在一些可能的问题。

例如，可能某个像素属于 5 个类别的概率分别为[0.8, 0.7, 0.6, 0.7, 0.6]，这样，最大值 0.8 是大于阈值 0.5 的，但是实际上和属于其他类别的概率也差不多，实际置信度可能不高。如前文所述，这里每一个数字具有该像素属于对应类别的概率的意义，因此没有对 5 个概率之和做归一化操作。

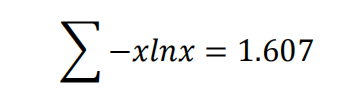
可以看到，在这种情况下，直接根据像素属于预测类别的概率是否大于阈值来判断是否可信可能未必准确。因此，笔者通过将概率最大值减去熵的方法来缓解这一问题。

仍然以之前这个概率[0.8, 0.7, 0.6, 0.7, 0.6]为例：

首先求得 max prob = 0.8，即该像素最可能是背景；

然后做 softmax 操作，得到归一化概率：[0.225, 0.203, 0.184, 0.203, 0.184]

从而求得熵：



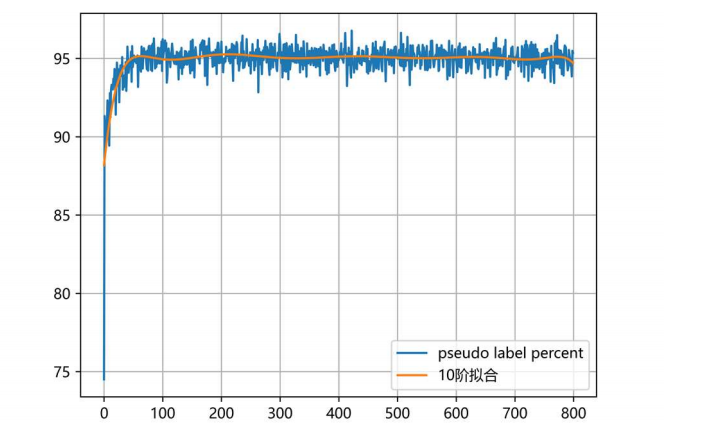
因此，该像素最终的判别指标为0.8 − 1.607 = −0.807。若此时设置阈值为−0.8，则由于−0.807 < −0.8，该像素将会被判别为不可信。

##### 4.3.2 P 控制实现固定比例过滤

如果设置过滤阈值为某一固定值，则在训练过程中，随着模型性能的提升，绝大多数像素的置信度都在提升，因此不可信的像素会越来越少、过滤掉的像素越来越少，过滤作用逐渐减弱。

为了保证一定的过滤效果，我们可以通过比例控制实现始终过滤掉一定比例（如 10%）的不可信像素：在训练过程中，在每个 batch 中统计不可信像素的比例P为设定的 过滤比例（如 10%），K为比例系数。

例如，设定过滤 5%的不可信标签，则在训练过程中采用的可信伪标签占比的变化情况如下：



对应的过滤阈值的变化情况为：



其中，横坐标表示记录的次数，实验过程中每 5 个 iter 记录一次。可以看到，控制效果较好，稳定后使用采纳约 95%的伪标签；阈值在初始的调整阶段迅速下降，以满足过滤 5%的要求，后期模型性能提升，阈值逐渐增长。

### 5后处理

在使用模型进行验证、测试时，直接对模型的输出取 argmax 得到的输出结果的局部连续性较差，主要存在两个问题：

(1) 一些零散的像素预测类别与周围不同；

(2) 有一些预测的缺陷。比如同一块像素，周围都预测为纸板，中间却有一个小空洞预测成了背景。

为了解决这样的问题，笔者引入了形态学后处理方法，主要包含两个步骤：

(1) 对模型输出的每个像素属于各个类别的可能性做灰度开、闭运算，然后再取 argmax 得到预测结果；

(2) 消除预测结果中的空洞。

下面举一个例子进行说明：

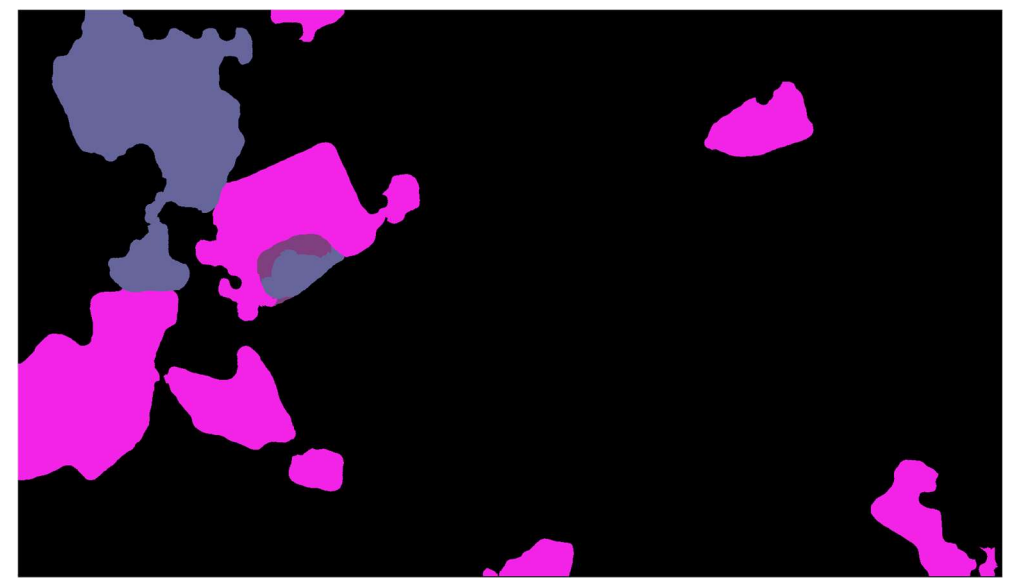
直接对模型输出取 argmax 得到的分割结果：



经过灰度开、闭运算之后再取 argmax 得到的结果：



消除分割区域空洞之后的结果：



可以看到，相比于原始的预测结果，对模型输出做灰度开、闭运算再取argmax 得到的结果具有更好的局部连续性，结果中不再包含一些零零星星的和周围预测类别不同的小像素点。但是，此时预测结果中依然包含一些明显的空洞。

经过第二步操作后，预测结果中的空洞被消除。最终，预测结果较未处理之前得到明显改善。

实验证明，加入形态学后处理后，测试 mIoU 稳定提升 0.5%左右。

### 6项目总结

虽然本项目是一个深度学习语义分割的项目，但是笔者从搭建初期开始就将它当作一个完整的工程项目来看待，因此在结构设计核框架搭建方面投入了 不少时间。最终，本项目主要有以下特点：

1. 代码层次结构完整，接口封装合理，注释十分丰富，方便后续开发。本项目中，由本人和清华大学韩东辰合作开发的部分共有 2688 行代码（不包括空行、注释），另有 1214 行注释。对于深度学习项目来说，这样的代码量并不算少。因此，为了提升项目的结构性，笔者多次重构项目框架，力求简洁可读。笔者为所有核心算法函数编写了非常详尽的注释，举例如下图：



1. 开发了若干辅助工具，方便了解模型存在的问题。在本项目中，笔者从零做起，开发了诸多辅助工具，如：训练日志记录器、平均器、数据可视化工具、训练日志参数绘图工具等等。其中，数据可视化工具可以直接将训练过程中的某个数据 tensor 对应的图片显示出来，方便检验 crop、mix up 等操作是否存在bug；训练日志参数绘图工具能够绘制训练日志中任意变量的变化曲线，并可以对其进行拟合，前文中展示固定比例过滤伪标签的曲线和后文中诸多曲线结果的展示即使用了这一工具。
2. 引入了诸多基础方法。本项目中借鉴了诸多工作的方法，引入了许多基础的算法，具体方法已在前文中给出，此处不再详述。

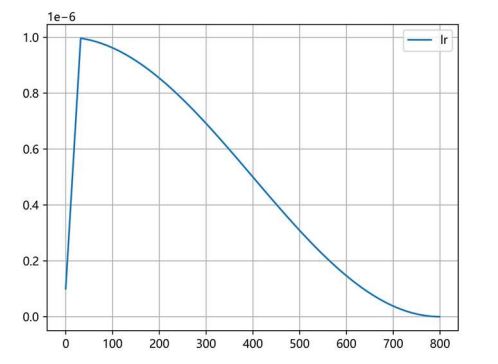
#### 6.2 算法创新

前文中已经详细介绍各算法，这里简要总结一下创新点。

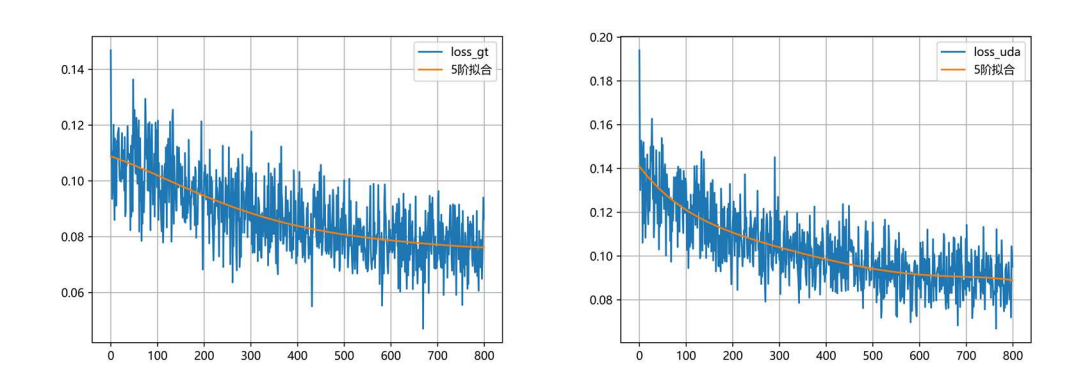
本项目中，笔者完成的算法创新主要包括：引入多种 Mix Up 模式、修改数据类别均衡策略、实现多个版本的 UDA 策略、引入图片颜色匹配方法、修改伪标签过滤指标和实现固定比例过滤、引入形态学后处理等等。此外，笔者还进行了大量实验，广泛探索了诸多因素对无监督域迁移训练的影响，其中也包括一些不是很成功的尝试。限于报告篇幅，此处不再一一介绍。

#### 6.3 实验结果

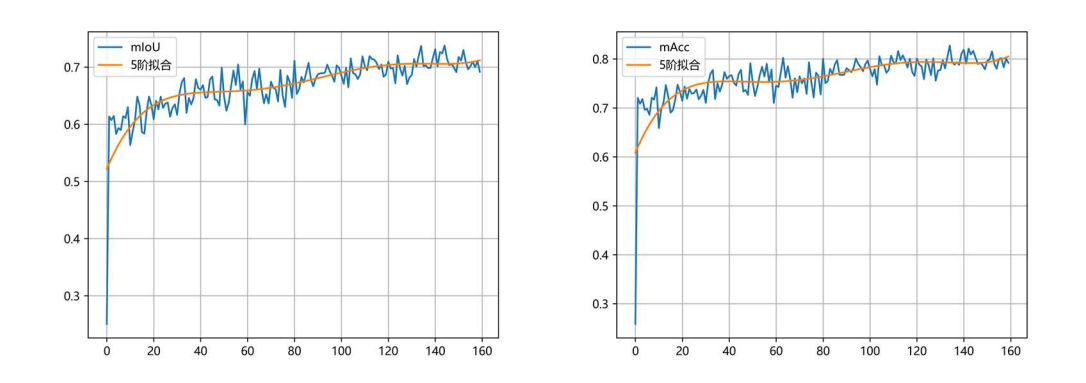
如前文所述，本项目中的训练主要分为两个阶段，分别是在 source domain 上进行有监督的预训练和无监督域迁移训练。限于报告篇幅，此处主要展示无监督域迁移训练中的结构。 在训练过程中，学习率采用 warm-up+cosine 的调整策略，如下：



训练过程中，有监督部分的 loss 和使用 mix data 进行 uda 训练的 loss 的变化情况分别如下：



模型在验证集上的 mIoU 和 mAcc 测试指标变化如下：



模型的测试结果需要在比赛官网上进行评测，本模型在测试集上的性能为mIoU(source only) = 51%，Acc=92.30 ，mIoU = 52%。，Acc=92.11。其中，mIoU(source only)指的是未经过第二阶段 UDA 训练，直接用 source domain 监督训练模型在测试集上测试的结果；mIoU则指的是完成两阶段训练的模型在测试集上的结果。

### 7思考与展望

在本项目中，笔者使用深度学习语义分割的方法，完成垃圾分类任务，并提出了创新性的无监督域迁移训练方法，取得较好的效果。

本项目有诸多优势，如模型性能较好、训练消耗资源较少、项目架构较为清晰、代码可读性较好等等，但同时也有一些不足，如对语义分割的一些技巧不是很熟练，模型decoder的设计有欠缺，CLIP的finetune会有模型过拟合的现象，合成数据集迁移效果差，Kaggle上的一些方法也没有很好的适配上去，UDA在最终test上任有很大的提升空间。

本项目，前期工作均是由清华大学韩东辰同学主导，在他的带领下，打好了扎实的基础。绠短汲深，笔者实力欠缺，最终比赛没有夺冠，甚是羞愧，希望在之后能够有机会继续探究 UDA 相关方向。