Algorithm 1: Pseudocode: Pseudo label refinement of S4MC, PyTorch-like style.

```
# X: predict prob of unlabeled data B x C x H x W
# k: number of neigbors

#create neighborhood tensor
neigborhood=[]
X = X.unsqueeze(1)
X = torch.nn.functional.pad(X, (1, 1, 1, 1, 0, 0, 0, 0))
for i,j in [(None,-2),(1,-1),(2,None)]:
    for k,l in [(None,-2),(1,-1),(2,None)]:
        if i==k and i==1:
            continue
        neighborhood.append(X[:,:,i:j, k:l])
neighborhood = torch.stack(neighborhood)

#pick k neighbors for union event
ktop_neighbors,neigbor_idx=torch.topk(neighborhood, k=k,axis=0)
for nbr in ktop_neighbors:
    beta = torch.exp((-1/2) * neigbor_idx)
    X = X + beta*nbr - (X*nbr*beta)
```

A Visual Results

We present in Figs. A.1 and A.2 an extension of Fig. 3, showing more instances from the unlabeled data and the corresponding pseudo labeled with the baseline model and S4MC. In Fig. A.2 we can see that our method can eliminated undesired entities, while sometimes it also eliminate good predictions, it the process of pseudo-labeling.

Our method can achieve more accurate predictions during the inference phase without refinements. This results in more seamless and continuous predictions, which accurately depict objects spatial configuration.

B Computational Cost

Let us denote the image size by $H \times W$ and the number of classes by C.

First, the predicted map of dimension $H \times W \times C$ is stacked with the padded-shifted versions, creating a tensor of shape [n,H,W,C]. K top neighbors are picked via top-k operation and calculate the union event as presented in Eq. (9). (The pseudo label refinement pytorch-like pseudo-code can be obtained in Algorithm 1 for $N = 4$ and k max neighbors.)

The overall space (memory) complexity of the calculation is $O(n \times H \times W \times C)$, which is negligible considering all parameters and gradients of the model. Time complexity adds three tensor operations (stack, topk, and multiplication) over the $H \times W \times C$ tensor, where the multiplication operates k times, which means $O(k \times H \times W \times C)$. This is again negligible for any reasonable number of classes compared to tens of convolutional layers with hundreds of channels.

To verify that, we conducted a training time analysis comparing FixMatch and FixMatch + S4MC over PASCAL with 366 labeled examples, using distributed training with 8 Nvidia RTX 3090 GPUs. FixMatch average epoch takes 28:15 minutes, and FixMatch + S4MC average epoch takes 28:18 minutes, an increase of about 0.2% in runtime.

C Implementation Details

All experiments were conducted for 80 training epochs with the stochastic gradient descent (SGD) optimizer with a momentum of 0.9 and learning rate policy of $lr = lr_{\text{base}} \cdot \left(1 - \frac{\text{iter}}{\text{total iter}}\right)^{\text{power}}$.

For the teacher averaging consistency, we apply resize, crop, horizontal flip, GaussianBlur, and with a probability of 0.5, we use Cutmix (Yun et al., 2019) on the unlabeled data.

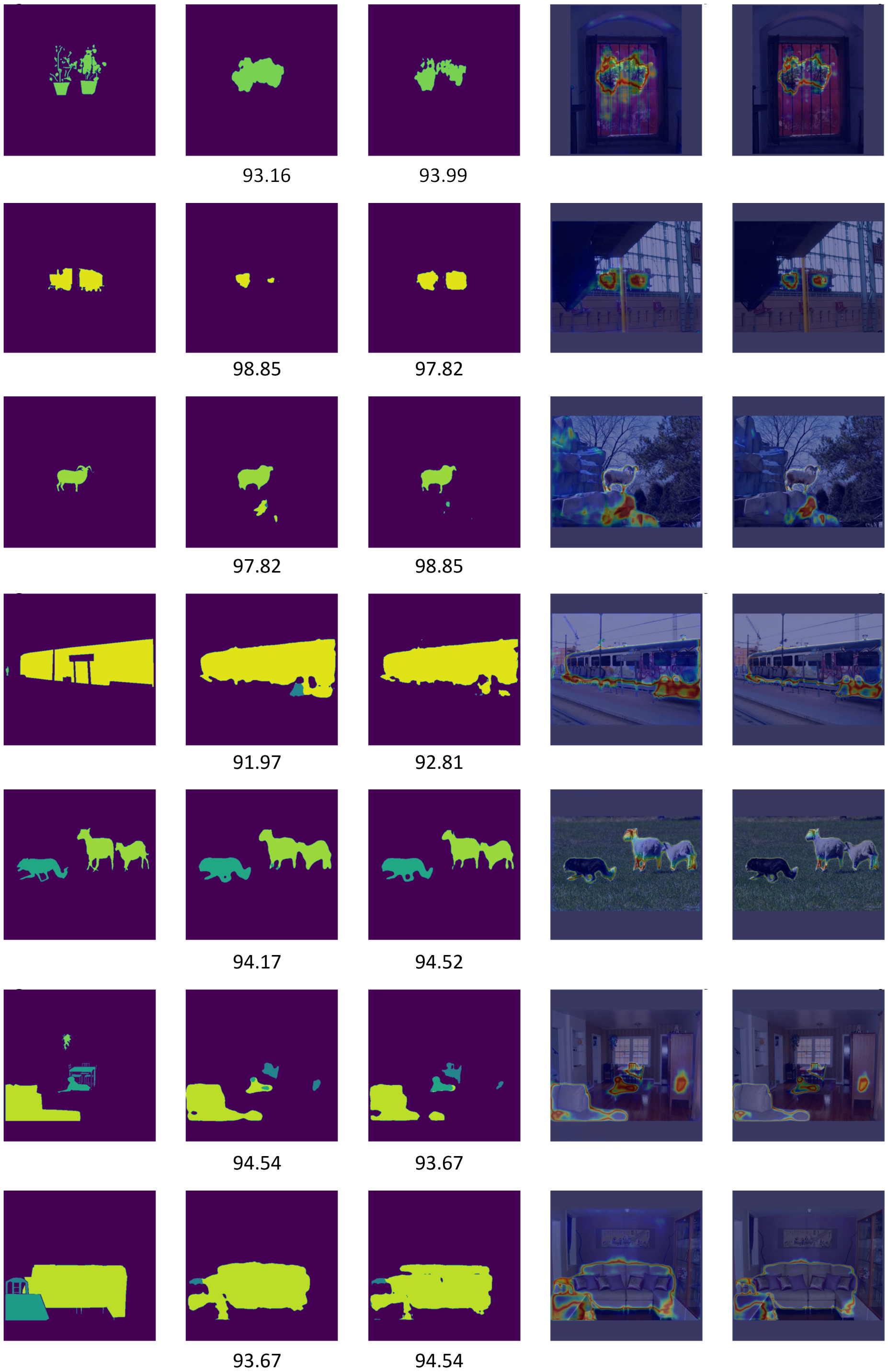


Figure A.1: **Example of refined pseudo labels**, structure of the figure as Fig. 3 and the numbers under the predictions show the pixel-wise accuracy of the prediction map.

For the augmentation variation consistency (Sohn et al., 2020a; Yang et al., 2023), we apply resize, crop, and horizontal flip for weak and strong augmentations as well as ColorJitter, RandomGrayscale, and Cutmix for strong augmentations.

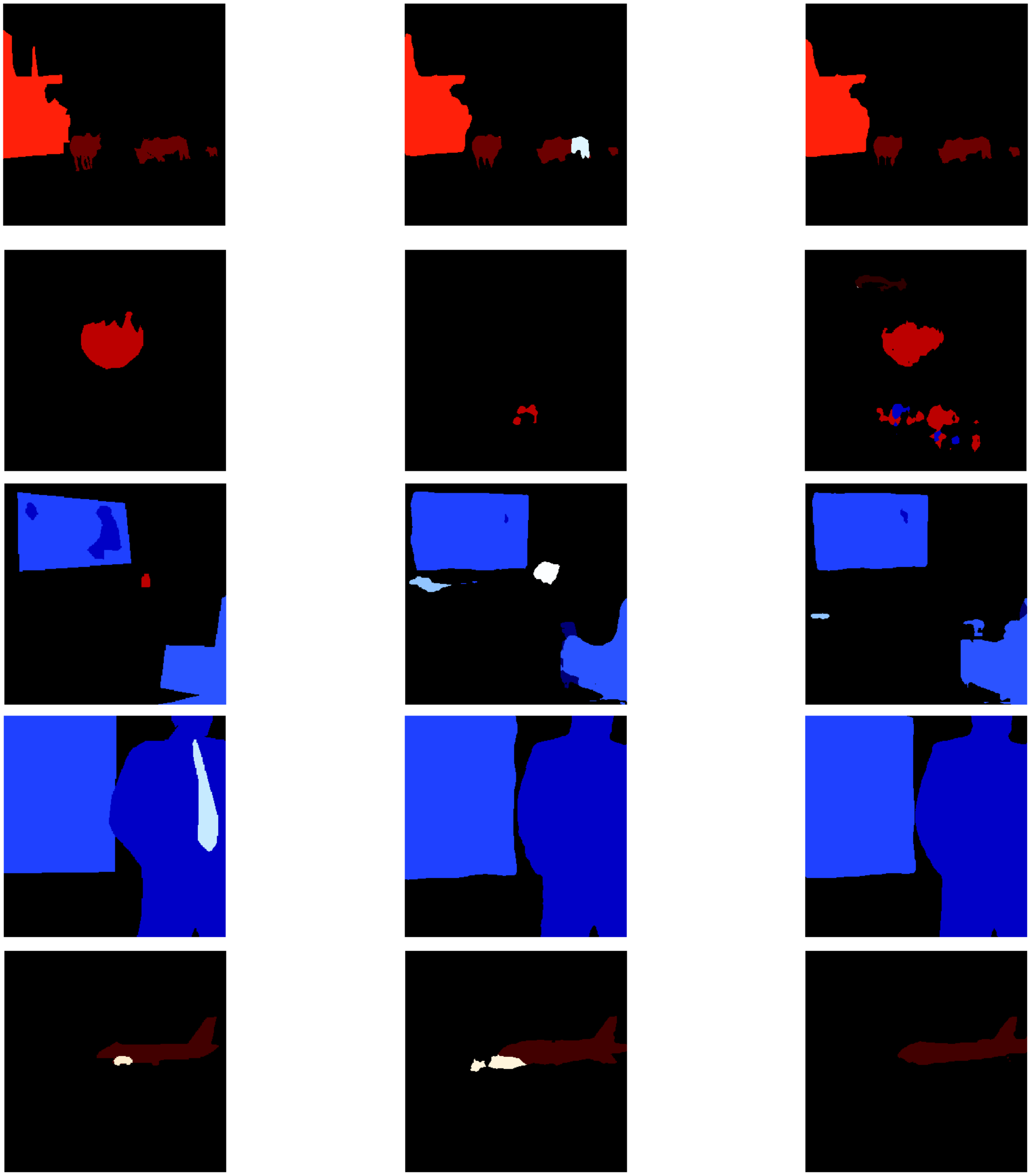


Figure A.2: Qualative results of our method comparison to UniMatch baseline over COCO with 1/32 of the labeled examples. The segmentation map Left to right: Ground Truth, UniMatch prediction, S4MC Prediction

For PASCAL VOC 12 $lr_{\text{base}} = 0.001$ and the decoder only $lr_{\text{base}} = 0.01$, the weight decay is set to 0.0001 and all images are cropped to 513×513 and $\mathcal{B}_l = \mathcal{B}_u = 3$.

For Cityscapes, all parameters use $\text{lr}_{\text{base}} = 0.01$, and the weight decay is set to 0.0005. The learning rate decay parameter is set to $\text{power} = 0.9$. Due to memory constraints, all images are cropped to 769×769 and $\mathcal{B}_\ell = \mathcal{B}_u = 2$. All experiments are conducted on a machine with 8 Nvidia RTX A5000 GPUs.

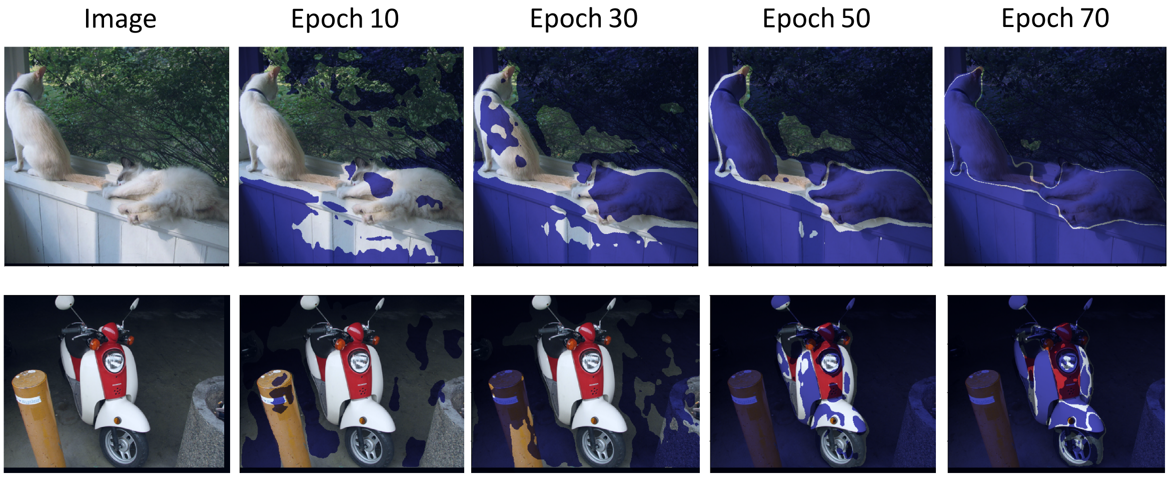


Figure E.1: Qualitative evolution of the pseudo labeling process of S4MC. The figure shows the progress over time of the pixels that assign as pseudo labels w.r.t time.

D Limitations and Potential Negative Social Impacts

Limitations. The constraint imposed by the spatial coherence assumption also restricts the applicability of this work to dense prediction tasks. Improving pseudo labels' quality for overarching tasks such as classification might necessitate reliance on data distribution and the exploitation of inter-sample relationships. We are currently exploring this avenue of research.

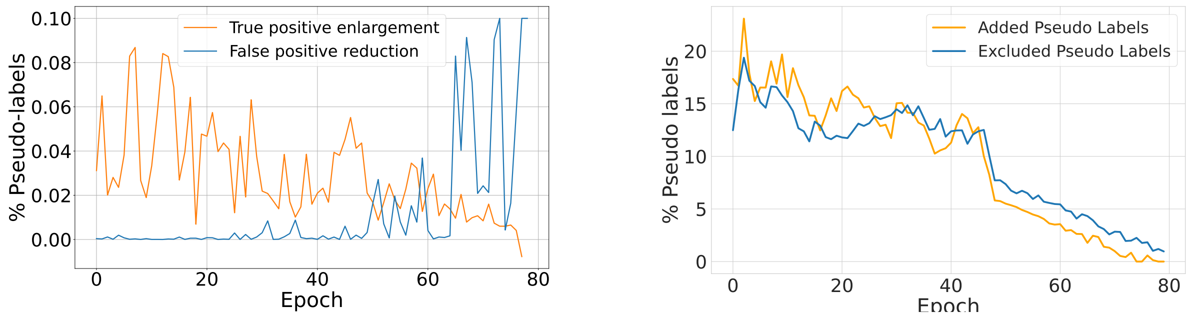
Societal impact. Similar to most semi-supervised models, we utilize a small subset of annotated data, which can potentially introduce biases from the data into the model. Further, our PLR module assumes spatial coherence. While that holds for natural images, it may yield adverse effects in other domains, such as medical imaging. It is important to consider these potential impacts before choosing to use our proposed method.

E Pseudo Labels Quality Analysis

The quality improvement and the quantity increase of pseudo labels are shown in Fig. 4. Further analysis of the quality improvement of our method is demonstrated in Fig. E.2a by separating the *true positive* and *false positive*, and Fig. E.1 shows qualitative evolution of the pseudo labeling process.

Within the initial phase of the learning process, the enhancement in the quality of pseudo labels can be primarily attributed to the advancement in true positive labels. In our method, the refinement not only facilitates the inclusion of a larger number of pixels surpassing the threshold but also ensures that a significant majority of these pixels are high quality.

As the learning process progresses, most improvements are obtained from a decrease in false positives pseudo labels. This analysis shows that our method effectively minimizes the occurrence of incorrect pseudo labeled, particularly when the threshold is set to a lower value. In other words, our approach reduces confirmation bias from decaying the threshold as the learning process progresses.



(a) **Quality of pseudo labels** from Fig. 4 separated *True positive* and *False positive* analysis. *True positive* explain the major part of improvement at early stage, while reducing *false positive* explain the enhancement later on.

(b) **Added and excluded pseudo labels**. The pixel-wise pseudo labels S4MC added and excluded over time (i.e. the sum of the graphs is the total pixels that change because of S4MC).

Figure E.2: Analysis of the results of DeepLab V3+ on PASCAL VOC 12

In Fig. E.1 we can see that at late stages of the training process, the bounderies of objects are the most ambiguate and thus the hardest to assign with pseudo labels with S4MC.

F Weak–Strong Consistency

We need to redefine the supervision branch to adjust the method to augmentation level consistency framework (Sohn et al., 2020a; Zhang et al., 2021; Wang et al., 2023). Recall that within the teacher averaging framework, we denote $f_{\theta_s}(\mathbf{x}_i)$ and $f_{\theta_t}(\mathbf{x}_i)$ as the predictions made by the student and teacher models for input $\mathbf{x}_i$, where the teacher serves as the source for generating confidence-based pseudo labels. In the context of image-level consistency, both branches differ by augmented versions $\mathbf{x}_i^w$, $\mathbf{x}_i^s$ and share identical weights f_θ. Here, $\mathbf{x}_i^w$ and $\mathbf{x}_i^s$ represent the weak and strong augmented renditions of the input $\mathbf{x}_i$, respectively. Following the framework above, the branch associated with weak augmentation generates the pseudo labels.

F.1 Confidence Function Alternatives

In this paper, we introduce a confidence function to determine pseudo label propagation. We introduced κ_{margin} and mentioned other alternatives have been examined.

Here, we define several options for the confidence function.

The simplest option is to look at the probability of the dominant class,

$$\kappa_{\max}(x_{j,k}^i) = \max_c p_c(x_{j,k}^i), \tag{F.1}$$

which is commonly used to generate pseudo labels.

The second alternative is negative entropy, defined as

$$\kappa_{\text{ent}}(x_{j,k}^i) = \sum_{c \in C} p_c(x_{j,k}^i) \log\big(p_c(x_{j,k}^i)\big). \tag{F.2}$$

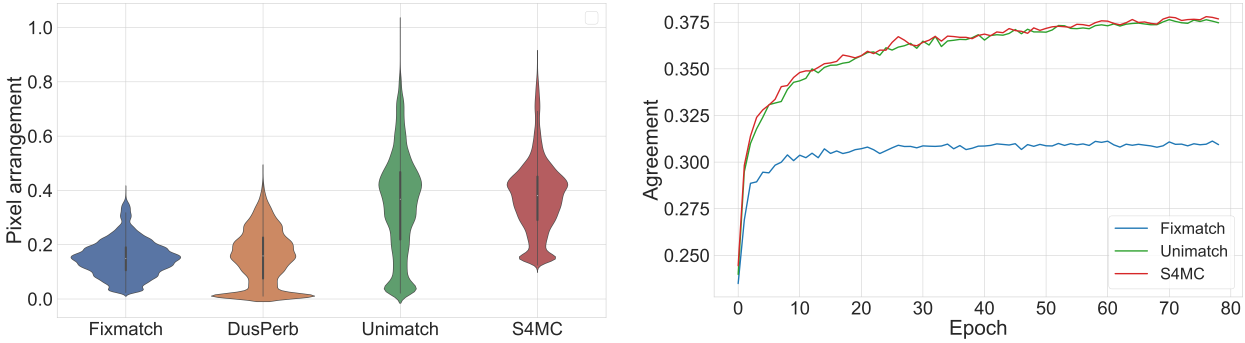
Note that this is indeed a confidence function since high entropy corresponds to high uncertainty, and low entropy corresponds to high confidence.

The third option is for us to define the margin function (Scheffer et al., 2001; Shin et al., 2021) as the difference between the first and second maximal values of the probability vector and also described in the main paper:

$$\kappa_{\text{margin}}(x_{j,k}^i) = \max_c(p_c(x_{j,k}^i)) - \max2_c(p_c(x_{j,k}^i)), \tag{F.3}$$

Table F.1: Ablation study on the confidence function κ, over Pascal VOC 12 with partition protocols using ResNet-101 backbone.

Function	**1/4 (366)**	**1/2 (732)**	**Full (1464)**
κ_{max}	74.29	76.16	79.49
κ_{ent}	75.18	77.55	79.89
κ_{margin}	75.41	77.73	80.58



(a) The spatial agreement as we define in in 9 compared between different variations of Unimatch and S4MC.

(b) The spatial agreement, compared between different variations of (Yang et al., 2023) and S4MC over time.

Figure F.1: Spatial agreement analysis off diffrent methods on PASCAL VOC 12 using ResNet-101 backbone.

where max2 denotes the vector's second maximum value. All alternatives are compared in Table F.1.

F.2 Decomposition and Analysis of Unimatch

Unimatch (Yang et al., 2023) investigating the consistency and suggest using FixMatch (Sohn et al., 2020a) and a strong baseline for semi-supervised semantic segmentation. Moreover, they provide analysis that shows that combining three students for each supervision signal (one feature level augmentation, Channel Dropout, denoted by CD, and two strong augmentations, denoted by S1 and S2) can enhance performance further more, since each student branch can learn a slightly different features. Fusing Unimatch and our method did not provide significant improvements, and we examined the contribution of different components of Unimatch. We measured the pixel agreement as described in Eq. (9) and showed that the feature perturbation branch has the same effect on pixel agreement as S4MC. Fig. F.1 present the distribution of agreement using FixMatch (S1), DusPerb (S1,S2), Unimatch (S1, S2, CD) and S4MC (S1, S2).

G Bounding the Joint Probability

In this paper, we had the union event estimation with the independence assumption, defined as

$$p_c^1(x^i_{j,k}, x^i_{\ell,m}) \approx p_c(x^i_{j,k}) \cdot p_c(x^i_{\ell,m}) \tag{G.1}$$

In addition to the independence approximation, it is possible to estimate the unconditional expectation of two neighboring pixels belonging to the same class based on labeled data:

$$p_c^2(x^i_{j,k}, x^i_{\ell,m}) = \frac{1}{|\mathcal{N}_l| \cdot H \cdot W \cdot |\mathbf{N}|} \sum_{i \in \mathcal{N}_l} \sum_{j,k \in H \times W} \sum_{\ell,m \in \mathbf{N}_{j,k}} \mathbb{1}\{y^i_{j,k} = y^i_{\ell,m}\}. \tag{G.2}$$

To avoid overestimating that could lead to overconfidence, we set

$$p_c(x^i_{j,k}, x^i_{\ell,m}) = \max(p_c^1(x^i_{j,k}, x^i_{\ell,m}), p_c^2(x^i_{j,k}, x^i_{\ell,m})) \tag{G.3}$$

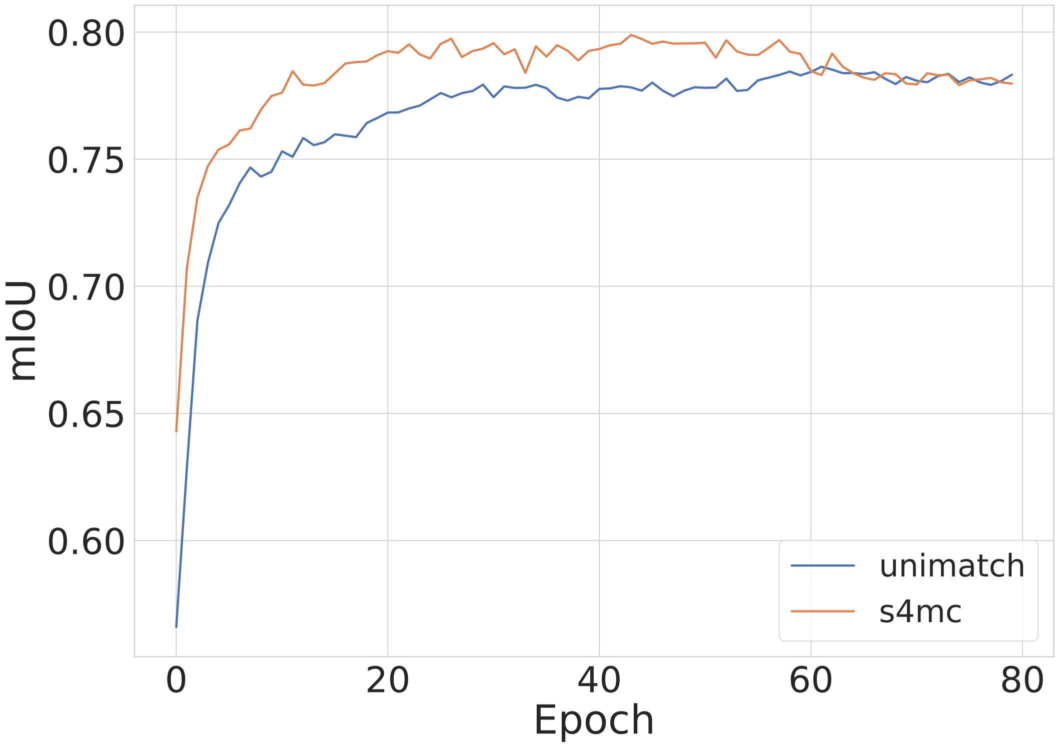


Figure H.1: Training curve of the mIoU of S4MC and Unimatch using Resnet-101 on PASCAL VOC 12 with 366 annotated examples

That upper bound of joint probability ensures that the independence assumption does not underestimate the joint probability, preventing overestimating the union event probability. Using Eq. (G.3) increase the mIoU by **0.22** on average, compared to non use of S4MC refinement, using 366 annotated images from PASCAL VOC 12 Using only Eq. (G.2) reduced the mIoU by **-14.11** compared to the non-use of S4MC refinement and harmed the model capabilities to produce quality pseudo labels.

H Convergence time

In this section we show that not only S4MC achieve competitive results compared with state-of-the-art methods, but also that PLR helps the model to converge faster. In Fig. H.1 we show that S4MC achieve the same mIoU as Unimatch after only 10 epochs, providing faster convergence in terms of training time using the exact same optimization process.