Abstract—Currently, state-of-the-art semi-supervised learning (SSL) segmentation methods employ pseudo labels to train their models, which is an optimistic training manner that supposes the predicted pseudo labels are correct. However, their models will be optimized incorrectly when the above assumption does not hold. In this paper, we propose a Pessimistic Consistency Regularization (PCR) which considers a pessimistic case that pseudo labels are not always correct. PCR makes it possible for our model to learn the ground truth (GT) in pessimism by adaptively providing a candidate label set containing $K$ proposals for each unlabeled pixel. Specifically, we propose a pessimistic consistency loss which trains our model to learn the possible GT from multiple candidate labels. In addition, we develop a candidate label proposal method to adaptively decide which pseudo labels are provided for each pixel. Our method is easy to implement and could be applied to existing baselines without changing their frameworks. Theoretical analysis and experiments on various benchmarks demonstrate the superiority of our approach to state-of-the-art alternatives.

Index Terms—Semi-supervised learning, Semantic segmentation.

I. INTRODUCTION

SEMANTIC segmentation enables accurate scene understanding [7], which benefits a variety of downstream tasks such as autonomous driving, computer-aided diagnosis, and augmented reality [31]. Recently, semantic segmentation methods based on deep learning have been widely explored and have achieved remarkable progress. However, these methods require large-scale pixel-level labeled data but pixel-level annotation is quite time-consuming. Therefore, semi-supervised learning (SSL) is introduced into the semantic segmentation task to utilize cheap and large amounts of unlabeled data.

Consistency regularization and pseudo label are two popular techniques to deal with unlabeled data in SSL segmentation. Consistency regularization aims to obtain invariant segmentation predictions for each unlabeled pixel under various perturbations, e.g., input perturbation [54], [60], feature perturbation [56], or network perturbation [6]. The pseudo label method produces an artificial label for each unlabeled pixel and then trains an SSL model by taking the pseudo label as the ground truth (GT) label [17], [18], [30]. Recently, methods combining consistency regularization and pseudo label have achieved the state-of-the-art (SOTA) performance in SSL segmentation [6], [54], [56], [60], which compute the consistency loss by pseudo labels to train their models.

These methods rely on an optimistic assumption that they could generate correct pseudo labels based on the knowledge learned from labeled data. Unfortunately, this assumption does not always hold in practice [12], [25] since the labeled data are usually sampled randomly from the whole data, and their distribution might be biased [26], [35]. Some previous works observed that wrong pseudo labels force their models to predict wrong classes with high confidence [42], which seriously degrades the performance of SSL models [24], [39], [52]. To mitigate this problem, some existing approaches propose to filter out low-confidence predictions and train their models by only high-confidence predictions [52], [60]. Some other works utilize low-confidence predictions by negative learning [38], [47]. These methods are based on the basic fact that an uncertain prediction usually gets confused among only a few classes instead of all classes. Hence, they use uncertain predictions as negative samples for those unlikely classes. However, filter-based approaches cannot utilize all unlabeled pixels, resulting in insufficient pixel utilization and categorically imbalanced training [47]. For negative learning-based methods, we analyze that they still implicitly learn the pseudo-label, which means the errors in their models accumulate when the pseudo label is wrong.

In this paper, we propose a Pessimistic Consistency Regularization (PCR) method, which is based on a form of pessimism concerning the pseudo label, that is, pseudo labels are not always correct. In order to make it possible for our model to learn the GT label in this pessimistic situation, we provide a candidate label set containing $K$ label proposals for each pixel. As shown in Figure 1, our candidate label set covers GT labels better than the existing pseudo label [6], [54], [56], [60], thus training a better SSL model than existing optimistic methods. Specifically, our PCR contains two key components. One is a candidate label proposal method, which adaptively provides $K$ pseudo label proposals for each pixel. And the other one is a pessimistic consistency loss, which allows our model to learn the possible GT label from the candidate label set in a multi-label classification manner.

We evaluate the effectiveness of our PCR on SSL segmentation with VOC2012, Cityscapes, and VOC2012 LowData datasets using CPS [6] and AEL [13] as baselines. Furthermore, we theoretically and empirically analyze that PCR outperforms existing methods since it revises the gradients of learning unlabeled data when the pseudo label is wrong. The main contributions of this work are:

- PCR provides a new idea for improving SSL segmentation methods, that is, learning the possible GT label from a multi-label classification manner.
- We propose a candidate label proposal algorithm to adaptively provide multiple possible labels for each pixel. In addition, we propose a pessimistic consistency loss which allows our model to learn the possible GT label in
Fig. 1: (a) Existing methods using top-1 prediction as pseudo-label to learn unlabeled data. (b) Our proposed PCR, which selects K predictions into a candidate label set to learn unlabeled data.

the candidate label set.

- PCR is easy to implement and could be applied to existing methods without changing their frameworks.

II. RELATED WORK

A. Semi-supervised Learning

The purpose of semi-supervised learning is to improve the performance of a model trained on limited labeled data by utilizing a large amount of unlabeled data. Current deep SSL classification methods utilize entropy minimization and consistency regularization to learn the unlabeled data. Entropy minimization based methods enforce the predictions of unlabeled data to be sharp which is commonly realized by training their model via pseudo labels [2], [3], [20], [24], [39], [49]. Consistency regularization based methods enforce their models to predict consistent results for the same image in different perturbations to make the classification boundaries pass through the data sparse region. VAT [33] leverages consistency regularization by adding adversarial noise on unlabeled images. UDA [49], MixMatch [3], ReMixMatch [2], and EnAET [46] employ data interpolation and weak and strong augmentation to computed the consistency loss. Temporal Model [23] and Mean teacher [42] use a teacher model and a student model and compute the consistency of their predictions to train the student model. And methods combining consistency regularization and entropy minimization have achieved SOTA performance currently [39], [57], whose common feature is computing consistency regularization using pseudo labels. There are also some works that focus on improving existing methods. Sinkhorn Label Allocation [41] develops an annealing strategy for label assignment allowing for the inclusion of prior knowledge on class proportions via flexible upper-bound constraints. Chen et al. [5] introduce the thought of negative sampling into semi-supervised learning by simply adding a loss function. Dash [52] uses a dynamic threshold to generate accurate pseudo labels based on FixMatch. Other Methods like SimPLE [13] and CReST [48] are proposed to utilize the information between classes and to solve the problem of category imbalance, respectively. There are also some graph-based methods solving the semi-supervised problem through label propagation [4], [19], [59].

B. Semi-supervised Semantic Segmentation

Semantic segmentation is a basic task of computer vision that requires abundant high-quality pixel-level annotation. Considering the expensive labor costs of pixel-level annotation, some works propose to explore the semi-supervised semantic segmentation problem and achieve promising results. Some previous methods use Generative Adversarial Network (GAN) to train their models via a discriminative loss over unlabeled data [32], or utilize consistency regularization [50], [51] and entropy minimization [14] to learn unlabeled data referring to SSL classification algorithms. More recently, SOTA semi-supervised segmentation methods combine both entropy minimization and consistency regularization to train their models. PseudoSeg [60], AEL [13] and Jianglong Yuan et al. [54] propose to use the pseudo label generated from weak augmented image to supervise the prediction of strong augmented image. CPS [6] designs a mutual learning mechanism that trains two student models with pseudo labels from each other. U2PL [47] proposes to utilize unreliable pixels by negative learning in a contrastive learning manner to make full use of the unlabeled pixels. Yi et al. [53] propose to use label propagation to refine the pseudo labels. In addition, some methods propose using pixel-level contrastive learning to utilize the information from unlabeled data [22], [58]. CAC [22] uses contrastive learning to learn a model which is robust to changing backgrounds. PC²Seg [58] proposes a new negative sampling technique that provides reliable negative samples for SSL segmentation. Different from existing methods, we propose for the first
time to pessimistically learn from unlabeled data using an adaptive candidate label set, which effectively reduces the errors brought by wrong pseudo-labels.

III. Method

We first describe the semi-supervised learning (SSL) segmentation formally. In particular, for the SSL segmentation task, we have a small labeled data set \( D_l = \{(x_i, y_i)\}_{i=1}^L \) and a large unlabeled data set \( D_u = \{x_u\}_{u=1}^U \), where \( L \) is the size of labeled dataset, and \( U \) is the size of unlabeled dataset \((L \ll U)\). The \( x_i, y_i, x_u \) are the image and label of the \( s \)-th labeled data and the image of the \( u \)-th unlabeled data, respectively.

The purpose of SSL segmentation is to learn the parameters \( \theta \) of a segmentation model \( F(\cdot; \theta) \) by optimizing a loss function that contains both supervised and unsupervised losses:

\[
L = \frac{1}{L} \sum_{i=1}^L L_{sup}(F(x_i; \theta)) + \beta \frac{1}{|U|} \sum_{u=1}^U L_{uns}(F(x_u; \theta)),
\]

where \( L_{sup} \) and \( L_{uns} \) are supervised loss and unsupervised loss, and \( \beta \) is the regularization weight.

A. Motivation and preliminaries

The unsupervised loss in Eq. 1 is formulated as the cross-entropy loss between model output and a pseudo label in current SOTA methods [6], [13], [54], [60]. And the pseudo label is also predicted by their models, whose paradigm is:

\[
\tilde{y}_u = 1(\arg \max_{c} F(x_u)), \quad z_u = \tilde{F}(\tilde{x}_u)
\]

\[
L_u^o = \frac{1}{|S|} \sum_{s=1}^{S} L_{us}^o (z_{us}, \tilde{y}_u)
\]

\[
= \frac{1}{|S|} \sum_{s=1}^{S} \sum_{c=1}^{C} -y_{us} \log \left( \frac{\exp(z_{us}^c)}{\sum_{n=1}^{C} \exp(z_{us}^n)} \right),
\]

where the \( \tilde{y}_u \) is the one-hot encoding of the pseudo label generated from a segmentation model \( F \). The \( 1 \) is the one-hot-encoding function. The \( z_u \) is the output of disturbed model \( \tilde{F} \) with disturbed input \( \tilde{x}_u \). The disturbed model is often realized by adding dropout layers [23], [36] into the model structure, or injecting random noises into the feature maps [28], [36]. And the disturbed input is usually realized by data augmentations [6], [13], [54], [60]. The \( S \) is the number of pixels in image \( x_u \) and \( C \) is the number of categories, and \( y_{us} \) and \( z_{us}^c \) are the elements of \( \tilde{y}_u \) and \( z_u \) for the \( c \)-th class of the \( s \)-th pixel. We name this loss in Eq. 2 as optimistic consistency loss \( L_u^o \) since it optimistically uses the pseudo label as the target.

We see the \( L_u^o \) is the average of the losses computed on each pixel. Hence, for simplicity, we next discuss our method using a single pixel as an example. Without loss of generality, our method works on all pixels. By the definition of \( L_u^o \) in Eq. 2 we compute its gradient for the \( s \)-th pixel in backpropagation:

\[
\frac{\partial L_{us}}{\partial z_{us}^c} = \begin{cases} p_{us}^c - 1, & \text{if } \tilde{y}_{us} = 1 \\ p_{us}^c, & \text{else} \end{cases} \quad \frac{\exp(z_{us}^c)}{\sum_{n=1}^{C} \exp(z_{us}^n)}
\]

\[
p_{us}^c = \frac{\exp(z_{us}^c)}{\sum_{n=1}^{C} \exp(z_{us}^n)},
\]

where the \( p_{us}^c \) is the predicted probability for the \( c \)-th class computed by softmax.

According to the gradient descent algorithm, the output \( z_{us}^c \) for a category \( c \) will increase if its gradient is less than 0, and vice versa. In other words, only the output for the pseudo label category \( (\tilde{y}_{us} = 1) \) is optimized to increase, and the outputs for other categories \( (\tilde{y}_{us} = 0) \) are optimized to decrease. This shows that when the pseudo label is correct, \( L_u^o \) increases the output for the GT category and decreases the outputs for other categories, thus effectively learning unlabeled data. However, once the pseudo label is wrong, it increases the output for the pseudo label category while decreasing the output for the GT category, which leads to learning unlabeled data incorrectly.

The limitation of the optimistic loss comes from that it optimistically assumes the pseudo label is correct. In practice, models often confront a pessimistic situation, that is, the pseudo label is not always correct. To address this issue, we propose pessimistic consistency regularization (PCR) which assumes that the GT category exists in a provided candidate label set that contains \( K \) candidate labels including the top-1 pseudo label. Compared with the existing optimistic assumption, our assumption is more pessimistic and prone to occur. This is because when the pseudo label is correct, both two assumptions hold. When the pseudo label is wrong, the optimistic assumption fails, but our assumption may still hold as long as the GT label appears in our candidate label set.

B. Pessimistic Consistency Loss

In our PCR, we generate a candidate label set \( Y_{us} = \{y_{us,1}, y_{us,2}, ..., y_{us,K}\} \) which contains \( K \) label proposals for each unlabeled pixel instead of only one pseudo label as in previous works. Hence we propose a new loss function to learn the possible GT label from \( Y_{us} \).

Different from optimistic methods which assume the top-1 pseudo label is correct, our PCR pessimistically thinks each category in the candidate label set \( Y_{us} \) has the probability to be the GT category. But the categories outside the candidate set cannot be the GT category, which is in line with optimistic methods. Hence, we hope that the predictions of our model for the \( K \) categories in the candidate label set are larger than the predictions for the rest \( C-K \) categories. We refer to some works in metric learning [27], [40], [44], [45] and formulate our optimization objective for each pixel as:

\[
\min_{i \in Y_{us}} (z_{i_{us}}^c) > \max_{j \not\in Y_{us}} (z_{j_{us}}^c), \quad (4)
\]

where \( z_{i_{us}}^c \) represents the output of our model for the \( i \)-th category. Eq. 4 means we force the minimum of the outputs for categories in \( Y_{us} \) to be larger than the maximum of the outputs for other categories. In other words, we enforce all the outputs for categories in \( Y_{us} \) to be larger than those for other categories. From Eq. 4 a straightforward loss function is formulated as:

\[
L_u^p = ReLU(\max_{j \not\in Y_{us}} (z_{j_{us}}^c) - \min_{i \in Y_{us}} (z_{i_{us}}^c)). \quad (5)
\]

However, this \( L_u^p \) is globally non-differentiable with respect to \( z_{us} = \{z_{us,1}, z_{us,2}, ..., z_{us,K}\} \) since the max and min functions in Eq. 5 are globally non-differentiable [29], [37]. And the
Based on these functional approximations, our pessimistic function also has a singularity at \( x = 0 \). Thanks to some existing functional approximation \([8], [10], [29], [34]\), we approximate the functions in Eq. [5] to make \( \mathcal{L}_p^{u_s} \) differentiable:

\[
\begin{align*}
\max(z^1, z^2, \ldots, z^n) & \approx \log(\sum_{i=1}^{n} \exp(z^i)) \\
\min(z^1, z^2, \ldots, z^n) & \approx -\log(\sum_{i=1}^{n} \exp(-z^i)) \\
\text{ReLU}(z) = \max(z, 0) & \approx \log(1 + \exp(z)).
\end{align*}
\]

(6)

Next, we analyze the behavior of \( \mathcal{L}_p^{u_s} \) in backpropagation. For simplicity, we denote the categories that belong to \( \mathcal{Y}_{u_s} \) as positive categories, and the categories that do not belong to \( \mathcal{Y}_{u_s} \) as negative categories. In backpropagation, we compute the gradient of \( \mathcal{L}_p^{u_s} \) with respect to the output \( z_{u_s} \) of our model:

\[
\begin{align*}
\frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} & = \frac{-\sum_{j \notin \mathcal{Y}_{u_s}} e^{z_{u_s}} \times e^{-z_{u_s}}}{1 + \sum_{j \notin \mathcal{Y}_{u_s}} e^{z_{u_s}} \times \sum_{i \in \mathcal{Y}_{u_s}} e^{-z_{u_s}}}, \quad i \in \mathcal{Y}_{u_s} \\
\frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} & = \frac{\sum_{j \notin \mathcal{Y}_{u_s}} e^{-z_{u_s}} \times e^{j}}{1 + \sum_{j \notin \mathcal{Y}_{u_s}} e^{z_{u_s}} \times \sum_{i \in \mathcal{Y}_{u_s}} e^{-z_{u_s}}}, \quad j \notin \mathcal{Y}_{u_s},
\end{align*}
\]

(8)

where the \( \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} \) and \( \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} \) denote the derivatives with respect to outputs for positive categories and negative categories, respectively. From Eq. [8] we see that our \( \mathcal{L}_p^{u_s} \) has following characteristics:

1. The output for the GT category increases when it appears in \( \mathcal{Y}_{u_s} \). This is because outputs for positive categories have negative gradients and are optimized to increase by gradient descent.

2. The existing \( \mathcal{L}_p^{u_s} \) is a special case of our \( \mathcal{L}_p^{u_s} \) when we set \( K = 1 \), as shown in Eq. [9]

\[
\mathcal{L}_p^{u_s} = -\log \left( \frac{e^{z_{u_s}}}{e^{z_{u_s}} + \sum_{j \neq i} e^{z_{u_s}}} \right) = \log(1 + e^{-z_{u_s}} \times \sum_{j \neq i} e^{z_{u_s}}),
\]

(9)

where \( i \) is the index of top-1 predicted pseudo label.

3. The gradients of \( \mathcal{L}_p^{u_s} \) are balanced, which is reflected in that the absolute value of the sum on positive gradients \( \sum_{i \in \mathcal{Y}_{u_s}} \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} \) is equal to the sum on negative gradients \( \sum_{j \notin \mathcal{Y}_{u_s}} \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} \), as Eq. [10]

\[
\left| \sum_{i \in \mathcal{Y}_{u_s}} \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} \right| = \sum_{j \notin \mathcal{Y}_{u_s}} \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} = \sum_{j \notin \mathcal{Y}_{u_s}} e^{z_{u_s}} \times \sum_{i \in \mathcal{Y}_{u_s}} e^{-z_{u_s}} = \frac{1}{1 + \sum_{j \notin \mathcal{Y}_{u_s}} e^{z_{u_s}} \times \sum_{i \in \mathcal{Y}_{u_s}} e^{-z_{u_s}}}.
\]

(10)

\section{C. Candidate Label Proposal}

In this section, we introduce how we adaptively provide the candidate label proposals for each pixel. Note that the number of proposals is important since too many labels will make our model predict multiple categories for one pixel, while too few labels (i.e. \( K = 1 \)) will make our model degenerate into the existing optimistic method.

We first propose to use the categories with top-K predicted probabilities as candidate labels since high-confidence predictions are prone to be correct \([3]\). We then design an easy but effective K value selection strategy to adaptively determine the K value for each pixel, as shown in Algorithm 1. In particular, we set a hyperparameter \( T \) which represents the upper bound of cumulative probability. For one pixel, we then compute the cumulative probability of its top-n predictions and record the value of \( n \) where the cumulative probability exceeds \( T \) for the first time. Finally, the K value for this pixel is set as \( \max(n - 1, 1) \).

Selecting the top-n predictions whose cumulative probability exceeds \( T \) guarantees that the GT category has a high probability of being selected. An counter-intuitive design in our Algorithm 1 is that we choose \( K = n - 1 \) instead of \( K = n \). This is because setting \( K = n - 1 \) alleviates the problem of gradient vanishing in training. To explain this, we convert the gradients of \( \mathcal{L}_p^{u_s} \) in Eq. [8] to probabilistic form:

\[
\begin{align*}
\frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} & = \frac{-\sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \times \frac{1}{p^{i}_{u_s}}}{1 + \sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \times \sum_{i \in \mathcal{Y}_{u_s}} \frac{1}{p^{i}_{u_s}}} \times \frac{1}{p^{i}_{u_s}}, \\
\frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} & = \frac{\sum_{i \in \mathcal{Y}_{u_s}} \frac{1}{p^{i}_{u_s}}}{1 + \sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \times \sum_{i \in \mathcal{Y}_{u_s}} \frac{1}{p^{i}_{u_s}}} \times \frac{1}{p^{i}_{u_s}},
\end{align*}
\]

(11)

Here we only need to analyze the gradients of positive categories, because characteristic (3) in Eq. [10] shows that the absolute value of the sum on positive gradients \( \sum_{j \notin \mathcal{Y}_{u_s}} \frac{\partial \mathcal{L}_p^{u_s}}{\partial z_{u_s}} \) is close to 0 when its numerator (i.e. \( \sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \)) is close to 0. If we choose \( K_{u_s} = n \), then the range of the cumulative probability sum \( \sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \) is \((T, 1)\), correspondingly, the range of \( \sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \) will be \((0, 1 - T)\). This means that it is possible for \( \sum_{j \notin \mathcal{Y}_{u_s}} p^{j}_{u_s} \) to approach 0 causing the problem.

\begin{algorithm}[H]
\caption{K value selection strategy}
\begin{algorithmic}
\Require{sorted prediction \( p = (p^1, p^2, \ldots, p^C) \)}
\Ensure{K value}
\Initialize{cumulative probability upper bound \( T \), category numbers \( C \), cumulative probability vector \( V = p \)}
\Compute{cumulative probability:}
\For{\( n = 1 \) to \( C \)}
\If{\( V^n > T \) or \( n = C \)}
\State{\Return{\( n \)}}
\EndIf
\EndFor
\Determine{K value:}
\State{\( K = \max(n - 1, 1) \)}
\State{\Return{K}}
\end{algorithmic}
\end{algorithm}
of gradient vanishing. When setting $K_{us} = n - 1$, we derive $\sum_{j \notin Y_{us}} p_{us}^j > 1 - T$, which provides a lower bound for the numerator of $\frac{\partial p_{us}^o}{\partial z_{us}}$. Hence, we alleviate the problem of gradient vanishing.

D. Adaptive weight for each pixel

Our PCR is implemented based on the assumption that the GT category exists in the candidate label set $Y_{us}$. Here we propose to integrate the confidence of our assumption into the training of our method. When our assumption is not tenable, the GT label will appear in the negative categories of which the largest predicted probability is $\max_{j \notin Y_{us}} (p_{us}^j)$. Hence, the $\max_{j \notin Y_{us}} (p_{us}^j)$ could reflect the confidence of our assumption. Specifically, $\max_{j \notin Y_{us}} (p_{us}^j)$ is negatively correlated with assumption confidence since high $\max_{j \notin Y_{us}} (p_{us}^j)$ means that GT label has a high probability outside $Y_{us}$, and vice versa.

Formally, the range of $\max_{j \notin Y_{us}} (p_{us}^j)$ is derived as:

$$\frac{1 - T}{C - K_{us}} < \max_{j \notin Y_{us}} (p_{us}^j) < \frac{\sum_{i \in Y_{us}} p_{us}^i}{K_{us}}.$$  \hspace{1cm} (12)

In practice, $T$ is close to 1 (e.g. 0.9), thus $\frac{1 - T}{C - K_{us}}$ is close to 0. For simplicity, we obtain the approximate range of $\max_{j \notin Y_{us}} (p_{us}^j)$ as $\left(0, \frac{\sum_{i \in Y_{us}} p_{us}^i}{K_{us}}\right)$. We then define our adaptive weight as a monotonically decreasing concave function as:

$$w_{us} = \frac{\log [1 + A \times \left(\frac{\sum_{i \in Y_{us}} p_{us}^i - \max_{j \notin Y_{us}} (p_{us}^j)}{K_{us}}\right)]}{\log [1 + A \times \left(\frac{\sum_{i \in Y_{us}} p_{us}^i}{K_{us}}\right)]},$$  \hspace{1cm} (13)

where $A$ is a scalar used to control the Radon of this function, which is fixed as 50 in the experiments. It is worth noting that our adaptive weight is different from the weights computed by top-1 confidence used to filter out low-confidence pixels [9], [21], [36]. Those weights are small for pixels with low top-1 probability, which results in those pixels not being sufficiently used in training [47]. But our weight is only small when the prediction of a pixel is confused in the top-(K+1) categories, thus our model still uses the information that its prediction should not belong to the other C-K-1 categories.

E. Overall training

As discussed above, our unsupervised loss is the weighted pessimistic consistency loss, as follows:

$$\mathcal{L}_{uns} = \frac{1}{|U||S|} \sum_{u=1}^{U} \sum_{s=1}^{S} w_{us} \times \mathcal{L}_{us},$$  \hspace{1cm} (14)

where the $w_{us}$ is the adaptive weight described in Section III-D. For labeled pixels, our goal is to minimize the standard cross-entropy loss, which is the same as existing works [6], [9], [13], [21], [36], [42], [43]. The only optimization object of our method is to minimize the ‘supervised loss + pessimistic consistency loss’, as follows:

$$\mathcal{L} = \mathcal{L}_{sup} + \beta \mathcal{L}_{uns}, \quad \mathcal{L}_{sup} = \frac{1}{|L||S|} \sum_{l=1}^{L} \sum_{s=1}^{S} \mathcal{L}_{CE}(p_{ls}, y_{ls})$$  \hspace{1cm} (15)

where $\mathcal{L}_{CE}$ is the cross-entropy loss function, $\beta$ is the regularization weight, $p_{ls}$ and $y_{ls}$ are the prediction and groundtruth of the s-th pixel in l-th labeled image.

IV. Why PCR works?

In this section, we provide an analysis of why pessimistic consistency regularization could improve the learning of unlabeled data, especially for mispredicted pixels.

A. The perspective of gradient ratio

In the SSL segmentation task, we always hope to increase the output for the GT category (named GT output in the following text) to provide a correct prediction. But in practice, previous optimistic methods only increase the output for the pseudo label category and decrease the outputs for all other categories, as derived in Section III-A. For our PCR, we could only increase the outputs for positive categories (i.e. categories in the candidate label set) and decrease the outputs for the negative categories by optimizing our pessimistic consistency loss, as analyzed in Section III-B. Therefore, we propose a gradient ratio $R$ to measure how properly the GT output is optimized. The gradient ratio for the s-th pixel in u-th unlabeled image is computed as:

$$R_{us} = \frac{\partial \mathcal{L}_{us}}{\partial z_{us}} / \sum_{i \in Y_{us}} \frac{\partial \mathcal{L}_{us}}{\partial z_{us}}_i,$$  \hspace{1cm} (16)

where $Y_{us}$ represents the candidate label set $Y_{us}$ when $\mathcal{L}_{us}$ is $\mathcal{L}_{pse}$ and $Y_{us}$ represents the pseudo label when $\mathcal{L}_{us}$ is $\mathcal{L}_{us}$. The gradient ratio $R_{us}$ is the ratio of the gradient for the GT category to the sum of the gradients for the positive categories. It ranges from [-1,1] and a positive $R_{us}$ means the GT output is optimized to increase, while a negative $R_{us}$ means the GT output is incorrectly optimized to decrease. Based on the prediction of each pixel, we consider $R_{us}$ in three cases:

**Case 1.** The pseudo label is correct, that is, the GT category is the top-1 prediction. In this case, the gradient ratio $R_{us}$ computed by $\mathcal{L}_{o}$ is $\mathcal{L}_{uns}$ are:

$$R_{o}^{o} = \frac{p_{gt}^{o} - 1}{p_{pse}^{o} - 1} = 1, \quad R_{o}^{u} = \frac{e^{-z_{us}^{o}}}{\sum_{l \in Y_{us}} e^{-z_{us}^{l}}} \in [0,1],$$  \hspace{1cm} (17)

where $p_{gt}^{o}$ and $p_{pse}^{o}$ are the predicted probability for GT category and pseudo label category. We see that $R_{o}^{u}$ and $R_{o}^{o}$ are both greater than 0. This means they both encourage the GT output to increase in this case.

**Case 2.** The top-1 prediction is wrong, but the GT category is in the categories with top-K probabilities, where K is computed by our K value selection strategy in Algorithm 1. That is, the pseudo label is wrong but the GT label is in our candidate label set $Y_{us}$. For case 2, the gradient ratio $R_{o}^{o}$ and $R_{o}^{u}$ are computed as:

$$R_{o}^{o} = \frac{p_{gt}^{o}}{p_{pse}^{o} - 1} \in [-1,0], \quad R_{o}^{u} = \frac{e^{-z_{us}^{o}}}{\sum_{l \in Y_{us}} e^{-z_{us}^{l}}} \in [0,1],$$  \hspace{1cm} (18)

We see that $R_{o}^{u}$ is larger than 0 while $R_{o}^{o}$ is less than 0. This means that $\mathcal{L}_{o}$ erroneously reduces GT output, but our
It is obvious that the cosine similarity between the gradient vector of the existing optimistic baseline and the unavailable GT label with cross-entropy loss as the ideal direction in most minibatchs. Hence we compute \( \text{sim}(G^p, G^{ide}) \) and \( \text{sim}(G^o, G^{ide}) \) quantitatively in Fig 3 (a). We additionally present \( \text{sim}(G^+, G^{ide}) \) and \( \text{sim}(G_{ide}, G^{ide}) \) in Fig 3 (b) since the \( G^{ide}, G^o \) and \( G^p \) are collinear, thus only the difference between \( G^+ \), \( G^o \) and \( G^p \) matters training.

As shown in Fig. 3 (a), we first observe that the positive rates of \( \text{sim}(G^p, G^{ide}) \) are more than 90% in all minibatches, which indicates that \( G^p \) makes our model go further in the ideal direction in most minibatches. Second, we see that the \( \text{sim}(G^p, G^{ide}) \) is greater than the \( \text{sim}(G^o, G^{ide}) \), which means our pessimistic gradient \( G^p \) is closer to the ideal gradient \( G^{ide} \) than the gradient obtained by existing optimistic method \( G^o \). Fig. 3 (b) has a similar trend to Fig. 3 (a) but lower values, because we remove the effect of collinear gradients \( G^{ide}, G^p \) and \( G^o \). These figures show that, in Case 2, PCR is beneficial (cosine similarly larger than 0) in most mini-batches and outperforms the existing optimistic method.

It is worth noting that \( \text{sim}(G^p, G^{ide}) \) only reflects that the angle between our gradient and the ideal gradient is a mostly acute angle. But the norms of \( G^p \) and \( G^{ide} \) also affects optimization of our model. If the norm of \( G^p \) is much larger than that of \( G^{ide} \), it will cause \( G^p \) over-optimize our model, hence even if their angle is small, it will also be detrimental to optimization. We prove that the norms of \( G^p \) and \( G^{ide} \) are both range of \([0, \sqrt{2}]\) in Proof A. In addition, we also count their norms in Figure 4. We see that the two norms are close and the norm of \( G^p \) is smaller than that of \( G^{ide} \), which means that our PCR won’t bring the problem of over-optimization.

\[ R^o_{us} = \frac{p^o_{us}}{p^o_{us} \cdot 1} \in [-1, 0], \quad R_p = -\frac{\sum_{j \in \mathbb{V}_{us}} e^{z_{us}^j}}{\sum_{j \in \mathbb{V}_{us}} e^{z_{us}^j}} \in [-1, 0]. \]  

(19)

It is obvious that \( R^o_{us} \) and \( R^p_{us} \) are both less than 0. This means that neither \( L^o_{us} \) nor \( L^p_{us} \) is beneficial for training the model to predict the GT category in this case. We plot the gradient ratio \( R \) for pixels in these three cases with our experiments on VOC2012 with 1/8 labeled data using the CPS [6] as the baseline. As shown in Figure 2, \( L^o_{us} \) significantly outperforms the existing \( L^o_{us} \) in Case 2, and they perform similarly in Case 1 and Case 3.

B. The perspective of gradient similarity

In Case 2 above, though \( L^p_{us} \) encourages the output for the GT category to increase which is better than \( L^o_{us} \), it also encourages the outputs for other positive categories to become larger. Ideally, the cross-entropy loss using the unavailable GT label only increases the GT output and decreases the outputs for all other categories. We name the gradient computed using the unavailable GT label with cross-entropy loss as the ideal gradient \( G^{ide} \).

Here, we propose to use the cosine similarity between the ideal gradient vector \( G^{ide} \) and pessimistic gradient vector \( G^p \) brought by \( L^p_{us} \) to further analyze our PCR in Case 2. If the cosine similarity is greater than 0, it means the projection of \( G^p \) on \( G^{ide} \) is positive, which indicates \( G^p \) is beneficial to our model since it makes our model go further in the ideal direction. For comparison, we also calculated the cosine similarity between the gradient vector of the existing optimistic method \( G^o \) and the ideal gradient \( G^{ide} \).

Based on Eq. [3] and Eq. [11], we then compute the three gradient vectors \( G^o, G^p \) and \( G^{ide} \) as:

\[ G^{ide} = (p^1, p^2, ..., p^K - 1, p^K, 0, ..., 0) \]
\[ G^o = (p^1 - 1, p^2, ..., p^K, 0, ..., 0) \]
\[ G^p = C_p \cdot \left( \frac{1}{p^1}, ..., \frac{1}{p^K}, 0, ..., 0 \right) \]
\[ + C_o \cdot (0, ..., 0, p^{K+1}, ..., p^C) \]

where \( C_p \) and \( C_o \) are the first terms of the two expressions in Eq. [11]. The \( G^{ide}_+, G^o_+ \) and \( G^p_+ \) are gradients for positive categories, while \( G^{ide}_-, G^o_- \) and \( G^-p \) are gradients for negative categories. Due to the complexity of predicted probability, the relationship between the cosine similarity \( \text{sim}(G^p, G^{ide}) = \frac{G^p \cdot G^{ide}}{|G^p| |G^{ide}|} \) and 0 is not mathematically absolute. Hence we compute \( \text{sim}(G^p, G^{ide}) \) and \( \text{sim}(G^o, G^{ide}) \) quantitatively in Fig 3 (a). We additionally present \( \text{sim}(G^+, G^{ide}) \) and \( \text{sim}(G^{ide}, G^{ide}) \) in Fig 3 (b) since the \( G^{ide}, G^o \) and \( G^p \) are collinear, thus only the difference between \( G^+ \), \( G^o \) and \( G^p \) matters training.

V. Experiments

A. Implementation details

Frameworks and Dataset: We evaluate the effectiveness of our PCR on two SOTA frameworks CPS [6] and AEL [13], and two data sets PASCAL VOC 2012 and Cityscapes. The CPS [6] is an easy but effective SSL segmentation framework, which have been verified effective in training both with and without CutMix [55] data augmentation. The AEL [13] is a strong framework which considers the pseudo-label imbalance problem in SSL segmentation.
Fig. 3: (a) is the cosine similarity between ideal gradient $G^{ide}$, pessimistic gradient $G^p$, and optimistic gradient $G^o$. Also we present the positive rate of the similarity between $G^{ide}$ and $G^p$ computed on pixels in each minibatch. (b) has a similar meaning to (a), but computes the similarity and positive rate only using the gradients for positive categories ($G^{ide}_+, G^p_+, G^o_+$). The lines in (a) and (b) represent the mean value and the shading represents the standard deviation. (a) and (b) are all computed on VOC2012 with 1/8 labeled data using the CPS [6] framework.

Fig. 4: The norms of $G^{ide}$ and $G^p$ computed on VOC2012 with 1/8 labeled data using the CPS [6] framework.

Cityscapes is a large-scale data set designed for urban street scene segmentation which consists of 19 semantic classes containing 2,975 images for training, 500 for validation, and 1,525 for testing. The PASCAL VOC 2012 is a generic object segmentation benchmark that consists of 20 object classes and 1 background class. It is divided into training, validation, and test sets including 1,464, 1,449, and 1,456 images, respectively. And there is also an augmented set [11] adding 10,582 images into the standard training set. Following the setting of previous works [6], [60], we implement two splits on VOC2012: the standard split (with augmented set) and the low data split (without augmented set).

**Experimental Setting:** Our experiments are implemented on the official code of CPS and AEL using Deeplab v3+ with pre-trained ResNet-101 as backbones. Specifically, on Cityscapes using CPS as the baseline, we use SGD optimizer with a weight decay of 1e-4. The initial learning rate is set to 0.02 and the momentum is fixed at 0.9. We use the default ‘poly’ learning rate decay policy to scale the learning rate by $(1 - \text{iter}/\text{max iter})^{0.9}$, and this policy is used in all our experiments. The input images are cropped to 800 x 800 and we train our model for 160 epochs with a batchsize of 64. The cumulative probability upperbound $T$ are set as 0.95 for models w/o cutmix and 0.9, 0.85, 0.85 for models w/ cutmix on the data splits of 1/32, 1/16, 1/8, respectively. When using AEL as baseline, the batchsize, learning rate, epoch, and image size are changed to 16, 0.01, 150, and 769. The cumulative probability upperbound $T$ are set as 0.9, 0.9, 0.85, 0.85 on the data splits of 1/32, 1/16, 1/8, respectively. On VOC2012 using CPS as the baseline, we use SGD optimizer with a weight decay of 1e-4. The initial learning rate is set to 0.01 and the momentum is fixed at 0.9. The input images are cropped to 512 x 512 and we train our model for 40 epochs with a batchsize of 32. The cumulative probability upperbound $T$ are set as 0.95 for the model w/o cutmix on 1/16 data split, and 0.9 for all other models. When using AEL as the baseline, the batchsize and epoch are changed to 16 and 50. And $T$ are set as 0.95, 0.9, 0.9 on the data splits of 1/16, 1/8, 1/4, respectively. On VOC2012 LowData, $T$ are set as 0.95, 0.85, 0.85, 0.85 on the data splits of 1/16, 1/8, 1/4 and 1/2 labeled data, respectively. And other hyperparameters are the same as used on standard VOC2012.

**B. Quantitative results**

In our experiments, the PCR model is trained using the same hyperparameters as the baseline model, only replacing the unsupervised loss with our weighted pessimistic consistency loss. The segmentation results on VOC2012, Cityscapes, and VOC2012 LowData are presented in Table I, Table II, and Table III, respectively. We see that PCR improves the CPS baseline under both with and without CutMix settings, this indicates that the improvement brought by PCR and the improvement brought by data augmentation (i.e., CutMix) can be accumulated. In addition, we see PCR is effective on various baselines, i.e., CPS and AEL, which means PCR is universal for various existing SSL frameworks.

**C. Ablation Study**

1) Adaptive Weight: In section III-D, we propose an adaptive weight function to weigh the pessimistic loss of each pixel. In our discussion in section III-D, we show that the adaptive weight function should be inversely proportional to

---

1https://github.com/charlesCXK/TorchSemiSeg.git
2https://github.com/hzhupku/SemiSeg-AEL.git
Fig. 5: The segmentation results obtained on the VOC2012 dataset with 1/16 labeled data. For clarity, we show groundtruth (GT) in the form of instance label, and the predictions of CPS and CPS+PCR are presented in the form of semantic label.

TABLE I: The mIoU on VOC2012. The results marked by † are reported in previous works [6], [9], [22].

| Method                      | 1/16  | 1/8    | 1/4    |
|-----------------------------|-------|--------|--------|
| MT† [42]                    | 70.59 | 73.20  | 76.62  |
| CCT† [46]                   | 67.94 | 73.00  | 76.17  |
| CutMix-Seg† [9]             | 72.56 | 72.69  | 74.25  |
| GCT† [21]                   | 69.77 | 73.30  | 72.25  |
| GuidedMix-Net† [43]         | -     | 73.40  | 75.50  |
| CAC† [22]                   | 72.40 | 74.60  | 76.30  |
| CPS w/o cutmix [6]          | 72.50 | 74.97  | 77.14  |
| CPS w/ cutmix [6]           | 74.48 | 76.44  | 77.68  |
| AEL [13]                    | 74.20 | 76.58  | 77.98  |
| CPS+PCR w/o cutmix          | 73.18 (+0.68) | 75.74 (+0.77) | 77.34 (+0.33) |
| CPS+PCR w/ cutmix           | 74.98 (+0.80) | 77.75 (+1.31) | 78.30 (+0.62) |
| AEL+PCR                     | 74.98 (+0.78) | 76.73 (+0.15) | 78.35 (+0.37) |

TABLE II: The mIoU on Cityscapes. The results marked by † are reported in previous works [6], [22], [47].

| Method                      | 1/32  | 1/16  | 1/8    |
|-----------------------------|-------|-------|--------|
| MT† [42]                    | -     | 68.08 | 73.71  |
| CCT† [46]                   | -     | 69.64 | 74.48  |
| GCT† [21]                   | -     | 66.90 | 72.96  |
| U2PL† [47]                  | -     | 74.90 | 76.48  |
| CPS w/o cutmix [6]          | 59.70 | 71.22 | 74.98  |
| CPS w/ cutmix [6]           | 72.51 | 74.72 | 77.62  |
| AEL [13]                    | 73.00 | 75.26 | 78.07  |
| CPS+PCR w/o cutmix          | 61.00 (+1.30) | 72.05 (+0.83) | 75.69 (+0.69) |
| CPS+PCR w/ cutmix           | 73.20 (+0.69) | 75.01 (+0.29) | 78.47 (+0.85) |
| AEL+PCR                     | 75.01 (+2.01) | 76.58 (+1.32) | 78.19 (+0.12) |

The segmentation performances are shown in Figure 7, which are obtained on VOC2012 with 1/8 labeled data using CPS+PCR w/ CutMix as the training method. We think the superiority of the concave function comes from that, for small values of \( \max_{j \in Y_u} (p_{j|u}) \), concave function provides higher weight than convex and linear functions, demonstrating high utilization of high-confidence pixels.

2) The hyperparameter \( T \): The \( T \) is a key hyperparameter in our method, which controls how pessimistic our PCR is. Specifically, when \( T \) is low, the cumulative probability is easy
Fig. 6: Examples of convex, linear and concave decreasing functions. These examples are plotted in the setting of $T = 0.95$, $K = 2$, and cumulative probability $= T$.

Fig. 7: Performance of our CPS+PCR w/ CutMix model using different adaptive weights. The ‘baseline’ represents the baseline CPS model.

to exceed $T$, hence the K values for pixels are small. This means that the degree of pessimism in PCR is low at this time, and it is close to the existing optimistic methods. When $T$ is high, the cumulative probability is hard to exceed $T$, meaning the degree of pessimism in PCR is high. Here we summarize two rules for setting a proper $T$ value. First, a $T$ value around 0.9 (e.g. 0.85, 0.9, 0.95) is usually a promising setting. Second, the value of $T$ is negatively correlated to the number of labeled data.

The effect of $T$ on the training behaviors. In the training process, $T$ affects the number of candidate labels for each pixel, that is, the K value. In addition, $T$ also affects the impurity of the candidate label set, which is defined as the proportion of pixels that do not contain the GT label in the candidate set to all pixels. Statistically, we formulate the average $K$ value and the impurity as:

$$\bar{K} = \frac{1}{U \times S} \times \sum_{u=1}^{U} \sum_{s=1}^{S} K_{us}$$

$$\text{impurity} = \frac{1}{U \times S} \times \sum_{u=1}^{U} \sum_{s=1}^{S} \mathbb{I}(y_{us} \notin \mathbb{Y}_{us}),$$

(22)

where $K_{us}$, $y_{us}$, and $\mathbb{Y}_{us}$ are the K value, GT label, and candidate label set for the $s$-th pixel of the $u$-th image. The $U$ is the size of the unlabeled dataset, and $S$ is the number of pixels contained in each image.

In Table IV we provide the average $K$ value and impurity in the first two columns, and we also provide the average $K$ value and impurity computed on pixels of $K > 1$ in the last two columns. All results in this section are implemented on Cityscapes using ResNet 50 as backbone. We see that small $T$ values lead to small K values, meaning the size of the candidate label set is small. Accordingly, it causes high impurity of the candidate label set since a small candidate label set has a relatively large possibility of missing the GT label. High impurity brings noise to the learning process, which is known as confirmation bias [1]. In contrast, using a large value of $T$ builds a large candidate label set, effectively reducing the impurity. However, too large $T$ value (e.g. 0.99) makes our model learn from multiple labels, which is also not suitable for a single-label classification task. In Table V we present the mIoU of our PCR models trained with various $T$ values. Given our observation on the trade-off between the impurity and the size of the candidate label set, we find a $T$ value around 0.9 always provides promising results. In our experiments, the $T$ value is set from $\{0.85, 0.9, 0.95\}$.

| $T$ value | $\bar{K}$ | Impurity | $K_{>1}$ | Impurity$_{>1}$ |
|-----------|-----------|----------|----------|-----------------|
| 0.5       | 1.0005    | 0.2210   | 2.0258   | 0.6250          |
| 0.75      | 1.0179    | 0.2181   | 2.1940   | 0.4829          |
| 0.85      | 1.0499    | 0.2134   | 2.3167   | 0.3985          |
| 0.9       | 1.0974    | 0.2058   | 2.4166   | 0.3332          |
| 0.95      | 1.1932    | 0.1878   | 2.5755   | 0.2638          |
| 0.99      | 1.4959    | 0.1757   | 2.9208   | 0.1561          |

TABLE V: The performances of CPS+PCR models trained with various $T$

| $T$ value | mIoU |
|-----------|------|
| 0.5       | 68.80|
| 0.75      | 68.97|
| 0.85      | 69.34|
| 0.9       | 69.71|
| 0.95      | 69.08|
| 0.99      | 67.52|

The relationship between $T$ and the amount of labeled data. As the number of labeled data increases, the difficulty of the semi-supervised segmentation task decreases. We find a high $T$ usually obtains good performance when labeled data are limited, while a low $T$ usually performs better when labeled data are sufficient. As shown in Table VI in the 1/32 labeled data setting, $T = 0.95$ obtains the highest improvement about 1.37%, while $T = 0.85$ and $T = 0.9$
TABLE VI: The relationship between cumulative probability upperbound $T$ and the amount of labeled data. The results are obtained on Cityscapes using CPS as baseline.

| $T$   | $1/32$ | $1/16$ | $1/8$ |
|-------|--------|--------|-------|
| base  | 54.40  | 68.68  | 73.06 |
| 0.85  | 55.22 (+0.82) | 69.34 (+0.66) | 74.37 (+1.31) |
| 0.9   | 55.40 (+1.00) | 69.71 (+1.03) | 74.43 (+1.37) |
| 0.95  | 55.77 (+1.37) | 69.08 (+0.40) | 74.03 (+0.97) |

Fig. 8: This figure is plotted on the Cityscapes with $1/32$ labeled data using CPS+PCR w/ CutMix as training method. The input size is $800 \times 800$ which means there are 640000 pixels in total.

only obtain improvements about 1%. In the $1/16$ labeled data setting, $T = 0.9$ obtains the best performance which improves baseline by 1.03%, and the other two $T$ values improve baseline by about 0.5%. In the $1/8$ labeled data setting, $T = 0.9$ and $T = 0.85$ obtains close performances which improve baseline by 1.3%, while $T = 0.95$ performs not as good as the previous two $T$ settings. These empirical results show that setting the $T$ value negatively corresponding to the amount of labeled data in PCR could obtain promising performances.

D. K Values in Training

We record the number of pixels with different $K$ values during the training process in Figure 8. We see that the $K$ values of most pixels are 1. During training, the number of pixels with $K > 1$ decreases, and the number of pixels with $K = 1$ increases. This shows that with training, our $K$ value selection strategy provides $K$ value of 1 for most pixels and $K > 1$ for a few uncertain pixels. At the late stage of training, the $K$ values for more than 93.75% (i.e. 600000 / 640000) pixels are 1, which means our model output low-entropy predictions in the late stage of training.

In addition, we also illustrate the $K$ value map of some images during training in Figure 9. We see that the $K$ values of most pixels in the background are 1 since background pixels are usually easy to classify. In the early stage of training, the pixels with $K > 1$ are mainly located on objects, since the classification of objects for our model is uncertain at early training. As the training progresses, the number of pixels with $K > 1$ gradually decreases and these pixels are mainly distributed at the boundary of objects. This is because our model has certain predictions for most pixels in the later stage of training. But for pixels located at the object boundary, their categories are fuzzy, for which our model makes uncertain predictions for them. Our PCR provides multiple candidate labels (i.e. $K > 1$) for these uncertain pixels to learn, which is in line with their fuzzy property. From this figure, we find that the experimental behavior of $K$ value is consistent with the design of our Algorithm 1 that is, PCR provides small $K$ values for certainly predicted pixels and provides large $K$ values for uncertainly predicted pixels.

TABLE VII: Seconds per epoch. These statistics are measured using 8 Tesla V100 GPUs under the $1/8$ labeled data setting.

| Method | Cityscapes | VOC2012 |
|--------|------------|---------|
| AEL [13] | 7.30s | 8.35s |
| AEL+PCR | 8.20s | 9.85s |

VI. LIMITATIONS

Though PCR works well, it has the limitation of high time complexity. PCR alleviates the errors in wrong pseudo labels by letting the model learn the possible GT category that appears in the candidate label set, which requires assigning a $K$ value to each pixel. This leads to high time complexity for pessimistic consistency loss.

From Eq. 4 we see that the time complexity of computing $\mathcal{L}_p$ is $O(C)$ when the $Y_u$ is determined, where $C$ is the number of classes. For original $\mathcal{L}_a$, it is a special case of $\mathcal{L}_p$ when the $K$ is fixed to 1, hence the time complexity of original $\mathcal{L}_a$ for one pixel is $O(C)$. When it comes to $\mathcal{L}_p$, we additionally need to decide the $K$ value for each pixel of which the time complexity is $O(K)$ since it needs $K$ times additions and $K$ times comparisons. Hence, the time complexity of computing $\mathcal{L}_p$ is $O(KC)$ which is $K$ times of computing the original $\mathcal{L}_a$. We also quantitatively provide the time of training our PCR in practice. As shown in Table VII PCR brings about 10% additional training cost.

VII. CONCLUSION

In this paper, we propose a PCR that learns the unlabeled data in a pessimistic case where the pseudo label is not always correct. It provides $K$ candidate label proposals for each sample to learn the possible GT in them and could be easily integrated with existing SSL methods. Theoretical analysis and experiments show that PCR effectively outperforms the existing optimistic methods especially when the pseudo label is wrong but the GT category exists in our candidate label set. There are still some unexplored problems in PCR, e.g., “a more effective pessimistic consistency loss function” and “a better candidate label-proposal method to reduce the number of samples in Case 3.”
Fig. 9: The visualization of K values. This figure is plotted on the VOC2012 with 1/16 labeled data using CPS+PCR w/ CutMix as training method.

APPENDIX A

THE PROOF THAT THE NORMS OF $G_{ide}$ AND $G_p$ ARE BOTH RANGE OF $[0, \sqrt{2}]$.

To provide this, we need to draw on two other conclusions. First, the absolute value of the sum on gradients for positive categories $|\sum_{i \in Y_{us}} \frac{\partial C_{us}}{\partial z_{us}}|$ is equal to the sum on negative gradients $\sum_{j \notin Y_{us}} \frac{\partial C_{us}}{\partial z_{us}}$, which holds for both $L_{us}$ and $L_{o us}$, as shown in Eq. 10. Second, the sum on negative gradients $\sum_{j \notin Y_{us}} \frac{\partial L_{o us}}{\partial z_{us}}$ is less than 1, which has also been shown in Eq. 10 since the denominator is always 1 greater than the numerator. With the above two conclusions, we compute the norms of $G_{ide}$ and $G_p$ in Eq. 23 and Eq. 24.

$$|N_{ide}| = \sqrt{(g_1)^2 + (g_2)^2 + ... + (g_C)^2}$$

$$= \sqrt{(g_y)^2 + \sum_{i \neq y} (g_i)^2}$$

$$\leq \sqrt{2 \sum_{i \neq y} g_i^2} \leq \sqrt{2}$$

(23)

$$|N_p| = \sqrt{(g_1)^2 + (g_2)^2 + ... + (g_C)^2}$$

$$= \sqrt{\sum_{i \in Y} (g_i)^2 + \sum_{i \notin Y} (g_i)^2}$$

$$\leq \sqrt{\sum_{i \in Y} g_i^2 + \sum_{i \notin Y} g_i^2}$$

(24)

REFERENCES

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E O’Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In 2020 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2020.

[2] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. International Conference on Learning Representations, 2019.

[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. Advances in Neural Information Processing Systems, 2019.

[4] Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepcev. Poisson learning: Graph based semi-supervised learning at very low label rates. In International Conference on Machine Learning, pages 1306–1316, 2020.

[5] John Chen, Vatsal Shah, and Anastasios Kyrillidis. Negative sampling in semi-supervised learning. In International Conference on Machine Learning, pages 1704–1714, 2020.
ensemble transformations. IEEE Transactions on Image Processing, 30:1639–1647, 2021.

[47] Yuchao Wang, Haochen Wang, Yujun Shen, Jingjing Fei, Wei Li, Guoqiang Jin, Liwei Wu, Rui Zhao, and Xinyi Le. Semi-supervised semantic segmentation using unreliable pseudo-labels. arXiv preprint arXiv:2203.03884, 2022.

[48] Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 10857–10866, 2021.

[49] Qiizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. Advances in Neural Information Processing Systems, year=2019.

[50] Yutong Xie, Jianpeng Zhang, Zhibin Liao, Johan Verjans, Chunhua Shen, and Yong Xia. Intra- and inter-pair consistency for semi-supervised gland segmentation. IEEE Transactions on Image Processing, 31:894–905, 2022.

[51] Xin Xu, Manh Cuong Nguyen, Yasin Yazici, Kangkang Lu, Hlaing Min, and Chun-Sheng Foo. Semicurv: Semi-supervised curvilinear structure segmentation. IEEE Transactions on Image Processing, 31:5109–5120, 2022.

[52] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In International Conference on Machine Learning, pages 11525–11536, 2021.

[53] Rumeng Yi, Yaping Huang, Qingji Guan, Mengyang Pu, and Runsheng Zhang. Learning from pixel-level label noise: A new perspective for semi-supervised semantic segmentation. IEEE Transactions on Image Processing, 31:623–635, 2022.

[54] Jianlong Yuan, Yifan Liu, Chunhua Shen, Zhibin Wang, and Hao Li. A simple baseline for semi-supervised semantic segmentation with strong data augmentation. Proceedings of the IEEE International Conference on Computer Vision, 2021.

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision, pages 6023–6032, 2019.

[56] Xiangyun Zeng, Rian Huang, Yuming Zhong, Dong Sun, Chu Han, Di Lin, Dong Ni, and Yi Wang. Reciprocal learning for semi-supervised segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 352–361. Springer, 2021.

[57] Bowen Zhang, Yadong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinzaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. Advances in Neural Information Processing Systems, 34, 2021.

[58] Yuanyi Zhong, Bodi Yuan, Hong Wu, Zhiqiang Yuan, Jian Peng, and Yu-Xiong Wang. Pixel contrastive-consistent semi-supervised semantic segmentation. Proceedings of the IEEE International Conference on Computer Vision, 2021.

[59] Liansheng Zhuang, Zihan Zhou, Shenghua Gao, Jingwen Yin, Zhouchen Lin, and Yi Ma. Label information guided graph construction for semi-supervised learning. IEEE Transactions on Image Processing, 26(9):4182–4192, 2017.

[60] Yuliang Zou, Zizhao Zhang, Han Zhang, Chun-Liang Li, Xiao Bian, Jia-Bin Huang, and Tomas Pfister. Pseudoseg: Designing pseudo labels for semantic segmentation. In International Conference on Learning Representations, 2020.