Relieving pixel-wise labeling effort for pathology image segmentation with self-training

Romain Mormont[1], Mehdi Testouri[2], Raphaël Marée[1], and Pierre Geurts[1]

[1] University of Liège, Belgium
{r.mormont,raphael.maree,p.geurts}@uliege.be
[2] University of Luxembourg, Luxembourg
mehdi.testouri@uni.lu

Abstract. Data scarcity is a common issue when training deep learning models for digital pathology, as large exhaustively-annotated image datasets are difficult to obtain. In this paper, we propose a self-training based approach that can exploit both (few) exhaustively annotated images and (very) sparsely-annotated images to improve the training of deep learning models for image segmentation tasks. The approach is evaluated on three public and one in-house dataset, representing a diverse set of segmentation tasks in digital pathology. The experimental results show that self-training allows to bring significant model improvement by incorporating sparsely annotated images and proves to be a good strategy to relieve labeling effort in the digital pathology domain.

Keywords: Deep learning, image segmentation, self-training, data scarcity, digital pathology

1 Introduction

Computational pathology is on the brink of revolutionizing traditional pathology workflows by providing artificial intelligence-based tools that will help pathologists making their work faster and more accurate. However, the development of such tools is hindered by data scarcity as pathology training datasets are not as numerous and large as in other fields of research. This is particularly true for image segmentation as per-pixel delineation of objects is one of the most time-consuming types of annotation there is.

In this work, we investigate a method for training a binary image segmentation model in a context where the segmentation ground truth is sparse. More precisely, we focus on a setup where the training set is composed of an exhaustively labeled subset $\mathcal{D}_l$ and a sparsely labeled subset $\mathcal{D}_s$. In particular, the images in $\mathcal{D}_l$ come with exhaustively labeled segmentation masks (*i.e.* pixels of all objects of interest have been labeled as positive) whereas in $\mathcal{D}_s$, some (unknown) pixels belonging to objects of interest have not been labeled, hence the sparsity. Typically, image segmentation methods require that the training images come with exhaustive pixel-wise labels. In our setup, they would allow us to only use $\mathcal{D}_l$ and force us to ignore $\mathcal{D}_s$. We believe that it is possible to include the sparsely labeled images as well, and hopefully improve the performance over using only $\mathcal{D}_l$. One way of achieving this would be to somewhat "complete" the sparse segmentation masks and make them exhaustively labeled. Generating pseudo-labels is precisely what self-training approaches do, therefore, we devise an automated self-training workflow to exploit both sets.

Our self-training workflow consists of two separate phases. During the "*warm-up*" phase, we train a U-Net [23] model in a classical supervised manner on $\mathcal{D}_l$ for a few epochs. Then, during the "*self-training*" phase (sketched in Figure 1), we repeat the following process for an arbitrary number of epochs : pseudo labels are generated for the unlabeled pixels from images in $\mathcal{D}_s$ using the currently trained model and the pseudo-labeled images are included in the training set for the next epoch. To control the impact of model uncertainty on pseudo-labeled pixels, we furthermore study different weighting schemes to tune their contributions to the training loss. We use binary ("hard") pseudo-labels and propose an auto-calibration approach for generating them. Experiments are carried out on three exhaustively labeled public datasets, namely MoNuSeg [12], GlaS [26] and SegPC-2021 [7], on which sparsity is artificially enforced for evaluation purpose. Through these experiments, we investigate the interest of our method and the impact of its hyperparameters in different scarcity conditions and in comparison with different baselines. In a second stage, we apply our method on an actual sparsely labeled dataset for cytology diagnosis.

Our main contributions and conclusions are as follows: **(1)** We design a self-training pipeline for binary segmentation to handle datasets composed on both exhaustively and sparsely annotated images (Section 3). **(2)** We show on three public datasets that this self-training approach is beneficial as, even in significant scarcity conditions, it improves over using only exhaustively labeled data (see Section 6.1). **(3)** We confirm the interest of the approach in a real-world scenario, where a significant improvement of performance is achieved by exploiting sparse annotations (see Section 6.3). **(4)** We show that, at fixed annotation budget, it is not necessarily better to focus the annotation effort on exhaustive labels rather than sparse labels (see Section 6.2).

2 Related works

Self-training is not a new family of methods and has been applied more recently in the context of deep learning. A classical approach of a self-training process is the teacher-student paradigm where the teacher model is used to generate pseudo-labels that will be used to further train the student model. The most straightforward approach consists in using a single model alternatively as a teacher for pseudo-labeling and then as a student for the training step [33,32], but some approaches have used more complex interactions between the teacher and the student [30,22]. A critical design choice for a self-training algorithm is how it will make use of the pseudo-labels. Label uncertainty can be used to filter out the pseudo-labels not deemed reliable enough [6,15]. Coupled with aggressive

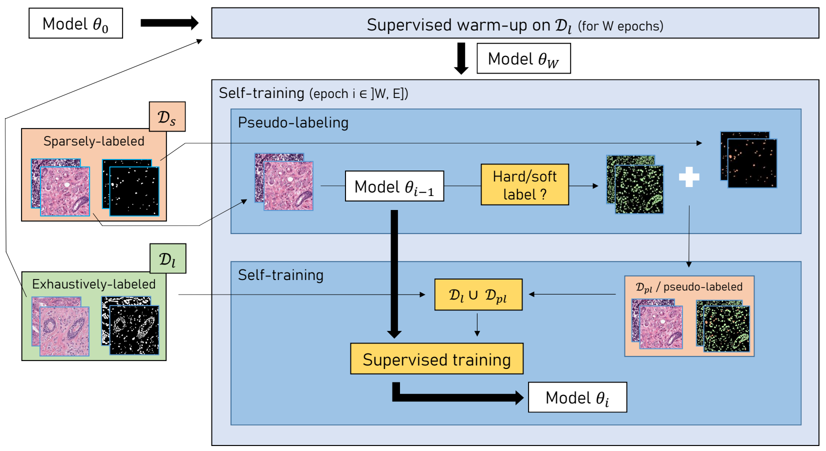


Fig. 1: Our approach illustrated. The model is first warmed-up on the exhaustively-labeled data then further traind by self-training on the combined sparsely- and exhaustively-labeled sets. A more formal definition is provided in Algorithm 1

data augmentation, consistency between pseudo-labels and model predictions is often formulated as a training loss [32,34,27,30].

It is not surprising that self-training has also been applied to medical image tasks to combat data scarcity [29,21] and to computational pathology in particular. Most works in this domain currently deal with image classification tasks [20,28,11,8,25] but detection and segmentation have also been explored. Jiayun Li *et al.* [17] combine weakly-supervised learning and self-training to predict per-pixel Gleason score across entire WSIs. Jiahui Li *et al.* [16] build a signet ring cell detector by training two models on pseudo-labels generated by one another. Segmentation architectures have also been trained with self-training for cardiac MRI [1] and COVID 19-related lung infection [5] segmentation. Bokhorst *et al.* [4] segment tissues from colorectal cancer patients into 13 classes representing different types of tissues. Their setup is similar to ours as their dataset is composed of both exhaustively- and sparsely-labeled images. They apply a weight map to tune and balance the contribution of individual pixels to the loss but they ignore unlabeled pixels entirely during training.

Beyond self-training, there exist methods, implemented in software such as QuPath [2], Ilastik [3] or Cytomine [19] and based on traditional computer vision or machine learning, that allow users to interactively complete a partial hand-drawn annotation of a given image. Among these methods are, for instance, Graph Cut [13] and GrabCut [24], or more recently DEXTRE [18] and NuClick [10]. While the self-training approach explored in this paper can certainly be exploited in an interactive mode, this question will be left as future work.

3 Methods

In the following section, we present our method, a self-training binary image segmentation algorithm. The self-training aspects and training implementation details are discussed separately in Sections 3.1 and 3.2 respectively.

We will denote by $\mathcal{D} = (X, Y) \subset \mathcal{X} \times \mathcal{Y}$ a segmentation dataset, where X and Y respectively represent a set of input images and their corresponding binary segmentation masks. We will further consider a training dataset composed of two sets: $\mathcal{D}_l = (X_l, Y_l) \subset \mathcal{X} \times \mathcal{Y}$, the exhaustively labeled set, and $\mathcal{D}_s = (X_s, Y_s) \subset \mathcal{X} \times \mathcal{Y}$, the sparsely labeled set. In $\mathcal{D}_l$, the masks Y_l are entirely determined, since the ground truth is known for all pixels (hence the exhaustiveness). In $\mathcal{D}_s$, ground truth is only partially known: given an image $\mathbf{x} \in X_s$, either a pixel x_{ij} belongs to a structure of interest in which case the mask pixel $y_{ij} = 1$, or it is not labeled in which case $y_{ij} = 0$ and no assumption can be made a priori about the fact that the pixel belongs to a structure of interest or not. We will denote $n_l = |\mathcal{D}_l|$ and $n_s = |\mathcal{D}_s|$ the sizes of the sets. The total number of training images for a dataset will be denoted $n = n_l + n_s$.

3.1 Self-training

Our self-training algorithm is described in Algorithm 1 (and depicted in Figure 1). It features a warm-up phase during which the model is classically trained on the set $\mathcal{D}_l$ (training implementation details are given in Section 3.2). The number of warm-up epochs $W > 0$ is fixed so that the model is able to converge on the labeled data. The warmed-up model is used as starting point for the self-training phase. Each self-training round e starts by pseudo-labeling $\mathcal{D}_s$ with the model θ_{e-1}. For an image $\mathbf{x} \in X_s$, the pseudo-label assigned to pixel x_{ij} is given by:

$$y_{ij}^{(pl)} = \begin{cases} 1, \text{ if } y_{ij} = 1 \\ g(\hat{y}_{ij}), \text{ otherwise} \end{cases} \tag{1}$$

where $\hat{y}_{ij}$ is the sigmoid output of model θ_{e-1} for pixel (i, j) given $\mathbf{x}$ as input and g is a function for generating the pseudo-label from $\hat{y}_{ij}$ (see below). In other words, we preserve the expert ground truth as pseudo-labels when available and use the model predictions for unlabeled pixels (this is the `Combine` step from Algorithm 1). With this strategy, entirely unlabeled images can also be included in $\mathcal{D}_s$. Our algorithm uses a single model (*i.e.* teacher = student) which is not reset between self-training rounds.

Soft and hard pseudo-labels. We considered two different pseudo-labeling strategies, or two different g functions (see Equation 1). Initially, we decided to simply take g to be the identity function $g(x) = x$ in which case the sigmoid output of the model was used as pseudo-label. This strategy is commonly called "*soft*" labeling. During the next self-training round, this soft label will be compared to the network prediction which can be seen as a form of consistency

Algorithm 1: Our self-training approach. The `Train` operation trains the given model on the provided dataset according to the protocol explained in Section 3.2. The `Predict` operation produces segmentation masks for a set of input images using the model. The `Combine` operation combines ground truth masks and pseudo labels from the given sets as explained in Section 3.1.

```
Data: The exhaustively- and sparsely labeled sets $\mathcal{D}_l$ and $\mathcal{D}_s$, a segmentation
      model $\theta_0$, $W$ and $E$ respectively the number of warm up epochs and
      the total number of epochs.
Result: A self-trained segmentation model $\theta_E$.
// Warm up
for $e \leftarrow 1$ to $W$ do
    $\theta_e$ = Train($\theta_{e-1}, \mathcal{D}_l$)
end
for $e \leftarrow W+1$ to $E$ do
    // Pseudo labeling
    $\hat{Y}_s$ = Predict($\theta_{e-1}, X_s$)
    $Y_{pl}$ = Combine($\hat{Y}_s, Y_s$)
    $\mathcal{D}_{pl} = (X_s, Y_{pl})$
    // Self-training
    $\theta_e$ = Train($\theta_{e-1}, \mathcal{D}_l \cup \mathcal{D}_{pl}$)
end
return $\theta_E$
```

constraint similar to those in [14,30,27]. However, early experiments have shown that this approach causes training instabilities. Therefore, we investigated a second strategy where the sigmoid output is binarized using a threshold $T_e \in [0, 1]$:

$$g(x) = \begin{cases} 1, \text{ if } x > T_e \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

where e is a self-training round. We call this strategy "*hard*" labeling as pseudo-labels are either 0 or 1. In addition to ensuring some sort of consistency between the pseudo-labels and the predictions, as in the "*soft*" approach, this thresholding also encourages the model to produce confident predictions (closer to 0 or 1). Because we want to avoid T_e to be an additional hyperparameter to tune, we propose an auto-calibration strategy based on the Dice score:

$$Dice_T(\mathbf{y}, \hat{\mathbf{y}}) = \frac{2 \times \sum_{i,j} \left[\mathbb{1}_{\hat{y}_{ij} \geq T} \times y_{ij}\right]}{\sum_{i,j} \mathbb{1}_{\hat{y}_{ij} \geq T} + \sum_{i,j} y_{ij}} \tag{3}$$

where T is the threshold applied to the model output to generate a binary prediction. The auto-calibration procedure selects T_e such that the Dice score in (3) is maximized for the images from an exhaustively labeled set $\mathcal{D}_a$:

$$T_e = \arg\max_T \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_a} \text{Dice}_T \left(\mathbf{y}, h(\mathbf{x}; \theta_e)\right). \tag{4}$$

The ideal choice for $\mathcal{D}_a$ would be to use an external validation set but, in extreme data scarcity conditions, extracting such a validation set would penalize the performance of the algorithm by removing a significant amount of training data. Therefore, in this context, we consider using the training subset $\mathcal{D}_l$ as $\mathcal{D}_a$. This approach has the advantage of not requiring additional training data but also induces a risk of overfitting which might hurt generalization performance. We hope that the overfitting problem would be compensated by the improvement brought by hard labeling.

3.2 Training

In this section, we will provide more information about the `Train` procedure from Algorithm 1 which trains a model θ with a dataset $\mathcal{D}$. We use U-Net [23] as a segmentation architecture. We set the initial number of feature maps to 8 instead of 64 in the original article, with the rest of the network scaled accordingly. The main goal of this reduction of model capacity is to limit overfitting given the highly scarce data conditions explored in this work, but it would be worth exploring more complex architectures as future work.

The number of rounds W and E and the number of training iteration per round are chosen independently per dataset. Every training iteration, we build a minibatch by sampling $B = 8$ images uniformly at random with replacement from $\mathcal{D}_l \cup \mathcal{D}_{pl}$ and by extracting one randomly located 512x512 patch and its corresponding mask from each of these images. The batch size was selected based on hardware memory constraints. We apply random data augmentation following best practices for machine learning in general and for self-training in particular [32,27]. We apply horizontal and vertical flips, color perturbation in the HED space [31] (bias and coefficient factors up to 2.5%), Gaussian noise (standard deviation up to 10%) and Gaussian blur (standard deviation up to 5%).

As a training loss, we average the per-pixel binary cross-entropy ℓ over all pixels of the batch, as defined in:

$$\ell(\hat{y}; y) = y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \tag{5}$$

$$\mathcal{L} = -\frac{1}{B} \sum_{b=1}^{B} \frac{1}{|\mathbf{y}_b|} \sum_{i} \sum_{j} w_{ij,b} \ell(\hat{y}_{ij,b}; y_{ij,b}) \tag{6}$$

We multiply the per-pixel loss by a weight $w_{ij,b}$ for pixel (i, j) of the b^{th} image of the batch in order to tune the contribution of this pixel to the loss (see below). We use Adam [9] as an optimizer with initial learning rate $\gamma = 0.001$ and default hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay).

Weighting schemes. Different strategies are evaluated for generating the per-pixel weight $w_{ij,b}$ in Equation 6. For the sake of simplicity, we will drop the batch identifier b in the subsequent equations and denote this weight by w_{ij}. We introduced this weight term to have the possibility to tune the contribution of pseudo-labeled pixels when computing the loss. It is important to note that this weight only applies to pseudo-labeled pixels and, therefore, the ground truth pixels will always be attributed a weight of $w_{ij} = 1$. It is also important to note that the weight is inserted as a constant in our loss and the weight function is not differentiated during back-propagation.

We study four different weighting strategies each producing an intermediate weight w_{ij}^{ξ} where ξ is the strategy identifier. Because we want to avoid the amplitude of the loss and gradients to be impacted by the weighting strategy, we normalize it to obtain the final weight $w_{ij} = w_{ij}^{\xi}/\overline{w}^{\xi}$, where $\overline{w}^{\xi}$ is the average weight over all pixels of a patch. Our weighting strategies are as follows.

- "*constant*": weight w_{ij}^{cst} given by hyperparameter constant $C \in \mathbb{R}^{+}$,
- "*entropy*": weight w_{ij}^{ent} given by the entropy of the model prediction,
- "*consistency*": weight w_{ij}^{cty} based on model predictions spatial consistency,
- "*merged*": weight w_{ij}^{mgd} computed by combining the two previous strategies.

These four strategies are detailed further in Supplementary Section B.

4 Data

In this section, we describe the datasets we use to evaluate our method. It includes three public exhaustively labeled segmentation datasets: MoNuSeg [12], GlaS [26] and SegPC-2021 [7]. The datasets are described in Table 1 and illustrated in Supplementary Section D. MoNuSeg contains images of tissues coming from different organs, patients and hospitals where epithelial and stromal nuclei are annotated. Given the variety of sources, the images exhibit significant variations of staining and morphology. The density of annotations is also quite high compared to the other datasets. GlaS features images containing both benign tissues and malignant tissues with colonic carcinomas. The gland annotations vary greatly in shape and size. Originally, SegPC-2021 contains 3 classes: background, cytoplasm and nucleus. In this work, we merge the two latter classes as we focus on binary segmentation. One of the challenges of this dataset is the presence of non-plasma cells which should be ignored by the algorithm although they are very similar to plasma cells. Moreover, artefacts are present on the images (*e.g.* cracked scanner glass in foreground, scale reference or magnification written on the image).

Additionally, we use a dataset that actually motivated the development of our method, a sparsely labeled dataset for thyroid nodule malignancy assessment. Pathologists[3] sparsely annotated nuclei and cell aggregates with polygon annotations in 85 whole-slide images on Cytomine [19]. The training set is a set of 4742 crops, one for each polygon annotation. The test set is a set of 45 regions of interest (2000x2000 pixels each) with annotations highlighting structures of interest (nuclei features and architectural cell patterns) made by a computer science student. This dataset is illustrated in Supplementary Section E.

[3] Isabelle Salmon's team from Erasme Hospital, Brussels, Belgium.

Table 1: Summary statistics for the four datasets used in this work. An image is a region of interest extracted from a whole-slide image, except for the training set of Thyroid FNAB where it is a crop of an annotation.

Dataset	Training set		Test set	
	Images	Annots	Images	Annots
MoNuSeg	30	17362	14	11484
GlaS	85	763	80	781
SegPC-2021	298	1643	300	900
Thyroid FNAB	4742		45	307

5 Experimental setup

In this section, we present context information for our experiments: what are our baselines, how we have simulated sparse datasets from the public datasets and what evaluation protocol we have applied.

Transforming the datasets. In order to fit the sparsely labeled settings described in Section 3, we generate new datasets from SegPC-2021, GlaS and MoNuSeg. This generation is controlled by two parameters: n_l and ρ. The former is the number of images to keep in the exhaustively labeled set $\mathcal{D}_l$. These images are chosen at random without replacement in the original training set. The latter parameter ρ is the percentage of annotations to remove from the images to make the remaining images sparsely labeled.

Baselines. We compare our self-training approach to three baselines. The first one, referred to as "*upper*", consists in using the full dataset, without removing any ground truth (*i.e.* $|\mathcal{D}_l| = n$ and $|\mathcal{D}_s| = 0$). Since it has access to all the annotations, this baseline is expected to represent an upper bound for all other strategies.

The second baseline consists in using the sparsely-annotated set $\mathcal{D}_s$ as if it was exhaustively annotated ($\mathcal{D}_{train} = \mathcal{D}_l \cup \mathcal{D}_s$). This strategy makes sense especially for moderately sparse datasets. Indeed, convolutional layers (as in U-Net) are able to cope with a bit of label noise given that gradients are averaged over feature maps to update the model parameters. Therefore, a bit of noise in certain parts of the images can be compensated by the feedback of ground truth labels in other locations. This baseline will be referred to as "$\mathcal{D}_l \cup \mathcal{D}_s$".

The third and last baseline, referred to as "$\mathcal{D}_l$ *only*", consists in not using at all the sparsely-annotated images during the training process (*i.e.* $\mathcal{D}_{train} = \mathcal{D}_l$).

Our self-training approach is of practical interest if it outperforms the two latter baselines. Moreover, the closer to the "*upper*" baseline the better.

Evaluation. We have built an evaluation protocol in order to ensure a fair comparison between the different strategies and the baselines. Ultimately, all approaches produce a segmentation model θ that will be the one evaluated. As an evaluation metric, we use the Dice score introduced in Equation 3 which requires a threshold T to turn the probability map produced by the model into a binary segmentation. To assess the performance of a model independently of the thresholding strategy, we pick T so that the Dice score is maximized on the test set. In other words, in order to determine the threshold, we apply the optimization procedure described in Equation 4 where $\mathcal{D}_a$ is the test set $\mathcal{D}_{\text{test}}$. The Dice score resulting from this optimization will be referred to as "Dice*". Obviously, in a context of extreme data scarcity, it will not be possible to tune the threshold this way because of the lack of a sufficiently large validation or test set. We will leave the design of fully automatic strategies to tune such threshold as future work but we believe that visually tuning such threshold on new images would be feasible in interactive applications where the segmentation models would be mostly used to assist the labeling of new images.

Every experiment and hyperparameters combination we evaluate is run with ten random seeds to evaluate the variability. The seed affects the dataset sparsity (n_l and ρ), model initialization, mini-batch sampling and data augmentation. We report Dice average and standard deviation over these random seeds.

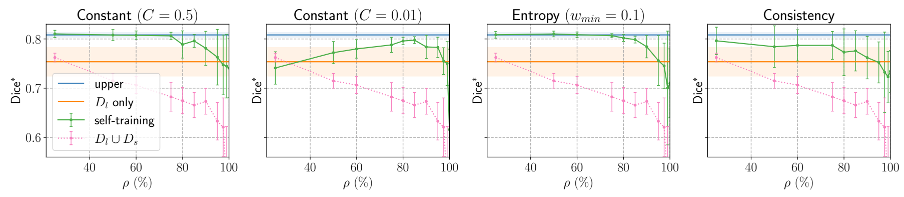
6 Results

6.1 Self-training performance at fixed n_l

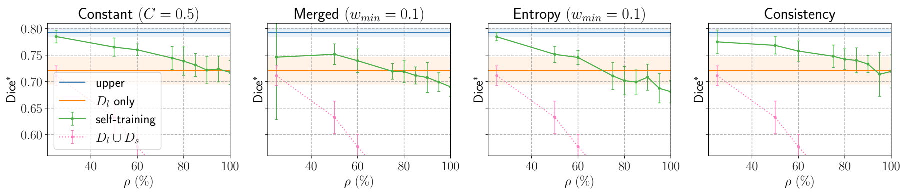
In order to study how our self-training approach performs under different data scarcity conditions, we have generated several versions of our datasets by varying ρ with n_l fixed and have run the baselines and different hyperparameters combinations on the generated datasets. As discussed in Section 3.1, we have used hard labels exclusively. The detailed hyperparameter combinations used in this section are provided in Supplementary Section A.

Results are shown for all three datasets in Figure 2. In general, self-training is always able to outperform significantly the "$\mathcal{D}_l$ *only*" and "$\mathcal{D}_l \cup \mathcal{D}_s$" baselines with a significantly reduced amount of sparse annotations (the exact value is dataset dependant, see below). Regarding the baselines, "*upper*" outperforms the two others. Moreover, using sparsely labeled images as if they were exhaustively labeled (*i.e.* $\mathcal{D}_l \cup \mathcal{D}_s$) appears not to be a good idea as it is outperformed by all self-training approaches and baselines in almost all scarcity conditions. The performance of this baseline increases as one adds more sparse annotations however and is able to catch up with the "$\mathcal{D}_l$ *only*" baseline in the lowest scarcity conditions validating the hypothesis presented in Section 5.

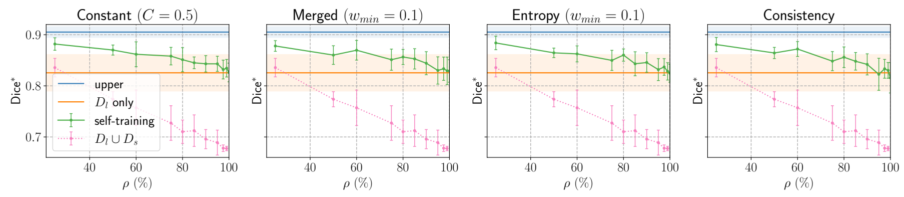
MoNuSeg. On this dataset, we can divide the analysis by differentiating three scarcity regimes: extreme ($\rho \in [95\%, 100\%]$), significant ($\rho \in [80\%, 90\%]$) and medium ($\rho \in [25\%, 75\%]$). Overall, most self-training approaches benefit from additional sparse annotations as their score increase when ρ decreases. This statement is true for all weighting strategies but the "*constant*" with $C = 0.1$ of which the performance plateau near $\rho = 85\%$, before decreasing as ρ decreases.



(a) MoNuSeg



(b) SegPC-2021



(c) GlaS

Fig. 2: Performance of our baselines and self-training approaches with different hyperparameters combinations with a varying ρ and a fixed labeled set size n_l. We only report the weighting schemes that show representative behaviour. Other weighting strategies are evaluated and reported in Supplementary Section C.

In the extreme regime, all self-training approaches exhibit high variance and are outperformed by the "$\mathcal{D}_l$ *only*" baseline, or yield comparable performance. In this situation, it appears to be better to work in a fully supervised fashion using only images from $\mathcal{D}_l$ rather then using our self-training approach. Indeed, it seems that the noise brought by the extreme annotation sparsity (or complete lack of annotation when $\rho = 100\%$) degrades the model significantly which cannot even make efficient use of the exhaustively labeled images anymore. For $\rho = 95\%$, two self-training approaches ("*constant*" with $C = 0.1$ and "*entropy*") are on average better then the baseline but variance is still high making it difficult to really conclude that they are more efficient.

The situation is reversed in the significant regime where most self-training approaches (except the "*consistency*" weighting strategy) outperform the "$\mathcal{D}_l$ *only*" baseline and variance decreases significantly as well. As for the "*upper*" baseline, it remains more efficient than self-training. For $\rho = 90\%$, the most efficient weighting strategy on average is "*constant*" with $C = 0.1$ which also exhibits the smallest variance of all the self-training approaches. We believe that such a low constant is particularly helpful to combat the noise brought by the high sparsity as pseudo-labeled pixels contribute way less during training. For $\rho = 85\%$ and 90%, the "*constant*" with $C = 0.1$ strategy plateaus whereas others catch up in terms of performance and variance decrease with the "*entropy*" and "*merged*" (plot for this strategy can be found in Supplementary Figure 1) approaches taking up the lead.

In the medium regime, three self-training approaches reach, and even slightly surpass, the upper baseline: "*constant*" with $C = 0.5$, "*entropy*" and "*merged*". This result is interesting because it means that our self-training approach is able to reach the performance of a fully supervised approach but using only $\sim 30\%$ of the original annotations (*i.e.* $\rho = 75\%$, approximately 5k annotations instead of 17k) which is a significant annotation budget saving. The approach "*constant* ($C = 0.1$)" decreases with ρ indicating that such a low C prevents the model to learn efficiently from the additional annotations (compared to the significant regime). This strategy even finished below the $\mathcal{D}_l \cup \mathcal{D}_s$ baseline at $\rho = 25\%$.

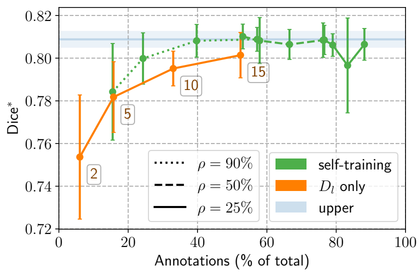
Overall, results on MoNuSeg are quite satisfactory. Although our approach is struggling in an extreme scarcity regime, it quickly catches up with the "*upper*" baseline as one adds more annotations to $\mathcal{D}_s$. In this case, the choice of weighting strategy matters and depends on the sparsity of the dataset.

SegPC-2021. Regarding the trend, our self-training approach behaves similarly on SegPC-2021 (see Figure 2b) compared to MoNuSeg: all self-training approaches without exception seem to benefit from additional annotations in $\mathcal{D}_s$. Moreover, the $\mathcal{D}_l \cup \mathcal{D}_s$ baseline is particularly inefficient and finishes just below the "$\mathcal{D}_l$ *only*" baseline at $\rho = 25\%$. However, in the extreme regime, the gap between self-training and the "$\mathcal{D}_l$ *only*" baseline is less than on MoNuSeg. The rate at which our approach improves over the "$\mathcal{D}_l$ *only*" is also slower as it takes a larger ρ (around 75%) for the performance of self-training to become significantly better than this baseline. The best-performing weighting strategies also differ. The best strategies overall are "*constant*" (with $C = 0.5$ or 1) and "*consistency*". The "*merged*" and "*entropy*" are worse than the others, although the latter catches up at $\rho = 25\%$. Only the "constant" and "entropy" strategies come close to catching up with the upper baseline but it takes proportionally more annotations compared to MoNuSeg as it happens around $\rho = 25\%$.

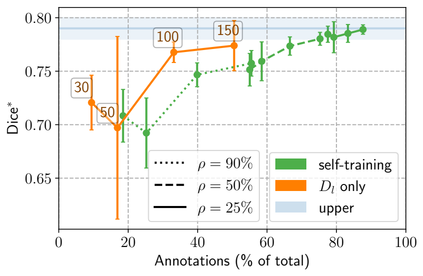
GlaS. On this dataset, all self-training approaches benefit from additional sparse annotations in $\mathcal{D}_s$. Compared to the "$\mathcal{D}_l$ *only*" baseline, the self-training approaches are never worse, even in the extreme scarcity regime, and it takes a ρ between 60% and 75% for self-training to become significantly better. Self-training is not able to catch up the "*upper*" baseline in this case.

6.2 Labeling a new dataset: sparsely or exhaustively?

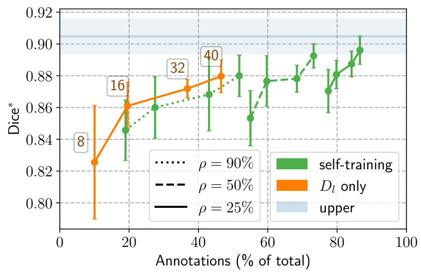
The fact that self-training is able to equal or outperform the "$\mathcal{D}_l$ *only*" and "*upper*" baselines suggests that it might be more interesting to consider an alternative annotation strategy to exhaustive labeling when annotating a new dataset.



(a) MoNuSeg , $n_l = \{2, 5, 10, 15\}$



(b) SegPC-2021 , $n_l = \{30, 50, 100, 150\}$



(c) GlaS , $n_l = \{8, 16, 32, 40\}$

Fig. 3: Self-training *vs.* the baselines for varying ρ and n_l. A point corresponds to 10 runs of a given approach (see line color) and some sparsity parameters, ρ (see line style) and n_l (see subcaptions, increasing from left to right on a constant ρ curve, see "$\mathcal{D}_l$ *only*" for example). A new dataset is generated for each run, with the corresponding n_l and ρ values. The x value of a point corresponds to the average percentage of annotations used by the 10 datasets, or, alternatively, to the annotation budget dedicated for labeling the dataset.

At fixed annotation budget, it might indeed be more interesting to combine sparse labeling and self-training rather than performing fully supervised training on an exhaustively labeled dataset (*i.e.* $\mathcal{D}_l$ only). To answer this question, we have conducted a set of experiments where we compare a self-training approach (entropy weighting strategy, $w_{min} = 0.1$) and the baselines all run against different sparsity regimes, varying both $\rho \in \{90\%, 50\%, 25\%\}$ and n_l (values are dataset specific). The results of these experiments are given in Figure 3 where the values of n_l we have used are also specified. In these plots, the performance of all methods are reported over a common metric, the percentage of annotations used, which can be equated with the annotation budget for creating the dataset.

Our experiments show very dataset-dependent results. On MoNuSeg, we observe that self-training outperforms supervised training for all tested budgets. This indicates that it would have been more interesting to sparsely annotate this

Table 2: Experiment on the Thyroid FNAB dataset. The self-training approach uses the "*entropy*" weighting strategy and $w_{min} = 0.1$.

Method	Dice*
Self-training	89.05 ± 0.85
$\mathcal{D}_l$ only	80.30 ± 5.39
$\mathcal{D}_l \cup \mathcal{D}_s$	83.62 ± 3.52

dataset. However, this conclusion does not hold for the other datasets as, within the same annotation budget, using "$\mathcal{D}_l$ *only*" outperforms self-training.

This experiment also allows to compare which labeling scheme is better for self-training: for a given annotation budget, is it better to favor a larger set $\mathcal{D}_l$ or to add more sparse annotations in $\mathcal{D}_s$? For MoNuSeg and SegPC-2021 , it appears that, for a similar annotation budget, self-training performance are comparable whatever the values of n_l and ρ. Therefore, for those datasets, it does not really matter if the annotation budget is spent for exhaustive or sparse labeling. For GlaS, however, there is a performance loss when switching from a lower ρ value to a higher (*e.g.* going from $(\rho, n_l) = (90\%, 40)$ to $(50\%, 8)$ in Figure 3c). It indicates that, it is more interesting to label images exhaustively rather than sparsely for this dataset.

At this point, the experiments in this section do not allow us to provide definitive guidelines on how to focus annotation efforts to achieve optimal performance on a new dataset, but at least, they show that it can be beneficial to sparsely annotate more images than to exhaustively annotate fewer images.

6.3 Experiments on Thyroid cytologyFNAB

The Thyroid FNAB dataset presents a great opportunity to test our method on a real case of sparsely labeled dataset. It is also interesting to note that this dataset is larger than the three public datasets used in this study (almost 5k images in total). In order to fit the sparsely-labeled settings we have introduced, we split the dataset in two subsets based on the nature of the annotations. The set $\mathcal{D}_l$ is assigned annotations of cell aggregates and $\mathcal{D}_s$ is assigned annotations of individual cells and nuclei. The motivation for this split is presented in Supplementary Section E.

Based on the results of Section 6.1, we have chosen to use the "*entropy*" weighting strategy with $w_{min} = 0.1$ for our self-training approach, as it provides consistently good results across datasets. We compare this approach with two of our three baselines: "$\mathcal{D}_l$ *only*" and "$\mathcal{D}_l \cup \mathcal{D}_s$". The "*upper*" baseline obviously cannot be evaluated because we do not have access to the complete ground truth. The resulting performances are given in Table 2.

We observe that our self-training approach significantly outperforms the two baselines and remains quite stable as its standard deviation is below 1%. This confirms the interest of self-training when working with a sparse dataset.

7 Conclusion

In this work, we have introduced a method based on self-training for training a deep binary segmentation model with sparsely labeled data. Using 4 datasets, including an actual sparsely labeled one, we have shown that the method could indeed make use of sparse annotations to improve model performance over using only exhaustively labeled data. For one of our datasets, our self-training approach using only 30% of the original training annotations is even able to reach performance comparable to using all of them in a supervised way.

In the future, we want to extend the method to multi-class segmentation and further study the impact of various training choices and hyperparameters (model complexity, weighting strategies, soft pseudo-labeling, *etc.*) that we could not explore due to time and computing resources constraints. We also want to further study how the type of dataset (variability in images, density of ground truth, large or small annotations, *etc.*) impacts the performance margins of self-training. Moreover, in this work, we have removed annotation randomly from the datasets. In practice, it is unlikely that the existing annotations are really randomly chosen and it would be interesting to study the effect of the labeling process. This work has mostly been focused on high scarcity conditions but self-training methods have also shown to be beneficial with very large labeled and unlabeled sets in other contexts. In computational pathology, whole slide images usually offer a great potential for a large pool of unlabeled data. Therefore, we would like to study how our method would perform in such a context. Finally, we would also like to explore how our self-training algorithm could be used in an interactive mode to assist new dataset labeling. Eventually, we want to implement the algorithm on the Cytomine [19] platform.

Acknowledgements Raphaël Marée is supported by the BigPicture EU Research and Innovation Action (Grant agreement number 945358). This work was partially supported by Service Public de Wallonie Recherche under Grant No. 2010235 - ARIAC by DIGITALWALLONIA4.AI. Computational infrastructure is partially supported by ULiège, Wallonia and Belspo funds. We would also like to thank Isabelle Salmon (ULB, Erasme Hospital, Belgium) and her team for providing the Thyroid FNAB dataset.

References

1. Bai, W., Oktay, O., Sinclair, M., Suzuki, H., Rajchl, M., Tarroni, G., Glocker, B., King, A., Matthews, P.M., Rueckert, D.: Semi-supervised learning for network-based cardiac mr image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 253–260. Springer (2017)
2. Bankhead, P., Loughrey, M.B., Fernández, J.A., Dombrowski, Y., McArt, D.G., Dunne, P.D., McQuaid, S., Gray, R.T., Murray, L.J., Coleman, H.G., et al.: Qupath: Open source software for digital pathology image analysis. Scientific reports **7**(1), 1–7 (2017)
3. Berg, S., Kutra, D., Kroeger, T., Straehle, C.N., Kausler, B.X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., Eren, K., Cervantes, J.I., Xu, B., Beuttenmueller, F., Wolny, A., Zhang, C., Koethe, U., Hamprecht, F.A., Kreshuk, A.: ilastik: interactive machine learning for (bio)image analysis. Nature Methods (Sep 2019). https://doi.org/10.1038/s41592-019-0582-9, `https://doi.org/10.1038/s41592-019-0582-9`
4. Bokhorst, J.M., Pinckaers, H., van Zwam, P., Nagtegaal, I., van der Laak, J., Ciompi, F.: Learning from sparsely annotated data for semantic segmentation in histopathology images. In: International Conference on Medical Imaging with Deep Learning–Full Paper Track (2018)
5. Fan, D.P., Zhou, T., Ji, G.P., Zhou, Y., Chen, G., Fu, H., Shen, J., Shao, L.: Inf-net: Automatic covid-19 lung infection segmentation from ct images. IEEE Transactions on Medical Imaging **39**(8), 2626–2637 (2020)
6. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. Advances in neural information processing systems **17** (2004)
7. Gupta, A., Gupta, R., Gehlot, S., Goswami, S.: Segpc-2021: Segmentation of multiple myeloma plasma cells in microscopic images. IEEE Dataport **1**(1), 1 (2021)
8. Jaiswal, A.K., Panshin, I., Shulkin, D., Aneja, N., Abramov, S.: Semi-supervised learning for cancer detection of lymph node metastases. arXiv preprint arXiv:1906.09587 (2019)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Koohbanani, N.A., Jahanifar, M., Tajadin, N.Z., Rajpoot, N.: Nuclick: a deep learning framework for interactive segmentation of microscopic images. Medical Image Analysis **65**, 101771 (2020)
11. Koohbanani, N.A., Unnikrishnan, B., Khurram, S.A., Krishnaswamy, P., Rajpoot, N.: Self-path: Self-supervision for classification of pathology images with limited annotations. IEEE Transactions on Medical Imaging **40**(10), 2845–2856 (2021)
12. Kumar, N., Verma, R., Anand, D., Zhou, Y., Onder, O.F., Tsougenis, E., Chen, H., Heng, P.A., Li, J., Hu, Z., et al.: A multi-organ nucleus segmentation challenge. IEEE transactions on medical imaging **39**(5), 1380–1391 (2019)
13. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. Acm transactions on graphics (tog) **22**(3), 277–286 (2003)
14. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. arXiv preprint arXiv:1610.02242 (2016)
15. Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML. vol. 3, p. 896 (2013)

16. Li, J., Yang, S., Huang, X., Da, Q., Yang, X., Hu, Z., Duan, Q., Wang, C., Li, H.: Signet ring cell detection with a semi-supervised learning framework. In: International conference on information processing in medical imaging. pp. 842–854. Springer (2019)
17. Li, J., Speier, W., Ho, K.C., Sarma, K.V., Gertych, A., Knudsen, B.S., Arnold, C.W.: An em-based semi-supervised deep learning approach for semantic segmentation of histopathological images from radical prostatectomies. Computerized Medical Imaging and Graphics **69**, 125–133 (2018)
18. Maninis, K.K., Caelles, S., Pont-Tuset, J., Van Gool, L.: Deep extreme cut: From extreme points to object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 616–625 (2018)
19. Marée, R., Rollus, L., Stévens, B., Hoyoux, R., Louppe, G., Vandaele, R., Begon, J.M., Kainz, P., Geurts, P., Wehenkel, L.: Collaborative analysis of multi-gigapixel imaging data using cytomine. Bioinformatics **32**(9), 1395–1401 (2016)
20. Peikari, M., Salama, S., Nofech-Mozes, S., Martel, A.L.: A cluster-then-label semi-supervised learning approach for pathology image classification. Scientific reports **8**(1), 1–13 (2018)
21. Peng, J., Wang, Y.: Medical image segmentation with limited supervision: A review of deep network models. IEEE Access (2021)
22. Pham, H., Dai, Z., Xie, Q., Le, Q.V.: Meta pseudo labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11557–11568 (2021)
23. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
24. Rother, C., Kolmogorov, V., Blake, A.: " grabcut" interactive foreground extraction using iterated graph cuts. ACM transactions on graphics (TOG) **23**(3), 309–314 (2004)
25. Shaw, S., Pajak, M., Lisowska, A., Tsaftaris, S.A., O'Neil, A.Q.: Teacher-student chain for efficient semi-supervised histology image classification. arXiv preprint arXiv:2003.08797 (2020)
26. Sirinukunwattana, K., Pluim, J.P., Chen, H., Qi, X., Heng, P.A., Guo, Y.B., Wang, L.Y., Matuszewski, B.J., Bruni, E., Sanchez, U., et al.: Gland segmentation in colon histology images: The glas challenge contest. Medical image analysis **35**, 489–502 (2017)
27. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Advances in Neural Information Processing Systems **33**, 596–608 (2020)
28. Su, H., Shi, X., Cai, J., Yang, L.: Local and global consistency regularized mean teacher for semi-supervised nuclei classification. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 559–567. Springer (2019)
29. Tajbakhsh, N., Jeyaseelan, L., Li, Q., Chiang, J.N., Wu, Z., Ding, X.: Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. Medical Image Analysis **63**, 101693 (2020)
30. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. Advances in neural information processing systems **30** (2017)

31. Tellez, D., Balkenhol, M., Otte-Höller, I., van de Loo, R., Vogels, R., Bult, P., Wauters, C., Vreuls, W., Mol, S., Karssemeijer, N., et al.: Whole-slide mitosis detection in h&e breast histology using phh3 as a reference to train distilled stain-invariant convolutional networks. IEEE transactions on medical imaging **37**(9), 2126–2136 (2018)
32. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10687–10698 (2020)
33. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: 33rd annual meeting of the association for computational linguistics. pp. 189–196 (1995)
34. Zhu, Y., Zhang, Z., Wu, C., Zhang, Z., He, T., Zhang, H., Manmatha, R., Li, M., Smola, A.: Improving semantic segmentation via self-training. arXiv preprint arXiv:2004.14960 (2020)